

# Robust Multi-Agent Reinforcement Learning via Adversarial Regularization: Theoretical Foundation and Stable Algorithms

Alexander Bukharin<sup>1</sup>, Yan Li<sup>1</sup>, Yue Yu<sup>1</sup>, Qingru Zhang<sup>1</sup>, Zehui Chen<sup>2</sup>, Simiao Zuo<sup>3</sup>, Chao Zhang<sup>1</sup>, Songan Zhang<sup>4</sup>, and Tuo Zhao<sup>1</sup>

<sup>1</sup>Georgia Institute of Technology

<sup>2</sup>Google

<sup>3</sup>Microsoft

<sup>4</sup>Ford Motor Company

October 18, 2023

## Abstract

Multi-Agent Reinforcement Learning (MARL) has shown promising results across several domains. Despite this promise, MARL policies often lack robustness and are therefore sensitive to small changes in their environment. This presents a serious concern for the real world deployment of MARL algorithms, where the testing environment may slightly differ from the training environment. In this work we show that we can gain robustness by controlling a policy's Lipschitz constant, and under mild conditions, establish the existence of a Lipschitz and close-to-optimal policy. Based on these insights, we propose a new robust MARL framework, ERNIE, that promotes the Lipschitz continuity of the policies with respect to the state observations and actions by adversarial regularization. The ERNIE framework provides robustness against noisy observations, changing transition dynamics, and malicious actions of agents. However, ERNIE's adversarial regularization may introduce some training instability. To reduce this instability, we reformulate adversarial regularization as a Stackelberg game. We demonstrate the effectiveness of the proposed framework with extensive experiments in traffic light control and particle environments. In addition, we extend ERNIE to mean-field MARL with a formulation based on distributionally robust optimization that outperforms its non-robust counterpart and is of independent interest. Our code is available at <https://github.com/abukharin3/ERNIE>.

## 1 Introduction

In the past decade advances in deep neural networks and greater computational power have led to great successes for Multi-Agent Reinforcement Learning (MARL), which has achieved success on a wide variety of multi-agent decision-making tasks ranging from traffic light control [Wiering, 2000] to StarCraft [Vinyals et al., 2019]. However, while much effort has been devoted to

applying MARL to new problems, there has been limited work regarding the robustness of MARL policies.

Despite the limited attention paid to robustness, it is essential for MARL policies to be robust. Most MARL policies are trained in a fixed environment. Since these policies are trained solely to perform well in that environment, they may perform poorly in an environment with slightly different transition dynamics than the training environment. In addition, while agents are fed with exact state information in training, MARL policies deployed in the real world can receive inaccurate state information (e.g., due to sensor error). Finally, even a single agent acting maliciously or differently than expected can cause a chain reaction that destabilizes the whole system. These phenomena cause significant concern for the real-world deployment of MARL algorithms, where the environment dynamics and observation noise can change over time. We observe that even when the change in the environment’s dynamics is small, the performance of MARL algorithms can deteriorate severely (See an example in Section 6). Thus there is an emerging need for MARL algorithms that are robust to changing transition dynamics, observation noise, and changing behavior of agents.

Although many robust RL methods have been proposed for the single agent case, three major barriers prevent their use for MARL. Theoretically, it is not clear if or when such methods can work for MARL. Methodologically, it is not straightforward to apply single agent robust RL methods to MARL, as single agent methods may not consider the interactions between several agents. Algorithmically, single agent robust RL algorithms are often unstable, and may not perform well when applied to inherently unstable MARL training. Therefore to learn robust MARL policies, we provide theoretical, methodological, and algorithmic contributions.

**Theory.** Theoretically, we first show that when the transition and reward function are smooth, a policy’s value function is also smooth. In our experiments, we show that this assumption can serve as a useful prior knowledge, even if the transition function is not smooth in every state. Second, we prove that a smooth and close-to-optimal policy exists in any such environment. Third, we show that a policy’s robustness is inversely proportional to its Lipschitz constant *with no smoothness assumption on the environment’s smoothness*. These observations advocate for using smoothness as an inductive bias to not only reduce the policy search space, but simultaneously improve the robustness of the learned policy. Finally, we prove that large neural networks are capable of approximating the target policy or Q functions with smoothness guarantees. These findings give us the key insight that in order to learn robust and high-performing deep MARL policies, we should enforce the policies’ smoothness.

**Method.** Based on these findings, we propose a new framework – advErsarially Regularized multiageNt reInforcement lEarning (ERNIE), that applies adversarial training to learn smooth and robust MARL policies in a principled manner. In particular, we develop an adversarial regularizer to minimize the discrepancy between each policy’s output given a perturbed observation and a non-perturbed observation. This adversarial regularization gives two main benefits: Lipschitz continuity and rich data augmentation with adversarial examples. The adversarial regularization encourages the learned policies to be Lipschitz continuous, improving robustness. Augmenting

the data with adversarial examples further provides robustness against environment changes. Specifically, new scenarios emerge when the environment changes, and data augmentation with adversarial examples provides a large coverage of these scenarios as long as the environment change is small. Adapting to adversarial examples during training ensures that the agents will perform reasonably even in the worst case.

To further provide robustness against the changing behaviors of a few malicious agents, we propose an extension of ERNIE that minimizes the discrepancy between the global Q-function with maliciously perturbed joint actions and non-perturbed joint actions. This regularizer encourages the policies to produce stable outputs even when a subset of agents acts sub-optimally, therefore granting robustness. Such robustness has not been considered in previous works.

**Algorithm.** We find that adversarial regularization can improve robust performance [Shen et al., 2020]. However, adversarial regularization can also be unstable. More concretely, conventional adversarial regularization can be formulated as a zero-sum game where the defender (the policy) and attacker (the perturbation) hold equal positions and play against each other. In this case, a small change in the attacker’s strategy may result in a large change for the defender, rendering the problem ill-conditioned. Coupled with the already existing stability issues that come with training MARL algorithms, this instability issue greatly reduces the power of adversarial regularization methods for MARL.

To address this issue, we reformulate adversarial training as a Stackelberg game. In a Stackelberg game, the leader (defender) has the advantage as it knows how the follower (attacker) will react to its actions and can act accordingly. This advantage essentially makes the optimization problem smoother for the defender, leading to a more stable training process.

**Extension to Mean-field MARL.** We further demonstrate the general applicability of ERNIE by developing its extension to robustify mean-field MARL algorithms. The mean-field approximation has been widely received as a practical strategy to scale up MARL algorithms while avoiding the curse of many agents [Wang et al., 2020]. However, as mean-field algorithms are applied to real-world problems, it is essential to develop robust versions. To facilitate policy learning that is more robust, we extend ERNIE to mean-field MARL with a formulation based on distributionally robust optimization [Delage and Ye, 2010, Goh and Sim, 2010].

To demonstrate the effectiveness of the proposed framework, we conduct extensive experiments that evaluate the robustness of ERNIE on traffic light control and particle environment tasks. Specifically, we evaluate the robustness of MARL policies when the evaluation environment deviates from the training environment. These deviations include observation noise, changing transition dynamics, and malicious agent actions. The results show that while state-of-the-art MARL algorithms are sensitive to small changes in their environment, the ERNIE framework enhances the robustness of these algorithms without sacrificing efficiency.

**Contributions.** We remark that adversarial regularization has been developed for single-agent RL, but never for MARL [Shen et al., 2020]. Our contribution in this paper has four aspects: (1) advances in theoretical understanding (2) development of new regularizers for MARL (3) new

algorithms for stable adversarial regularization in MARL (4) comprehensive experiments in a number of environments.

## 2 Background

In this section, we introduce the necessary background for MARL problems together with related literature. We consider the setting of *cooperative MARL*, where agents work together to maximize a global reward.

- **Cooperative Markov Games.** We consider a partially observable Markov game  $\langle \mathcal{S}, \mathcal{O}^N, \mathcal{A}^N, \mathcal{P}, \mathcal{R}, N, \gamma \rangle$  in which a set of agents interact within a common environment. We let  $\mathcal{S} \subseteq \mathbb{R}^S$  denote the global state space,  $\mathcal{O} \subseteq \mathbb{R}^O$  denote the observation space for each of the  $N$  agents,  $\mathcal{A} \subseteq \mathbb{R}^A$  denote the action space,  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$  denote the transition kernel,  $\gamma$  denotes the discount factor, and  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}^N$  denotes the reward function. At every time step  $t$ , each of the  $N$  agents selects an action according to its policy, which can be stochastic or deterministic. Then, the system transitions to the next state according to the transition kernel and each agent receives a reward  $r_{i,t}$ . We denote the global reward at time  $t$  as  $r_t^g$ . The goal of each agent is to find a policy that maximizes the discounted sum of its own reward,  $\sum_{t \geq 0} \gamma^t r_{i,t}$ .

- **Robust RL.** In recent years many single agent robust RL techniques have been proposed. Most of these methods use information about the underlying simulator to train agents over a variety of relevant environment settings [Morimoto and Doya, 2005, Pinto et al., 2017, Abdullah et al., 2019, Pan et al., 2019, Wang and Zou, 2022]. Although these methods can provide robustness against a wide range of environment changes, they suffer from long training times and require expert knowledge of the underlying simulator, which is not practical. Another direction of research focuses on perturbation based methods [Shen et al., 2020, Zhang et al., 2020]. Perturbation based methods train the policy to be robust to input perturbations, encouraging the policy to act reasonably in perturbed or previously unseen states. Kumar et al. [2022] certify robustness by adding smoothing noise to the state; it is not clear how this affects the learned policy's optimality. Another related line of work [Iyengar, 2005, Nilm and El Ghaoui, 2005, Panaganti et al., 2022, Li et al., 2022] studies robust markov decision processes and provides a principled way to learn robust policies. However, such methods often require strict assumptions on the perturbation/uncertainty. Inspiring our work, Shen et al. [2020] proposes to learn a smooth policy in single agent RL, but they do so to reduce training complexity rather than increase robustness and provide no theoretical justification for their method. Instead, we theoretically connect smoothness to robustness, extend perturbation based methods to MARL, and develop a more stable perturbation computation technique, and develop an extension to mean-field MARL.

- **Robust MARL.** Recently, some works have studied the robustness of MARL systems. Lin et al. [2020] studies how to attack MARL systems and finds that MARL systems are vulnerable to attacks on even a single agent. Zhang et al. [2021] develop a framework to handle MARL with model uncertainty by formulating MARL as a robust Markov game. However, their proposed method only considers uncertainty in the reward function, while this article focuses on robust-

ness to observation noise and changing transition dynamics. Li et al. [2019a] modify the MADDPG algorithm to consider the worst-case actions of the other agents in continuous action spaces with the M3DDPG algorithm. M3DDPG aims to grant robustness against the actions of other agents, which is less general than the robustness against observation noise, changing transition dynamics, and malicious agents that our method aims for. Wang et al. [2022] consider robustness against uncertain transition dynamics, but their algorithm is not applied to deep MARL. More recently, He et al. [2023], Han et al. [2022] introduces the concept of robust equilibrium and proposes to learn an adversarial policy to perturb each agent’s observations. Finally Zhou et al. [2023] propose to learn robust policies by minimizing the cross-entropy loss between agent’s actions in non-perturbed states and perturbed states.

The ERNIE framework is also related to several existing works which use similar adversarial training methods but target different domains such as trajectory optimization [Zhao et al., 2022], semi-supervised learning [Miyato et al., 2018, Hendrycks et al., 2019, Zuo et al., 2022], fine-tuning language models [Jiang et al., 2020, Yu et al., 2021], and generalization in supervised learning [Zuo et al., 2021].

### 3 From Lipschitz Continuity to Robustness

This section presents the theoretical motivation for our algorithm by showing that Lipschitzness (smoothness) serves as a natural way to gain robustness, while reducing the policy search space. We start by observing that certain natural environments exhibit smooth transition and reward functions, especially when the transition dynamics are governed by physical laws (e.g., MuJuCo environment [Todorov et al., 2012], Pendulum [Brockman et al., 2016]).<sup>1</sup> Formally, this is stated as the following.

**Definition 3.1.** Let  $\mathcal{S} \subseteq \mathbb{R}^d$ . We say the environment is  $(L_r, L_{\mathbb{P}})$ -smooth, if the reward function  $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ , and the transition kernel  $\mathbb{P} : \mathcal{S} \times \mathcal{S} \times \mathcal{R}$  satisfy

$$|r(s, a) - r(s', a)| \leq L_r \|s - s'\| \quad \text{and} \quad \|\mathbb{P}(\cdot | s, a) - \mathbb{P}(\cdot | s', a)\|_1 \leq L_{\mathbb{P}} \|s - s'\|,$$

for  $(s, s', a) \in \mathcal{S} \times \mathcal{S} \times \mathcal{A}$ .  $\|\cdot\|$  denotes a metric on  $\mathbb{R}^d$ . We say a policy  $\pi$  is  $L_{\pi}$ -smooth if

$$\|\pi(\cdot | s) - \pi(\cdot | s')\|_1 \leq L_{\pi} \|s - s'\|.$$

Without loss of generality, we assume  $|r(s, a)| \leq 1$  for any  $(s, a) \in \mathcal{S} \times \mathcal{A}$ . We then present our theory. Due to the space limit, we defer all technical details to the appendix.

- **From smooth environments to smooth values.** We proceed to show that if the environment is smooth, then the value functions for smooth policies are also smooth.

**Theorem 3.1.** Suppose the environment is  $(L_r, L_{\mathbb{P}})$ -smooth. Then the Q-function of any policy  $\pi$ , defined as

$$Q^{\pi}(s, a) = \mathbb{E}^{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right], \quad \forall (s, a),$$

---

<sup>1</sup>See Remark 3.2 for discussions when the smoothness property holds approximately.

is Lipschitz continuous in the first argument. That is,

$$|Q(s, a) - Q(s', a)| \leq L_Q \|s - s'\|, \quad (1)$$

where  $L_Q := L_r + \gamma L_{\mathbb{P}}/(1 - \gamma)$ . Suppose in addition the policy is  $L_\pi$ -smooth. Then the value function, defined as

$$V^\pi(s) = \mathbb{E}^\pi \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s \right], \forall s,$$

is Lipschitz continuous. That is,

$$|V^\pi(s) - V^\pi(s')| \leq L_V \|s - s'\|,$$

where  $L_V := L_\pi/(1 - \gamma) + L_Q$ .

In view of Theorem 3.1, it is clear that whenever the environment and the policy are smooth, then the value functions are also smooth. A natural and important follow-up question to ask is whether this claim holds in the reverse direction. More concretely, we ask whether it is reasonable to seek a policy that is also smooth with respect to the state while maximizing the reward. If the claim holds true, then seeking a smooth policy can serve as an efficient and unbiased prior knowledge, that can help us reduce the policy search space significantly, while still guaranteeing that we are searching for high-performing policies.

- **Existence of smooth and nearly-optimal policies.** The following result shows that for any  $\epsilon > 0$ , there exists an  $\epsilon$ -optimal policy that is  $\mathcal{O}(L_Q/\epsilon)$  smooth, where  $L_Q$  defined in Theorem 3.1 only depends on the smoothness of reward and transition. This structural observation naturally suggests seeking a smooth policy for smooth environments.

**Theorem 3.2.** Suppose the environment is  $(L_r, L_{\mathbb{P}})$ -smooth. Then for any  $\epsilon > 0$ , there exists an  $\epsilon$ -optimal policy  $\pi$  that is also smooth, i.e.,

$$V^*(s) - V^\pi(s) \leq \frac{2\epsilon}{1-\gamma}, \quad \forall s \in \mathcal{S} \text{ and } \|\pi(\cdot|s) - \pi(\cdot|s')\|_1 \leq |\mathcal{A}| \log |\mathcal{A}| L_Q \|s - s'\|/\epsilon,$$

where  $L_Q$  is defined as in Theorem 3.1.

Notably, the proof of Theorem 3.2 relies on the key observation that any smooth  $Q$ -function satisfying (1) can be fed into the softmax operator, which induces a smooth policy. This observation also provides a way for value-based methods (e.g., Q-learning) to learn a smooth policy. Namely, one can first learn a smooth surrogate of the optimal  $Q$ -function, and then feed the learned surrogate into the softmax operator to induce a close-to-optimal policy that is also smooth.

- **Robustness against observation noise with smooth policies.** We have so far established that smooth policies naturally exist in a smooth environment as close-to-optimal policies, and thus smoothness serves as a strong prior for policy search. We will further demonstrate that the benefits of smooth policy go beyond boosting learning efficiency, by bringing in the additional advantage of robustness against observation uncertainty.

**Theorem 3.3.** Let  $\pi(a|s)$  be  $L_\pi$ -smooth policy. For any perturbation sequence  $\{\delta_s^t\}_{t \geq 0, s \in \mathcal{S}}$ , define a perturbed policy (non-stationary)  $\tilde{\pi} = \{\tilde{\pi}_t\}_{t \geq 0}$  by

$$\tilde{\pi}_t(a|s) = \pi(a|s + \delta_s^t),$$

with  $\|\delta_s^t\| \leq \epsilon$  for all  $t \geq 0$ . Accordingly, define the value function of the non-stationary policy  $\tilde{\pi}$

$$V^{\tilde{\pi}}(s) = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_t \sim \tilde{\pi}_t(\cdot | s_t + \delta_{s_t}^t), s_{t+1} \sim \mathbb{P}(\cdot | s_t, a_t) \right].$$

Then we have  $|V^\pi(s) - V^{\tilde{\pi}}(s)| \leq \frac{2L_\pi\epsilon}{(1-\gamma)^2}$ . Similarly, we have

$$|Q^\pi(s, a) - Q^{\tilde{\pi}}(s, a)| \leq \frac{2L_\pi\epsilon}{(1-\gamma)^2},$$

where  $Q^{\tilde{\pi}}$  is defined similarly as  $V^{\tilde{\pi}}$ .

Theorem 3.3 establishes the following fact: for a discounted MDP with finite state and finite action space, the value of the policy when providing the perturbed state is close to the value of the policy when given the non-perturbed state, provided the policy is Lipschitz continuous in its state. As an important implication, the learned smooth policy will be robust in the state observation, in the sense that the accumulated reward will not deteriorate much when noisy, or even adversarially constructed state observations are given to the policy upon decision making.

We emphasize that Theorem 3.3 holds without any smoothness assumption on the transition or the reward function. It should also be noted that there are various notions of robustness in MDP, e.g., robustness against changes in transition kernel [Ruszczyński, 2010, Li et al., 2022], which we defer as future investigations.

Before we conclude this section, we briefly remark on certain generality of our discussion.

*Remark 3.1 (Applicability to MARL).* The discussions in this section do not depend on the size of the state space, and apply to the multi-agent setting without any change. To see this, note that our discussion holds for any discrete state-action space. Setting  $\mathcal{S}$  as the joint state space and  $\mathcal{A}$  as the joint action space, then the obtained results trivially carry over to the cooperative MARL setting.

*Remark 3.2 (Environments with approximate smoothness).* Many environments are partially smooth, in the sense that the transition or the reward is non-smooth only on a small fraction of the state space. Typical examples include the Box2D environment [Brockman et al., 2016], where the agent receives smooth reward when in non-terminal states (airborne for Lunar Lander), and receives a lump-sum reward in the terminal state (land/crash) – a vanishing fraction of the entire state space. Given the environment being largely smooth, it should be expected that for most states the optimal policy is locally smooth. Consequently, inducing a smoothness prior serves as a natural regularization to constrain the search space when solving these environments, without incurring a large bias.

*Remark 3.3 (Non-smooth environments).* From the perspective of robust statistics, achieving robustness often necessitates a certain level of smoothness in the learned policy, regardless of the smoothness of the optimal policy. In scenarios where the environment itself is non-smooth, the optimal policy can also be non-smooth. However, it is important to note that such non-smooth optimal policies are typically not robust. This means that by trading-off between the approximation bias and robustness, the smooth policy learnt by our method has the potential to outperform non-smooth policies in perturbed environments.

## 4 Wide Networks

Section 3 tells us that smooth and close to optimal policies exist under certain conditions and ERNIE provides algorithms to find them. Now, a practical question remains: can neural networks be used to learn such policies? We show that as long as the width is sufficiently large, there exists a neural network with the desired optimality and smoothness properties. This finding further supports ERNIE’s deployment as a tool for practical deep MARL.

Before we continue with further analysis, we will first introduce some necessary preliminaries. Specifically, we consider the Sobolev space, which contains a class of smooth functions [Brezis and Brézis, 2011].

**Definition 4.1.** Given  $\alpha \geq 0$  and domain  $\Omega \subset \mathbb{R}^d$ , we define the Sobolev space  $W^{\alpha,\infty}(\Omega)$  as

$$W^{\alpha,\infty}(\Omega) = \left\{ f \in L^\infty(\Omega) : D^\alpha f \in L^\infty(\Omega), \forall |\alpha| \leq \alpha \right\},$$

where  $D^\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \dots \partial x_D^{\alpha_D}}$  with multi-index  $\alpha = [\alpha_1, \dots, \alpha_D]^\top \in \mathbb{N}^D$ .

For  $f \in W^{\alpha,\infty}(\Omega)$ , we define its Sobolev norm as

$$\|f\|_{W^{\alpha,\infty}(\Omega)} = \max_{|\alpha| \leq \alpha} \|D^\alpha f\|_{L^\infty(\Omega)}$$

The Sobolev space has been widely investigated in the existing literature on function approximation of neural networks. For a special case  $\alpha = 1$ ,  $\|f\|_{W^{1,\infty}} < \infty$  implies both the function value and its gradient are bounded.

We consider an  $L$ -layer ReLU neural network

$$f(x) = W_L \cdot \sigma(\dots \sigma(W_1 s + b_1) \dots) + b_L, \quad (2)$$

where  $W_1, \dots, W_L$  and  $b_1, \dots, b_L$  are weight matrices and intercept vectors of proper sizes, respectively, and  $\sigma(\cdot) = \max\{\cdot, 0\}$  denotes the entry-wise ReLU activation. We denote  $\mathcal{F}$  as a class of neural networks:

$$\begin{aligned} \mathcal{F}(L, p) = & \{ f \mid f(x) \text{ in the form (2) with } L \text{-layers} \\ & \text{and width bounded by } p \}. \end{aligned} \quad (3)$$

We next present the function approximation results.

**Theorem 4.1** (Function approximation with Lipschitz continuity). *Suppose that the target function  $f^*$  satisfies*

$$f^* \in W^{\alpha, \infty}(\Omega) \quad \text{and} \quad \|f^*\|_{W^{\alpha, \infty}(\Omega)} \leq 1$$

for some  $\alpha \geq 2$ . Given a pre-specified approximation error  $\epsilon$ , there exists a neural network  $\tilde{f} \in \mathcal{F}(L, p)$  with  $L = \tilde{O}(\log(\epsilon^{-1}))$  and  $p = \tilde{O}(\epsilon^{-\frac{d}{\alpha-1}})$ , such that

$$\|\tilde{f} - f^*\|_\infty \leq \epsilon \quad \text{and} \quad \|\tilde{f}\|_{\text{Lip}} \leq 1 + \sqrt{d} \epsilon^{1-1/\alpha},$$

where  $\tilde{O}$  hides some negligible constants or log factors and  $\|\tilde{f}\|_{\text{Lip}}$  denotes the Lipschitz constant of  $\tilde{f}$ .

For reinforcement learning,  $f^*$  in Theorem 4.1 can be viewed as either the near-optimal smooth policy  $\pi^*$  or optimal smooth action-value function  $Q^*$ , and the input can be viewed as the state  $s$  or the state-action pair  $(s, a)$ . As can be seen, a wider neural network not only better approximates a smooth target function  $f^*$  well, but also further reduces the upper bound of its Lipschitz constant, which leads to a more robust policy. Moreover, we can certify the existence of a neural network  $\tilde{f}$  such that  $\|\tilde{f}\|_{\text{Lip}}$  is below 2, given a sufficient width  $p = \tilde{O}\left(d^{\frac{d\alpha}{2(\alpha-1)^2}}\right)$ . This result indicates that when training policies with the ERNIE algorithm, we should use wide neural networks.

## 5 Method

Section 3 shows that promoting smoothness leads to both robust and high-performing policies for smooth environments, which can be achieved by sufficiently wide neural networks. Based on this insight, we propose our robust MARL framework, advErsarially Regularized multiageNt reInforcement lEarning (ERNIE).

### 5.1 Learning Robust Policy with ERNIE

Section 3 shows that the robustness of a policy depends on its Lipschitz constant. Therefore, in ERNIE we propose to control the Lipschitz constant of each policy with adversarial regularization.

Given a policy  $\pi_{\theta_k}$ , where  $k$  is the agent index, the ERNIE regularizer is defined by

$$R_\pi(o_k; \theta_k) = \max_{\|\delta\| \leq \epsilon} D(\pi_{\theta_k}(o_k + \delta), \pi_{\theta_k}(o_k)). \quad (4)$$

Here  $\delta(o_k, \theta_k)$  is a perturbation adversarially chosen to maximize the difference between the policy's output for the perturbed observation  $o_k + \delta(o_k, \theta_k)$  and the original observation  $o_k$ . In this case  $\epsilon$  controls the perturbation strength and  $\|\cdot\|$  is usually taken to be the  $\ell_2$  or  $\ell_\infty$  norm. Note that  $R_\pi(o_k; \theta_k)$  essentially measures the local Lipschitz smoothness of policy function  $\pi_\theta$  around the observation  $o_k$ , defined in metric  $D(\cdot, \cdot)$ . Therefore minimizing  $R_\pi(o_k; \theta_k)$  will encourage the policy to be smooth.

Regularization Eq. (4) allows straightforward incorporation into MARL algorithms that directly perform policy search. For actor-critic based policy gradient methods, the regularizer Eq. (4) can

be directly included into the objective for updating the actor (policy) networks. When optimizing stochastic policies (e.g., MAPPO [Yu et al., 2022]),  $D$  can be taken to be the KL divergence and for deterministic policies (e.g., MADDPG [Lowe et al., 2017] or Q-learning [Tsitsiklis and Van Roy, 1996]), we set  $D$  to be the  $\ell_p$  norm.

More concretely, let  $\mathcal{L}(\theta)$  denote the policy optimization objective, i.e., the negative weighted value function of the policy. We then augment  $\mathcal{L}(\theta)$  with Eq. (4), and minimize the regularized objective

$$\min_{\theta} \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda \sum_{n=1}^N \mathbb{E}_{\pi_n} [R_{\pi}(o_n; \theta_n)], \quad (5)$$

where  $\lambda$  is a hyperparameter. We remark that Shen et al. [2020] has explored similar regularization for single-agent RL (with a goal of improving sample efficiency), but as we explain in sections 5.2, 5.3, and 5.4, successful application to MARL robustness is highly non-trivial.

## 5.2 Stackelberg Training with Differentiable Adversary

Although accurately solving Eq. (5) will result in a high-performing and robust policy, we note that Eq. (5) is a nonconvex-nonconcave minimax problem. In practice, we can use multiple steps of projected gradient ascent to approximate the worse-case state perturbation  $\delta(o_k, \theta_k)$ , followed by one-step gradient descent for updating the policies/Q-function. Even though this optimization method already significantly improves robustness over the baseline algorithms, we observe that the training process could be quite unstable. We hypothesize that the intrinsic instability of MARL algorithms due to simultaneous updates of multiple agents is greatly amplified by the non-smooth landscape of adversarial regularization.

To promote a more stable training process, we propose to reformulate adversarial training in ERNIE as a Stackelberg game. The reformulation defines adversarial regularization as a leader-follower game [Von Stackelberg, 2010]:

$$\begin{aligned} R_{\pi}(o, \delta^K_{\theta}(o); \theta) &= D(\pi_{\theta}(o + \delta^K(o, \theta)), \pi_{\theta}(o)) \\ \text{s.t. } \delta^K(o, \theta) &= \underbrace{U_{\theta} \circ U_{\theta} \circ \dots \circ U_{\theta}}_{\text{K-fold composition}} (\delta^0(o, \theta)). \end{aligned} \quad (6)$$

Here  $\circ$  denotes the operator composition (i.e  $f \circ g = f(g(\cdot))$ ), and

$$\delta^{k+1}(o, \theta) = U_{\theta}(\delta^k(o, \theta)) = \delta^k(o, \theta) + \eta \nabla_{\delta} D(\pi_{\theta}(o + \delta^k(o, \theta)), \pi_{\theta}(o))$$

is a one-step gradient ascent for maximizing the divergence of the perturbed and original observation.

Compared to the vanilla adversarial regularizer in Eq. (4), the perturbation  $\delta$  is treated as a function of the model parameter  $\theta$ . This formulation allows the leader ( $\theta$ ) to anticipate the action of the follower ( $\delta$ ), since the follower's response given observation  $o$  is fully specified by  $\delta^K(o, \theta)$ . This structural anticipation effectively produces an easier and smoother optimization problem for

the leader ( $\theta$ ), whose gradient, termed the Stackelberg gradient, can be readily computed by

$$\frac{\partial R_\pi(o, \delta_\theta^K(o); \theta)}{\partial \theta} = \underbrace{\frac{\partial R_\pi(o, \delta^K, \theta)}{\partial \theta}}_{\text{leader}} + \underbrace{\frac{\partial R_\pi(o, \delta^K(\theta), \theta)}{\partial \delta^K(\theta)} \frac{\delta^K(\theta)}{\partial \theta}}_{\text{leader-follower interaction}}$$

Note that the gradient used in Eq. (5) only contains the “leader” term, such that interaction between the model  $\theta$  and the perturbation  $\delta$  is ignored. The computation of the Stackelberg gradient can be reduced to Hessian vector multiplication using finite difference method [Pearlmutter and Siskind, 2008], which only requires two backpropogations and extra  $\mathcal{O}(d)$  complexity operation. Thus no significant computational overhead is introduced for solving Eq. (6).

The benefit of Stackelberg training for MARL is twofold. First, a smoother optimization problem results in a more stable training process. This extra stability is essential given the inherent instability of MARL training. Second, giving the policy  $\theta$  priority over the attack  $\delta$  during the training process allows for a better training data fit than normal adversarial training allows. This better fit allows the MARL policies trained with Stackelberg training to perform better in lightly perturbed environments than those trained with normal adversarial regularization.

### 5.3 Robustness against Malicious Actions

Given the complex interactions of agents within of a multi-agent system, a robust policy for any given agent should meet the criterion that the action made is not overly dependent on any small subset of agents. This is particularly the case when the agents are homogeneous in nature [Wang et al., 2020, Li et al., 2021], and thus there should be no notion of *coreset agents* in the decision-making process that could heavily influence the actions of other agents. We proceed to show how ERNIE could be adopted to induce such a notion of robustness.

The core idea of ERNIE for this scenario is to encourage policy/Q-function smoothness with respect to *joint actions*. Similar to our treatment in Eq. (4), we now seek to promote learning a Q-function that yields a consistent value when perturbing the actions for any small subset of agents. Specifically, for discrete action space, we define a regularizer on the global Q-function as

$$R_\omega^A(s, \mathbf{a}) = \max_{D(\mathbf{a}, \mathbf{a}') \leq K} \|Q(s, \mathbf{a}; \omega) - Q(s, \mathbf{a}'; \omega)\|_2^2, \quad (7)$$

where  $D(\mathbf{a}, \mathbf{a}') = \sum_i I(\mathbf{a}_i \neq \mathbf{a}'_i)$ . The regularizer Eq. (7) seeks to compute the worst subset of changed actions with cardinality less than  $K$ . For continuous action spaces, one could replace the metric  $D$  in Eq. (7) by a differentiable metric defined over the action space (e.g.,  $\|\cdot\|_2$ -norm), and then evaluate the regularizer with projected gradient ascent.

To evaluate the adversarial regularizer for the discrete action space, we propose to solve Eq. (7) in a greedy manner by finding the worst-case change in the action of a single agent at a time, until the action of  $K$  agents is changed. Specifically, at each training step, we search through all the agents/actions and then pick the actions that produce the top- $K$  changes in the Q-function,

resulting in a  $\mathcal{O}(|\mathcal{A}| * N * K)$  computation. Our complete algorithm can be found in Appendix G, and we find that in our numerical study, perturbing the action of a single agent ( $K = 1$ ) is sufficient for increased robustness.

Similar to the regularizer in Eq. (4), the regularizer in Eq. (7) provides the benefits of Lipschitz smoothness (with respect to the Hamming distance) and data augmentation with adversarial examples. If the behavior of a few agents changes (either maliciously or randomly), the behavior of policies trained by conventional methods may change drastically. On the other hand, policies trained by our method will continue to make reasonable decisions, resulting in more stable performance (see section 6).

#### 5.4 Extension to Mean-field MARL

MARL algorithms have been known to suffer from the curse of many agents [Wang et al., 2020], as the search space of policies and value functions grows exponentially w.r.t. the number of agents. A practical approach to tackle this challenge of scale is to adopt the mean-field approximation, which views each agent as realizations from a distribution of agents. This distributional perspective requires a distinct treatment of ERNIE applied to the mean-field setting.

Mean-field MARL avoids the curse of many agents by approximating the interaction between each agent and the global population of agents with that of an agent and the average agent from the population. In particular, we can approximate the action-value function of agent  $j$  as  $Q^j(\mathbf{s}, \mathbf{a}) = Q^j(s_j, d_s, a_j, \bar{a}_j)$ , where  $\mathbf{a}$  is the global joint action,  $\mathbf{s}$  is the global state,  $\bar{a}_j$  is the average action of agent  $j$ 's neighbors, and  $d_s$  is the empirical distribution of states over the population. Such an approximation has found widespread applications in practical MARL algorithms [Yang et al., 2018, Li et al., 2019b, 2021, Gu et al., 2021], and can be motivated in a principled fashion for agents of homogeneous nature [Wang et al., 2020, Li et al., 2021].

To learn robust and scalable policies, we extend ERNIE to the mean-field setting by applying adversarial regularization to the approximation terms  $d_s$  and  $\bar{a}^j$ . It is important to note that as the terms  $d_s$  and  $d'_s$  represent distributions over states, we bound the attack by the Wasserstein distance [Rüschendorf, 1985]. In what follows we only apply the regularizer to  $d_s$  for simplicity. This leads to a new regularizer defined over the mean field state

$$R_{\mathcal{W}}^Q(s; \theta) = \max_{\mathcal{W}(d'_s, d_s) \leq \epsilon} \sum_{a \in \mathcal{A}} \|Q_\theta(s, d'_s, a) - Q_\theta(s, d_s, a)\|_2^2,$$

where  $\mathcal{W}$  denotes the Wasserstein distance metric. Since the explicit Wasserstein constraint may be difficult to optimize in practice, we can instead enforce the constraint through regularization, as displayed in Appendix B.

## 6 Experiments

We conduct extensive experiments to demonstrate the effectiveness of our proposed framework. In each environment, we evaluate MARL algorithms trained by the ERNIE framework against

baseline robust MARL algorithms. To evaluate robustness, we train MARL policies in a non-perturbed environment and evaluate these policies in a perturbed environment. The reported results are gathered over five runs for each algorithm. Given the space limit, we put additional results in Appendix D.

**Traffic Light Control.** In this scenario, cooperative agents learn to minimize the total travel time of cars in a traffic network. We use QCOMBO [Zhang et al., 2019] and COMA [Foerster et al., 2018] (results in appendix D) as our baseline algorithms and conduct experiments using the Flow framework [Wu et al., 2017]. We train the MARL policies on a two-by-two grid (four agents). We then evaluate the policies on a variety of realistic environment changes, including different car speeds, traffic flows, network topologies, and observation noise. In each setting, we plot the reward for policies trained with ERNIE, the baseline algorithm (QCOMBO), and another baseline where the attack  $\delta$  is generated by a Gaussian random variable (Baseline-Gaussian, see Appendix H). Implementation details can be found in Appendix F.

Figure 1, 2a, and 2b show that the baseline algorithm is vulnerable to small changes in the training environment (higher reward is better). On the other hand, ERNIE achieves more stable reward on each environment change. This observation confirms that the ERNIE framework can improve robustness against observation noise and changing transition dynamics. The Gaussian baseline performs well on some environment changes, like when the observations are perturbed by Gaussian noise, but performs poorly on other environment changes, like when the car speed is changed. We hypothesize that while some environment changes may be covered by Gaussian perturbations, other environment changes are unlike Gaussian perturbations, resulting in a poor performance from this baseline.

**Robustness Against Malicious Actions.** We also evaluate the extension of ERNIE to robustness against changing agent behavior, which we refer to as ERNIE-A. To change agent behavior, we adversarially change the action of a randomly selected agent a small percentage of the time, i.e. 5% or 3% of the time. As can be seen in Figures 2c and 2d, the two baseline algorithms perform poorly when some agent’s behavior changes. In contrast, ERNIE-A is able to maintain a higher reward.

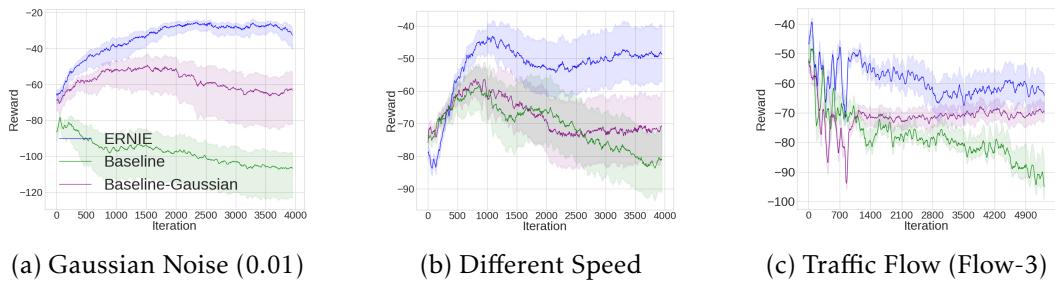


Figure 1: Evaluation curves on different environment changes for traffic light control.

**Particle Environments.** We evaluate ERNIE on the cooperative navigation, predator-prey, tag, and cooperative communication tasks. In each setting, we investigate the performance of the baseline

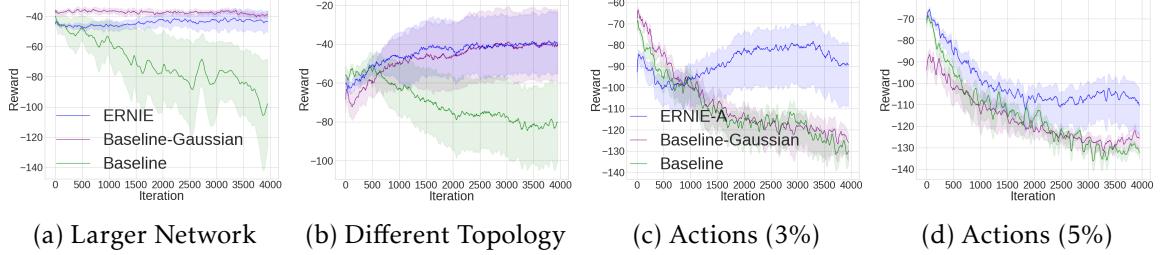


Figure 2: Performance on changed traffic network topologies and with malicious agents. In Figures (c) and (d) we perturb the actions according to the specified percentages.

algorithm (MADDPG), ERNIE, M3DDPG, and the baseline-gaussian in environments with varying levels of observation noise. We also compare ERNIE to the RMA3C algorithm proposed in [Han et al., 2022]. In Figure 3, we find ERNIE performs better or equivalently than MADDPG in all settings. Surprisingly, M3DDPG can provide some robustness against observation noise, even though it is designed to provide robustness against malicious actions.

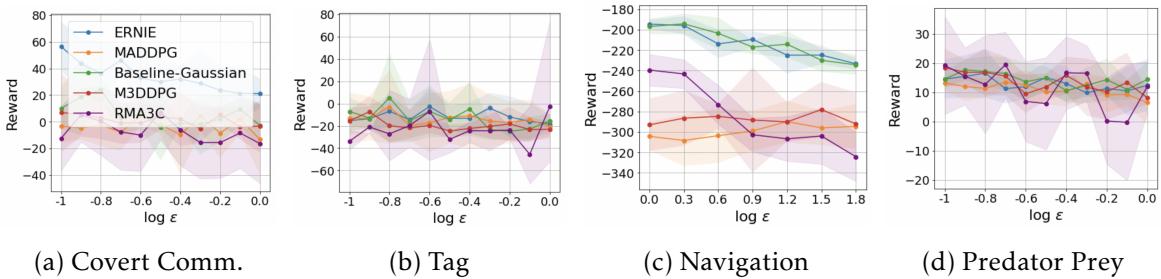


Figure 3: Training reward versus noise level ( $\epsilon$ ) in the evaluation environment for the particle games.

**Mean-field MARL.** We evaluate the performance of the mean-field ERNIE extension on the cooperative navigation task [Lowe et al., 2017] with different numbers of agents. We compare the performance of the baseline algorithm, ERNIE, and M3DDPG under various levels of observation noise. We use mean-field MADDPG as our baseline and follow the implementation of [Li et al., 2021]. As can be seen in Figure 4, ERNIE displays a higher reward and a slower decrease in performance across noise levels.

**Hyperparameter Study.** We investigate the sensitivity of ERNIE to the hyperparameters  $K$  (the number of attack steps) and  $\epsilon$  (the perturbation strength). We plot the performance of different hyperparameter settings in the traffic light control task with perturbed car speeds on three random seeds. From Figures 5c and 5d we can see that adversarial training ( $K > 0$ ) outperforms the baseline ( $K = 0$ ) for all  $K$ . Similarly, we can see that all values of  $\epsilon$  outperform the baseline ( $\epsilon = 0$ ). This indicates that ERNIE is robust to different hyperparameter settings of  $\epsilon$  and  $K$ .

**Sensitivity and Ablation Study.** The advantage of the ERNIE framework goes beyond improving the mean reward. To show this, we evaluate 10 different initializations of each algorithm in two traffic environments: one with different speeds and another environment with observation noise.

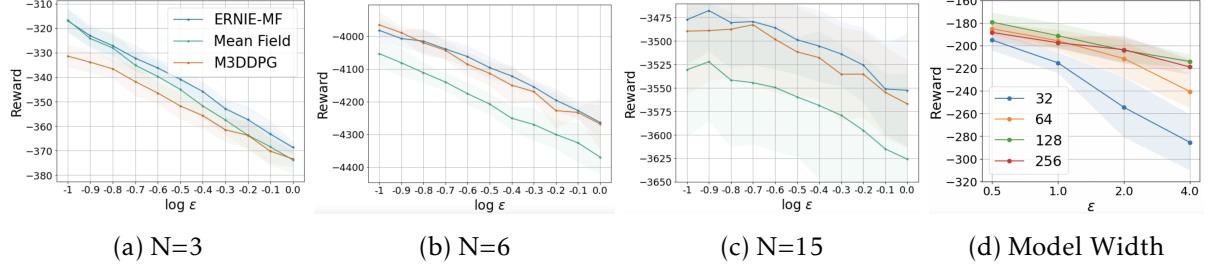


Figure 4: (a)-(c) Training reward versus noise level (mean  $\pm$  standard deviation over 5 runs) with a various number of agents (N) (d) Network width and robustness.

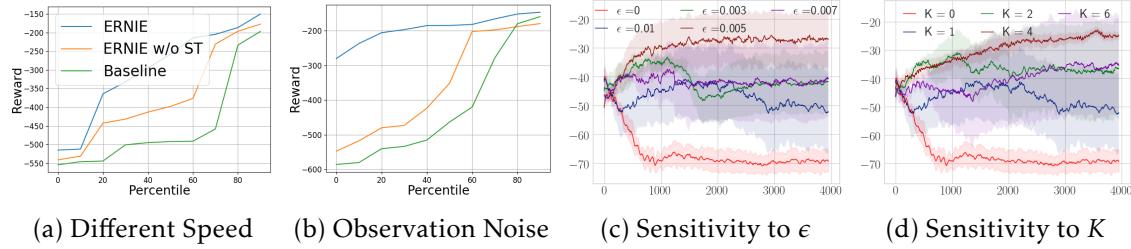


Figure 5: Sensitivity and ablation experiments.

We then sort the cumulative rewards of the learned policies and plot the percentiles in Figures 5a and 5b. Although the best-case performance is the same for all algorithms, ERNIE significantly improves the robustness to failure. As an ablation, we evaluate the effectiveness of ERNIE with and without the Stackelberg formulation of adversarial regularization (ST). ERNIE displays better performance in both settings, indicating that Stackelberg training can lead to a more stable training process. Ablation experiments in other environments can be found in Appendix D.4.

**Robustness and Network Width.** In Section 4, we show that in order to learn a robust policy with ERNIE, we should use a sufficiently wide neural network. Therefore, we evaluate the robust performance of ERNIE using policy networks with 32, 64, 128, and 256 hidden units. We carefully tune their regularization parameters such that all networks perform similarly in the lightly perturbed environment. As seen in Figure 4d, when the perturbed testing environment deviates more from the training environment, the performance of the narrower policy networks (32/64 hidden units) significantly drops, while the wider networks (128/256 hidden units) are more stable. We also observe that when the policy networks are sufficiently wide (128/256), their robustness is similar.

**Robotics Experiments.** Additional experiments in multi-agent drone control environments can be found in Appendix D.1, which further verify the enhanced robustness that ERNIE provides.

## 7 Discussion

ERNIE is motivated by smoothness, but real-world environments are not always smooth. In section 3, we hypothesize that most environments are at least partially smooth, implying that smoothness can serve as useful prior knowledge while providing robustness (our experiments validate this). To increase ERNIE’s flexibility, future work could adaptively select  $\lambda$  based on the current state to allow for state-dependent smoothness.

## 8 Acknowledgments

We would like to thank Haoming Jiang and Ethan Wang for their early contributions to this project.

## References

- Marco A Wiering. Multi-agent reinforcement learning for traffic light control. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1151–1158, 2000.
- Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.
- Qianli Shen, Yan Li, Haoming Jiang, Zhaoran Wang, and Tuo Zhao. Deep reinforcement learning with robust and smooth policy. In *International Conference on Machine Learning*, pages 8707–8718. PMLR, 2020.
- Lingxiao Wang, Zhuoran Yang, and Zhaoran Wang. Breaking the curse of many agents: Provable mean embedding q-iteration for mean-field reinforcement learning. In *International Conference on Machine Learning*, pages 10092–10103. PMLR, 2020.
- Erick Delage and Yinyu Ye. Distributionally robust optimization under moment uncertainty with application to data-driven problems. *Operations research*, 58(3):595–612, 2010.
- Joel Goh and Melvyn Sim. Distributionally robust optimization and its tractable approximations. *Operations research*, 58(4-part-1):902–917, 2010.
- Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.
- Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.
- Mohammed Amin Abdullah, Hang Ren, Haitham Bou Ammar, Vladimir Milenkovic, Rui Luo, Mingtian Zhang, and Jun Wang. Wasserstein robust reinforcement learning. *arXiv preprint arXiv:1907.13196*, 2019.

Xinlei Pan, Daniel Seita, Yang Gao, and John Canny. Risk averse robust adversarial reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8522–8528. IEEE, 2019.

Yue Wang and Shaofeng Zou. Policy gradient method for robust reinforcement learning. In *International Conference on Machine Learning*, pages 23484–23526. PMLR, 2022.

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. *Advances in Neural Information Processing Systems*, 33:21024–21037, 2020.

Aounon Kumar, Alexander Levine, and Soheil Feizi. Policy smoothing for provably robust reinforcement learning. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=mwdfa18NBrJ>.

Garud N Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):257–280, 2005.

Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

Kishan Panaganti, Zaiyan Xu, Dileep Kalathil, and Mohammad Ghavamzadeh. Robust reinforcement learning using offline data. *Advances in neural information processing systems*, 35:32211–32224, 2022.

Yan Li, Tuo Zhao, and Guanghui Lan. First-order policy optimization for robust markov decision process. *arXiv preprint arXiv:2209.10579*, 2022.

Jieyu Lin, Kristina Dzeparoska, Sai Qian Zhang, Alberto Leon-Garcia, and Nicolas Papernot. On the robustness of cooperative multi-agent reinforcement learning. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 62–68. IEEE, 2020.

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*, 2021. URL <https://openreview.net/forum?id=sCZbhBvqQaU>.

Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4213–4220, 2019a.

Yudan Wang, Yue Wang, Yi Zhou, Alvaro Velasquez, and Shaofeng Zou. Data-driven robust multi-agent reinforcement learning. In *2022 IEEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2022.

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.

Songyang Han, Sanbao Su, Sihong He, Shuo Han, Haizhao Yang, and Fei Miao. What is the solution for state adversarial multi-agent reinforcement learning? *arXiv preprint arXiv:2212.02705*, 2022.

Ziyuan Zhou, Guanjun Liu, and Mengchu Zhou. A robust mean-field actor-critic reinforcement learning against adversarial perturbations on agent states. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

Zhigen Zhao, Simiao Zuo, Tuo Zhao, and Ye Zhao. Adversarially regularized policy learning guided by trajectory optimization. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, pages 844–857. PMLR, 23–24 Jun 2022. URL <https://proceedings.mlr.press/v168/zhao22b.html>.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in neural information processing systems*, 32, 2019.

Simiao Zuo, Yue Yu, Chen Liang, Haoming Jiang, Siawpeng Er, Chao Zhang, Tuo Zhao, and Hongyuan Zha. Self-training with differentiable teacher. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 933–949, Seattle, United States, July 2022. Association for Computational Linguistics.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online, July 2020. Association for Computational Linguistics.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1063–1077, Online, June 2021. Association for Computational Linguistics.

Simiao Zuo, Chen Liang, Haoming Jiang, Xiaodong Liu, Pengcheng He, Jianfeng Gao, Weizhu Chen, and Tuo Zhao. Adversarial regularization as stackelberg game: An unrolled optimization approach. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6562–6577, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

- Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.
- Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.
- Andrzej Ruszczyński. Risk-averse dynamic programming for markov decision processes. *Mathematical programming*, 125(2):235–261, 2010.
- Haim Brezis and Haim Brézis. *Functional analysis, Sobolev spaces and partial differential equations*, volume 2. Springer, 2011.
- Chao Yu, Akash Velu, Eugene Vinitksy, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.
- Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.
- John Tsitsiklis and Benjamin Van Roy. Analysis of temporal-difference learning with function approximation. *Advances in neural information processing systems*, 9, 1996.
- Heinrich Von Stackelberg. *Market structure and equilibrium*. Springer Science & Business Media, 2010.
- Barak A Pearlmutter and Jeffrey Mark Siskind. Reverse-mode AD in a functional framework: Lambda the ultimate backpropagator. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 30(2):1–36, 2008.
- Yan Li, Lingxiao Wang, Jiachen Yang, Ethan Wang, Zhaoran Wang, Tuo Zhao, and Hongyuan Zha. Permutation invariant policy optimization for mean-field multi-agent reinforcement learning: A principled approach. *arXiv preprint arXiv:2105.08268*, 2021.
- Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580. PMLR, 2018.
- Minne Li, Zhiwei Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference*, pages 983–994, 2019b.
- Haotian Gu, Xin Guo, Xiaoli Wei, and Renyuan Xu. Mean-field multi-agent reinforcement learning: A decentralized network approach. *arXiv preprint arXiv:2108.02731*, 2021.
- Ludger Rüschendorf. The wasserstein distance and approximation theorems. *Probability Theory and Related Fields*, 70(1):117–129, 1985.

- Zhi Zhang, Jiachen Yang, and Hongyuan Zha. Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. *arXiv preprint arXiv:1909.10651*, 2019.
- Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.
- Cathy Wu, Aboudy Kreidieh, Kanaad Parvate, Eugene Vinitsky, and Alexandre M Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *arXiv preprint arXiv:1710.05465*, 10, 2017.
- Ingo Gühring, Gitta Kutyniok, and Philipp Petersen. Error bounds for approximations with deep relu neural networks in  $\mathbf{w}$   $s$ ,  $p$  norms. *Analysis and Applications*, 18(05):803–859, 2020.
- Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. A fast proximal point method for computing exact wasserstein distance. In *Uncertainty in Artificial Intelligence*, pages 433–453. PMLR, 2020.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Jacopo Panerati, Hehui Zheng, SiQi Zhou, James Xu, Amanda Prorok, and Angela P Schoellig. Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7512–7519. IEEE, 2021.
- Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

## A Proofs

*Proof of Theorem 3.1.* From the definition of Q-function, we have

$$\begin{aligned} |Q^\pi(s, a) - Q^\pi(s', a)| &\leq |r(s, a) - r(s', a)| + \gamma |\sum_{s_1 \in \mathcal{S}} \mathbb{P}(s_1|s, a) V^\pi(s_1) - \sum_{s_1 \in \mathcal{S}} \mathbb{P}(s_1|s', a) V^\pi(s_1)| \\ &\leq L_r \|s - s'\| + \frac{\gamma L_p}{1-\gamma} \|s - s'\|, \end{aligned}$$

where the last inequality also uses the fact that  $\|V^\pi\|_\infty \leq \frac{1}{1-\gamma}$ .

From the relation of  $Q^\pi$  and  $V^\pi$ , we have

$$\begin{aligned} &|V^\pi(s) - V^\pi(s')| \\ &= |\langle Q^\pi(s, \cdot), \pi(\cdot|s) \rangle - \langle Q^\pi(s', \cdot), \pi(\cdot|s') \rangle| \\ &= |\langle Q^\pi(s, \cdot), \pi(\cdot|s) - \pi(\cdot|s') \rangle + \langle Q^\pi(s, \cdot) - Q^\pi(s', \cdot), \pi(\cdot|s') \rangle| \\ &\leq |\langle Q^\pi(s, \cdot), \pi(\cdot|s) - \pi(\cdot|s') \rangle| + |\langle Q^\pi(s, \cdot) - Q^\pi(s', \cdot), \pi(\cdot|s') \rangle| \\ &\leq \frac{L_\pi}{1-\gamma} \|s - s'\| + L_Q \|s - s'\|. \end{aligned}$$

□

*Proof of Theorem 3.2.* Let  $Q^* \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|}$  denotes the optimal state-action value function. Let  $\pi^*$  denote any optimal policy. From the Bellman optimality condition, it is clear that

$$\sup(\pi(\cdot|s)) \subseteq \text{Argmax}_{a \in \mathcal{A}} Q^*(s, a). \quad (8)$$

From the performance difference lemma, we obtain that for any policy  $\pi$ , the optimality gap of  $\pi$  can be bounded by

$$\begin{aligned} 0 \leq V^{\pi^*}(s) - V^\pi(s) &= -\left( V^\pi(s) - V^{\pi^*}(s) \right) \\ &= -\mathbb{E}_{s' \sim d_s^\pi} [\langle Q^{\pi^*}(s', \cdot), \pi(\cdot|s') - \pi^*(\cdot|s) \rangle] \\ &= \mathbb{E}_{s' \sim d_s^\pi} [\langle Q^{\pi^*}(s', \cdot), \pi^*(\cdot|s') - \pi(\cdot|s') \rangle] \\ &\leq \sup_{s \in \mathcal{S}} \langle Q^{\pi^*}(s, \cdot), \pi^*(\cdot|s) - \pi(\cdot|s) \rangle \end{aligned}$$

where the last inequality uses (8), which implies that inner product is non-negative for every  $s \in \mathcal{S}$ .

Now consider the special policy

$$\pi_\eta(\cdot|s) = \text{Softmax}(\eta Q^*(s, \cdot)),$$

where operator  $\text{Softmax} : \mathbb{R}^{|\mathcal{A}|} \rightarrow \mathbb{R}^{|\mathcal{A}|}$  is defined as  $[\text{Softmax}(x)]_i = \exp(x_i) / \sum_j \exp(x_j)$ .

For any  $\epsilon > 0$ , let us define

$$\mathcal{A}_\epsilon = \left\{ a \in \mathcal{A} : Q^*(s, a) \leq \max_{a' \in \mathcal{A}} Q^*(s, a') - \epsilon \right\}.$$

Consequently, we have

$$\langle Q^{\pi^*}(s, \cdot), \pi^*(\cdot|s) - \pi_\eta(\cdot|s) \rangle \leq \frac{\epsilon}{1-\gamma} + \sum_{a \in \mathcal{A}_\epsilon} \pi_\eta(a|s).$$

It then suffices to set  $\eta$  properly to control the second term above. Specifically, we have

$$\sum_{a \in \mathcal{A}_\epsilon} \pi_\eta(a|s) \leq \frac{\sum_{a \in \mathcal{A}_\epsilon} \exp(-\eta\epsilon)}{1+|\mathcal{A}_\epsilon|\exp(-\eta\epsilon)} \leq |\mathcal{A}_\epsilon^c| \exp(-\eta\epsilon),$$

By setting  $\eta = \log|\mathcal{A}|/\epsilon$ , we immediately obtain that  $\sum_{a \in \mathcal{A}_\epsilon} \pi_\eta(a|s) \leq \epsilon$ , and hence

$$\langle Q^{\pi^*}(s, \cdot), \pi^*(\cdot|s) - \pi_\eta(\cdot|s) \rangle \leq \frac{\epsilon}{1-\gamma} + \sum_{a \in \mathcal{A}_\epsilon} \pi_\eta(a|s) \leq \frac{2\epsilon}{1-\gamma}, \quad \forall s \in \mathcal{S}.$$

Hence, we obtain that  $V^{\pi^*}(s) - V^\pi(s) \leq \frac{2\epsilon}{1-\gamma}$ ,  $\forall s \in \mathcal{S}$ .

It remains to show that  $\pi_\eta(\cdot|s)$  with the  $\eta = \log|\mathcal{A}|/\epsilon$  is indeed Lipschitz continuous with respect to state  $s$ . To this end, let us denote the Jacobian matrix of Softmax at point  $x$  as  $\mathcal{J}_x$ . Simple calculation yields that

$$\begin{aligned} [\mathcal{J}_x]_{i,i} &= \frac{\exp(x_i) \sum_{j \neq i} \exp(x_j)}{(\sum_j \exp(x_j))^2}, \\ [\mathcal{J}_x]_{i,j} &= -\frac{\exp(x_i) \exp(x_j)}{(\sum_j \exp(x_j))^2}. \end{aligned}$$

From the intermediate value theorem, we obtain that  $\|\text{Softmax}(x) - \text{Softmax}(y)\|_1 \leq \|\mathcal{J}_z\|_1 \|x - y\|_1$ , for  $z = \alpha x + (1 - \alpha)y$  and  $\alpha \in [0, 1]$ . Since it is clear that  $\|\mathcal{J}_z\|_1 \leq 1$ , we conclude that

$$\|\pi_\eta(\cdot|s) - \pi_\eta(\cdot|s')\|_1 \leq \eta \|Q^*(s, \cdot) - Q^*(s', \cdot)\|_1 \stackrel{(a)}{\leq} \eta L_Q |\mathcal{A}| \|s - s'\| \leq |\mathcal{A}| \log|\mathcal{A}| L_Q \|s - s'\| / \epsilon,$$

where inequality (a) applies Theorem 3.1.  $\square$

*Proof of Theorem 3.3.* We first recall that for any stationary policy  $\pi$ , the value function  $V^\pi$  at any state  $\bar{s}$  admits the following description,

$$\begin{aligned} V^\pi(\bar{s}) &= \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = \bar{s} \right] \\ &= \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi(s_t = s, a_t = a | s_0 = \bar{s}) r(s, a) \\ &= \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi(s_t = s | s_0 = \bar{s}) \pi(a|s) r(s, a). \end{aligned} \tag{9}$$

Similarly, given the definition of  $V^{\tilde{\pi}}$  for the non-stationary policy, we know that

$$V^{\tilde{\pi}}(\bar{s}) = \sum_{s \in \mathcal{S}, a \in \mathcal{A}} \sum_{t=0}^{\infty} \gamma^t \mathbb{P}^{\tilde{\pi}}(s_t = s | s_0 = \bar{s}) \tilde{\pi}(a|s) r(s, a). \tag{10}$$

By definition, we have the following observations,

$$\mathbb{P}^{\tilde{\pi}, t} := \mathbb{P}^{\tilde{\pi}}(s_t = \cdot | s_0 = \bar{s}) = \prod_{i=0}^{t-1} \mathbb{P}_i^{\tilde{\pi}} e(\bar{s}), \quad \mathbb{P}^{\pi, t} := \mathbb{P}^\pi(s_t = \cdot | s_0 = \bar{s}) = (\mathbb{P}^\pi)^t e(\bar{s}),$$

where  $\mathbb{P}_i^{\tilde{\pi}}(s', s) := \sum_{a \in \mathcal{A}} \tilde{\pi}_i(a|s) \mathbb{P}(s'|s, a)$ , and  $\mathbb{P}_i^\pi(s', s) := \sum_{a \in \mathcal{A}} \pi_i(a|s) \mathbb{P}(s'|s, a)$ , and  $e(s) \in \mathbb{R}^{|\mathcal{S}|}$  denotes the one-hot vector with non-zero entry corresponding to the state  $s$ . Hence

$$\begin{aligned}
& \left\| \mathbb{P}^{\tilde{\pi}, t} - \mathbb{P}^{\pi, t} \right\|_1 \\
&= \left\| \prod_{i=0}^{t-1} \mathbb{P}_i^{\tilde{\pi}} e(\bar{s}) - (\mathbb{P}^\pi)^t e(\bar{s}) \right\|_1 \\
&\leq \left\| \prod_{i=0}^{t-1} \mathbb{P}_i^{\tilde{\pi}} - (\mathbb{P}^\pi)^t \right\|_1 \\
&\leq \left\| \prod_{i=0}^{t-1} \mathbb{P}_i^{\tilde{\pi}} - \mathbb{P}^\pi \prod_{i=1}^{t-1} \mathbb{P}_i^{\tilde{\pi}} + \mathbb{P}^\pi \prod_{i=1}^{t-1} \mathbb{P}_i^{\tilde{\pi}} - (\mathbb{P}^\pi)^2 \prod_{i=2}^{t-1} \mathbb{P}_i^{\tilde{\pi}} + \dots + (\mathbb{P}^\pi)^{t-1} \mathbb{P}_{t-1}^{\tilde{\pi}} - (\mathbb{P}^\pi)^t \right\|_1 \\
&\leq \left\| \prod_{i=0}^{t-1} \mathbb{P}_i^{\tilde{\pi}} - \mathbb{P}^\pi \prod_{i=1}^{t-1} \mathbb{P}_i^{\tilde{\pi}} \right\|_1 + \left\| \mathbb{P}^\pi \prod_{i=1}^{t-1} \mathbb{P}_i^{\tilde{\pi}} - (\mathbb{P}^\pi)^2 \prod_{i=2}^{t-1} \mathbb{P}_i^{\tilde{\pi}} \right\|_1 + \dots + \left\| (\mathbb{P}^\pi)^{t-1} \mathbb{P}_{t-1}^{\tilde{\pi}} - (\mathbb{P}^\pi)^t \right\|_1 \quad (11)
\end{aligned}$$

To handle each term above, we make use of the following lemma.

**Lemma A.1.** For any  $P, Q \in \mathbb{R}^d$  that are left stochastic matrices, and any matrix  $\Delta$  of the same dimension, we have

$$\|P\Delta Q\|_1 \leq \|\Delta\|_1.$$

*Proof.* Note that  $\|\cdot\|_1$  is an induced norm and hence is sub-multiplicative. In addition, we have  $\|P\|_1 = \|Q\|_1 = 1$  since they are left stochastic matrices. We have

$$\|P\Delta Q\|_1 \leq \|P\Delta\|_1 \leq \|\Delta\|_1.$$

□

Now for the  $k$ -th term in inequality (11), it can be rewritten and bounded as

$$\left\| (\mathbb{P}^\pi)^{k-1} \left( \mathbb{P}_k^{\tilde{\pi}} - \mathbb{P}^\pi \right) \prod_{i=k+1}^{t-1} \mathbb{P}_i^{\tilde{\pi}} \right\|_1 \stackrel{(a)}{\leq} \left\| \mathbb{P}_k^{\tilde{\pi}} - \mathbb{P}^\pi \right\|_1 \stackrel{(b)}{\leq} L_\pi \epsilon.$$

where the inequality (a) follows from Lemma A.1; and (b) use the following fact

$$\begin{aligned}
\sum_{s' \in \mathcal{S}} |\mathbb{P}_k^{\tilde{\pi}}(s', s) - \mathbb{P}^\pi(s', s)| &= \sum_{s' \in \mathcal{S}} \left| \sum_{a \in \mathcal{A}} (\tilde{\pi}_k(a|s) - \pi(a|s)) \mathbb{P}(s'|s, a) \right| \\
&\leq \sum_{a \in \mathcal{A}} |\tilde{\pi}_k(a|s) - \pi(a|s)| \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, a) \\
&= \|\tilde{\pi}_k(\cdot|s) - \pi(\cdot|s)\|_1 \leq L_\pi \epsilon,
\end{aligned}$$

together with the definition of matrix  $\|\cdot\|_1$ -norm. Thus we obtain

$$\left\| \mathbb{P}^{\tilde{\pi}, t} - \mathbb{P}^{\pi, t} \right\|_1 \leq t L_\pi \epsilon. \quad (12)$$

Given (12), we can further obtain that

$$\begin{aligned} & \left| \sum_{t=0}^{\infty} \gamma^t \sum_{s \in \mathcal{S}} \mathbb{P}^{\pi}(s_t = s | s_0 = \bar{s}) \sum_{a \in \mathcal{A}} \pi(a|s) r(s, a) - \sum_{t=0}^{\infty} \gamma^t \sum_{s \in \mathcal{S}} \mathbb{P}^{\tilde{\pi}}(s_t = s | s_0 = \bar{s}) \sum_{a \in \mathcal{A}} \pi(a|s) r(s, a) \right| \\ & \leq \sum_{t=0}^{\infty} \gamma^t \left\| \mathbb{P}^{\pi, t} - \mathbb{P}^{\tilde{\pi}, t} \right\|_1 \\ & \leq \sum_{t=0}^{\infty} \gamma^t \cdot t L_{\pi} \epsilon \leq \frac{L_{\pi} \epsilon}{(1-\gamma)^2}. \end{aligned} \quad (13)$$

In addition, it is also clear that

$$\left| \sum_{t=0}^{\infty} \gamma^t \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mathbb{P}^{\tilde{\pi}}(s_t = s | s_0 = \bar{s}) (\pi(a|s) - \tilde{\pi}_t(a|s)) r(s, a) \right| \leq \frac{L_{\pi} \epsilon}{1-\gamma}. \quad (14)$$

Hence by combining (9), (10), (13) and (14), we conclude that

$$|V^{\pi}(\bar{s}) - V^{\tilde{\pi}}(\bar{s})| \leq \frac{2L_{\pi}\epsilon}{(1-\gamma)^2}.$$

From the relation between  $Q^{\pi}$  and  $V^{\pi}$ , and the above inequality, we have

$$|Q^{\pi}(s, a) - Q^{\tilde{\pi}}(s, a)| = \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, a) |V^{\pi}(s') - V^{\tilde{\pi}}(s')| \leq \frac{2L_{\pi}\epsilon}{(1-\gamma)^2}.$$

□

**Theorem A.1** (Function approximation with Lipschitz continuity). *Suppose that the target function  $f^*$  satisfies*

$$f^* \in W^{\alpha, \infty}(\Omega) \quad \text{and} \quad \|f^*\|_{W^{\alpha, \infty}(\Omega)} \leq 1$$

for some  $\alpha \geq 2$ . Given a pre-specified approximation error  $\epsilon \in (0, 1/\sqrt{d}]$ , there exists a neural network model  $\tilde{f} \in \mathcal{F}(L, p)$  with  $L = \tilde{O}(\log(1/\epsilon))$  and  $p = \tilde{O}(\epsilon^{-\frac{d}{\alpha-1}})$ , such that

$$\|\tilde{f} - f^*\|_{\infty} \leq \epsilon \quad \text{and} \quad \|\tilde{f}\|_{\text{Lip}} \leq 1 + \sqrt{d}\epsilon,$$

where  $\tilde{O}$  hides some negligible constants or log factors.

*Proof.* Theorem A.1 can be proved based on Gühring et al. [2020], where under the same condition, they show

$$\|\tilde{f} - f^*\|_{W^{1, \infty}(\Omega)} \leq \epsilon.$$

Since  $f^* \in W^{\alpha, \infty}$  and  $\|f^*\|_{W^{\alpha, \infty}(\Omega)} \leq 1$ , we have

$$\|\nabla f^*\|_2 \leq 1 \quad \text{and} \quad \|\nabla \tilde{f} - \nabla f^*\|_{\infty} \leq \epsilon,$$

Note that though  $\tilde{f}$  is using a ReLU activation,  $\nabla \tilde{f}$  is well-defined except a measure zero set. Eventually, we obtain

$$\|\tilde{f}\|_{\text{Lip}} \leq \sup_{\Omega} \|\nabla \tilde{f}\|_2 \leq \sup_{\Omega} \|\nabla f^* + \nabla \tilde{f} - \nabla f^*\|_2 \leq \sup_{\Omega} \|\nabla f^*\|_2 + \sqrt{d} \|\nabla \tilde{f} - \nabla f^*\|_{\infty} \leq 1 + \sqrt{d}\epsilon \leq 2.$$

□

## B ERNIE for Mean-Field MARL

As mentioned in 5.4, to learn robust policies we aim to use the regularizer

$$R_{\mathcal{W}}^Q(s; \theta) = \max_{\mathcal{W}(d'_s, d_s) \leq \epsilon} \sum_{a \in \mathcal{A}} \|Q_\theta(s, d'_s, a) - Q_\theta(s, d_s, a)\|_2^2.$$

However, this optimization problem is difficult to optimize due to the explicit Wasserstein distance constraint. To avoid this computational difficulty, we instead solve the regularized problem

$$R_{\mathcal{W}}^Q(s; \theta) = \max \sum_{a \in \mathcal{A}} \|Q_\theta(s, d'_s, a) - Q_\theta(s, d_s, a)\|_2^2 - \lambda_{\mathcal{W}} \mathcal{W}(d'_s, d_s).$$

The Wasserstein distance term can be computed using IPOT methods with little added computational cost, and we can therefore use this regularizer in a similar manner to the original ERNIE regularizer [Xie et al., 2020].

## C Traffic Light Control Implementation Details

In our experiments we train four agents in a two by two grid. The length of each road segment is 400 meters and cars enter through each in-flowing lane at a rate of 700 car/hour. The control frequency is 1 Hz, i.e. we need to input an action every second. The reward is based on the following attributes for each agent  $n$ :

- $q^n$ : The sum of queue length in all incoming lanes.
- $wt^n$ : Sum of vehicle waiting time in all incoming lanes.
- $dl^n$ : The sum of the delay of all vehicles in the incoming lanes.
- $em^n$ : The number of emergency stops by vehicles in all incoming lanes.
- $fl^n$ : A Boolean variable indicating whether or not the light phase changed.
- $vl^n$ : The number of vehicles that passed through the intersection.

We can then define the individual reward as

$$R^n = -0.5q^n - 0.5wt^n - 0.5dl^n - 0.25em^n - fl^n + vl^n.$$

All algorithms have the same training strategy. Each agent is trained for five episodes with 3000 SUMO time steps each. At the beginning of training the agent makes random decisions to populate the road network before training begins. Each algorithm is evaluated for 5000 time steps, where the first 1000 seconds are used to randomly populate the road. For adversarial regularization, we use the  $\ell_2$  norm to bound the attacks  $\delta$ .

### C.1 Evaluation Traffic Flows

The traffic flows used to evaluate the MARL policies in a different environment are shown in table 1. In each flow the total number of cars is similar to the number of cars in the training environment.

### C.2 Details on Changes to Network Topology

In addition to evaluating traffic light control MARL algorithms when the traffic pattern/speed changes, we also evaluate said MARL algorithms when the traffic network topology slightly changes. We consider two changes to the traffic topology: we slightly change the length of road segments and we evaluate the agents in a larger grid.

To test performance on a larger grid, we evaluate the trained agents on a four by four and six by six traffic light network. Because we only train four agents, we duplicate the trained agents in order to fill out the grid. In the four by four case, we will have four sets of the originally trained four agents, arranged to cover each of the four two by two grids. This setting is especially relevant to the real world deployment of MARL-controlled traffic lights as directly training on a large network may be computationally infeasible.

Table 1: Evaluation Traffic Flows

Flow Number	Traffic Flow
1	[1000, 1000, 80, 80, 800, 800, 550, 550]
2	[1000, 1000, 20, 20, 700, 300, 900, 900]
3	[700, 700, 70, 700, 1400, 600, 80, 80]
4	[1000, 1200, 200, 200, 300, 300, 900, 900]
5	[300, 300, 900, 900, 700, 900, 10, 10]

### C.3 Computing resources

Experiments were run on Intel Xeon 6154 CPUs and Tesla V100 GPUs.

### C.4 Training Details

Both the actor and critic functions are parametrized by a three-layer multi-layer perceptron with 256 nodes per hidden layer. We use the ADAM optimizer [Kingma and Ba, 2014] to update parameters and use a grid search to find  $\lambda_Q$  and  $\lambda_\pi$ .

## D Additional Results

In this section we include results that we could not fit in the main paper due to limited space. In particular, we show an evaluation of COMA’s robustness with the ERNIE framework and some

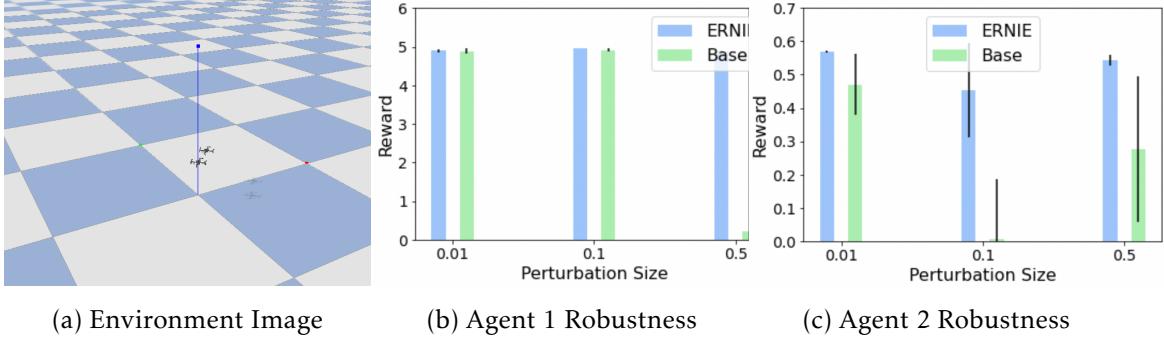


Figure 6: Evaluation of ERNIE in the multi-agent drone environment (see Figure 3a). The baseline algorithm we use is MAPPO. We then perturb the observation of each of the two agents with Gaussian noise to evaluate robustness (see Figure 3b-c). The task is follow the leader, where the agents have to navigate while remaining close to each other.

additional environment changes.

### D.1 Multi-Agent Drone Control

To evaluate the performance of ERNIE in multi-agent robotics environments, we use the multi-agent drone environment [Panerati et al., 2021]. We find that ERNIE can indeed provide enhanced robustness against input perturbations. The results can be found in Figure 6.

### D.2 ERNIE for COMA (Traffic Light Control)

We apply ERNIE to improve the robustness COMA for traffic light control. Figure 7 shows the performance of COMA with and without ERNIE on various environment changes. From Figure 7 we can see that the ERNIE and ERNIE w/o ST frameworks are able to outperform the baseline in all of the perturbed environments, indicating increased robustness. From table 2, we can again see that the ERNIE framework provides increased robustness to every environment change. Interestingly, ERNIE outperforms ERNIE w/o ST in the training environment and in the setting with small amounts of observation noise (see Figure 7), suggesting that the Stackelberg formulation allows for a better fit to the lightly perturbed data than conventional adversarial training does.

Table 2: Evaluation rewards and standard deviation for the traffic light control task under different environment perturbations. The baseline algorithm is COMA.

<i>Algorithm</i>	<i>Train</i>	<i>Obs. Noise (0.1)</i>	<i>Obs. Noise (1.0)</i>	<i>Speed (30 m/s)</i>
ERNIE	<b>-78.39(3.12)</b>	-86.37(7.1)	-102.45(3.45)	-91.73(6.84)
ERNIE w/o ST	-83.07(3.52)	<b>-84.66(6.15)</b>	<b>-101.37(3.05)</b>	<b>-91.69(8.82)</b>
Baseline	-93.97(7.34)	-108.05(8.68)	-113.24(5.82)	-108.45(3.98)

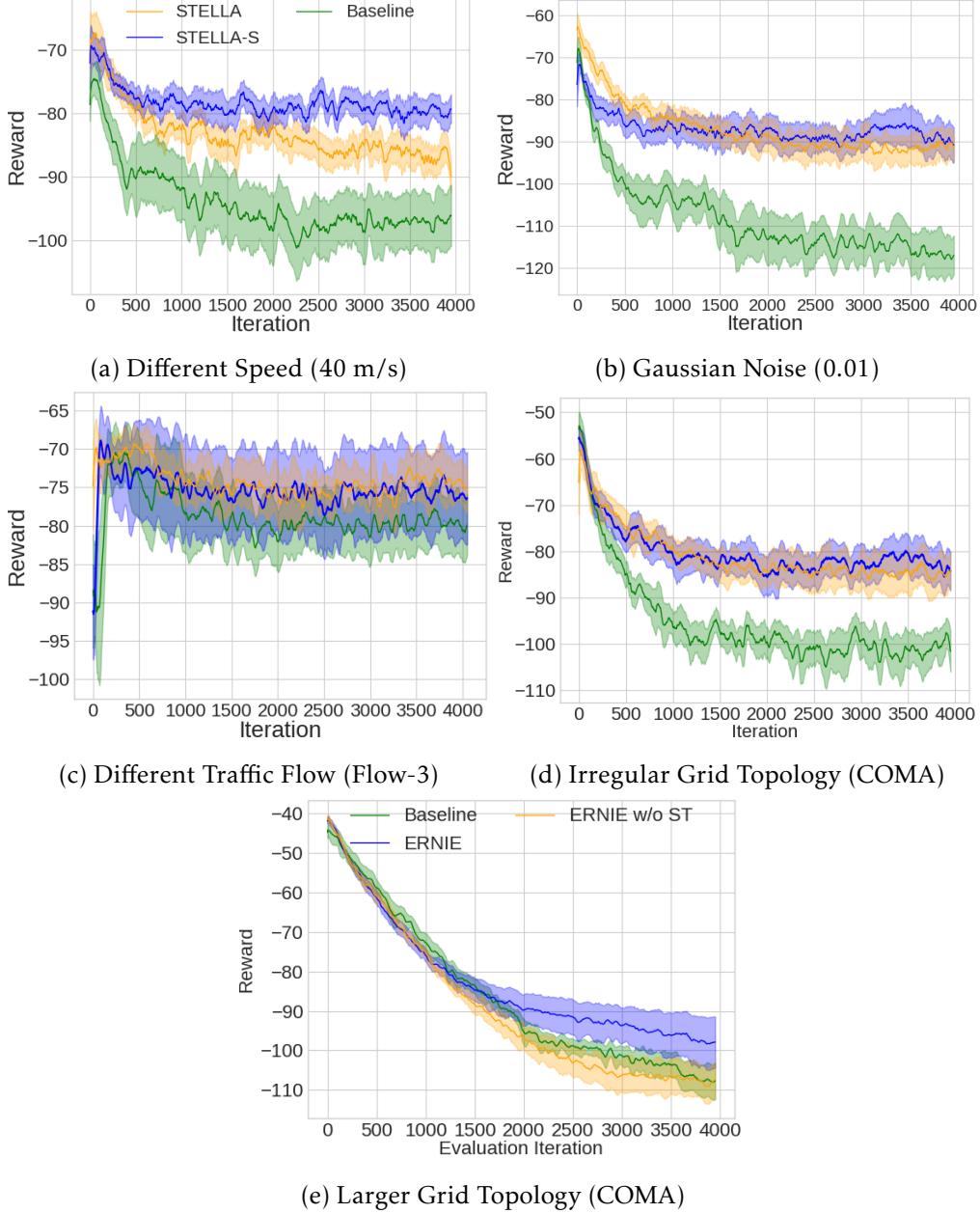


Figure 7: Evaluation curves from COMA on different environment changes for traffic light control.

### D.3 Additional Results on Changed Networks

In addition to evaluating the performance of ERNIE and the baseline algorithms on the  $4 \times 4$  network, we evaluate the performance of these algorithms on a  $6 \times 6$  network. The results shown in table 3 shows that ERNIE and ERNIE-S again outperform the baseline algorithm in the changed environment, indicating increased robustness.

We also evaluate the performance of ERNIE in another irregular traffic network from Atlanta. This grid can be seen in Figure 8, and the performance of ERNIE and the baselines can be seen in

Table 3: Evaluation rewards and standard deviations on larger networks.

<i>Algorithm</i>	$4 \times 4$	$6 \times 6$
Baseline (QCOMBO)	-401.64(22.25)	-320.66(40.80)
ERNIE w/o ST	-221.24(13.88)	-213.20(14.04)
ERNIE	<b>-217.21(8.36)</b>	<b>-152.60(3.91)</b>
Baseline (COMA)	-384.17	-330.55(5.70)
ERNIE	-394.14(1.29)	-337.25(3.86)
ERNIE w/o ST	<b>-369.40(6.04)</b>	<b>-319.16(3.95)</b>

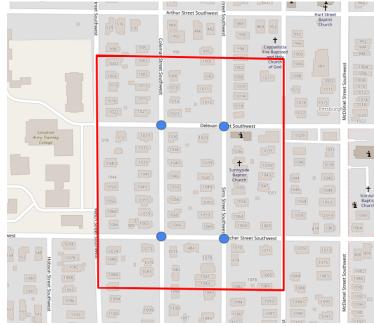


Figure 8: Irregular  $2 \times 2$  traffic network from Atlanta.

table 4. As with the other environment changes, we can see that the ERNIE framework exhibits increased robustness over the baseline algorithms.

Table 4: Evaluation rewards and standard deviation on irregular networks

<i>Algorithm</i>	<i>Atlanta</i>
Baseline (QCOMBO)	-435.69(27.09)
ERNIE w/o ST	-339.48(28.98)
ERNIE	<b>-285.84(28.44)</b>
Baseline (COMA)	-477.54(4.41)
ERNIE w/o ST	<b>-402.12(5.67)</b>
ERNIE	-432.87(5.43)

#### D.4 Additional Ablation Experiments

To further verify the effectiveness of the Stackelberg reformulation of adversarial regularization, we compare the performance of ERNIE with and without ST (Stackelberg Training) in the particle environments. The results are shown in Figure 9, where we can see that the Stackelberg formulation performs better or equivalently to normal adversarial regularization in all settings.

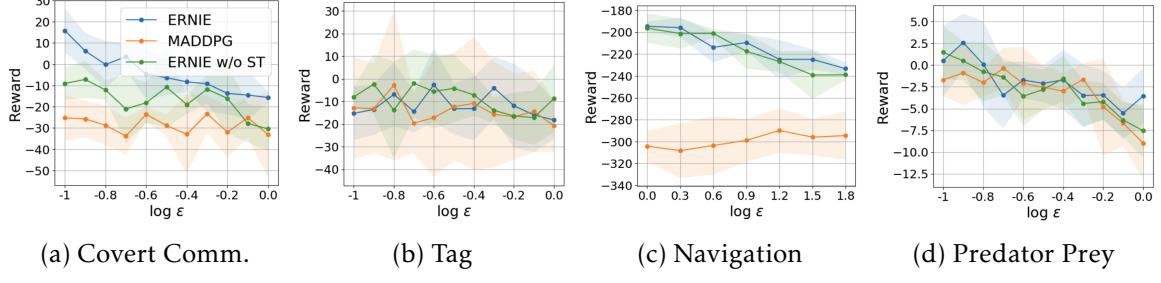


Figure 9: Ablation study comparing ERNIE with and without Stackelberg Training (ST).

## D.5 Time Comparison

In the cooperative navigation environment with 3 agents, we find that the baseline MADDPG takes 1.127 seconds for 50 episodes, ERNIE takes 1.829 seconds, and M3DDPG takes 3.250 seconds. Although ERNIE is more expensive than vanilla training, it is significantly more efficient than competitive baselines.

## E Baseline Algorithms

In this section we describe the baseline algorithms in detail.

### E.1 QCOMBO

QCOMBO [Zhang et al., 2019] is a Q-learning based MARL algorithm that couples independent and centralized learning with a novel regularization method. QCOMBO consists of three components, an *individual part*, a *global part*, and a *consistency regularization*. The individual part consists of Q-learning for each agent

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{n=1}^N \mathbb{E} \left[ \frac{1}{2} (y_t^n - Q^n(o_t^n, a_t^n; \theta^n))^2 \right],$$

where  $y_t^n = r_t^n + \gamma \max_{\hat{a}^n} Q^n(o_{t+1}^n, \hat{a}^n; \theta^n)$ ,  $\forall n \in [N]$ , and  $\theta = [\theta^1, \dots, \theta^n]$  denotes the concatenation of local parameters.

The global part consists of a global Q-network that learns a global Q function. We parameterize the global Q-function by  $\omega$ , and minimize the approximate Bellman residual

$$\mathcal{L}(\omega) = \mathbb{E} \left[ \frac{1}{2} (y_t - Q(s_t, \mathbf{a}_t; \omega))^2 \right], \quad (15)$$

where  $y_t = r_t^g + \gamma Q(s_{t+1}, \mathbf{a}_t'; \omega)$  and  $\mathbf{a}_t' = (a_1^t, \dots, a_N^t)$ ,  $a_n^t \in \text{argmax}_{\hat{a}^n \in A^n} Q^n(o_{t+1}^n, \hat{a}^n; \theta^n)$ . Finally a consistency regularization

$$\mathcal{L}_{\text{reg}}(\omega, \theta) = \mathbb{E} \left[ \frac{1}{2} (Q(s, \mathbf{a}; \omega) - \sum_{n=1}^N Q^n(o^n, a^n; \theta^n))^2 \right]$$

ensures that global and individual utility functions are similar, to encourage cooperation. The complete QCOMBO loss is then given by

$$\mathcal{L}_{\text{QC}}(\omega, \theta) = \mathcal{L}(\omega) + \mathcal{L}(\theta) + \lambda_Q \mathcal{L}_{\text{reg}}(\omega, \theta).$$

Here  $\lambda_Q$  is a hyperparameter that can be tuned. In execution decisions are made with the individual utility functions,  $\{Q^n\}_{n=1}^N$ . In practice we apply ERNIE to the individual Q-functions  $Q^n$ .

## E.2 MADDPG

MADDPG is a multi-agent version of Deep Deterministic Policy Gradient (DDPG). DDPG uses the actor-critic architecture where a state-action value function  $Q_\phi$  is used to update a deterministic policy  $\mu_\theta$ . The state-action value function is updated to minimize the squared bellman loss

$$\mathcal{L}(\phi) = \mathbb{E}_{s_t \sim \rho}[(Q_\phi(s_t, a_t) - y_t)^2]$$

where  $y_t = r_t + Q'(\mathbf{s}_{t+1}, \mu'_\theta(\mathbf{s}_{t+1}))$  and  $Q'_\phi(\cdot), \mu'_\theta(\cdot)$  are target networks. The policy function is updated with the policy gradient taking the form

$$\mathbb{E}_{s_t \sim \rho}[\nabla Q_\phi(s_t, a_t)|_{a_t=\mu_\theta(s_t)} \nabla \mu_\theta(s_t)].$$

The target networks are gradually updated throughout training to track the actor and critic networks.

MADDPG extends DDPG to the multi-agent setting with the paradigm of centralized training with decentralized execution. In particular, MADDPG employs a centralized state-action value function  $Q_\phi$  and independent actor functions  $\{\mu_{\theta_1}, \dots, \mu_{\theta_N}\}$ . Denoting  $\mathbf{a}_t$  as the joint action of the agents at time  $t$ , the state-action value function is updated to minimize the squared bellman loss

$$\mathcal{L}(\phi) = \mathbb{E}_{s_t \sim \rho}[(Q_\phi(s_t, \mathbf{a}_t) - y_t)^2]$$

where  $y_t = r_t + Q'(\mathbf{s}_{t+1}, \mu'_{\theta_1}(o_{1,t+1}), \dots, \mu'_{\theta_N}(o_{N,t+1}))$  and  $Q'_\phi(\cdot), \mu'_{\theta_1}(\cdot), \dots, \mu'_{\theta_N}(\cdot)$  are target networks. Each policy function  $\mu_{\theta_i}$  is updated with the policy gradient

$$\mathbb{E}_{s \sim \rho}[\nabla Q_\phi(s, \mathbf{a})|_{\mathbf{a}=\mu_{\theta_1}(o_1), \dots, \mu_{\theta_N}(o_N)} \nabla \mu_{\theta_i}(o_i)].$$

where  $\rho$  is the state visitation distribution. Note that the state-action value function is only used during training and that actions are only taken with the decentralized policy functions. In practice we apply ERNIE to the individual policies  $\mu_\theta$ .

## E.3 COMA

COMA is a policy gradient algorithm that directly seeks to minimize the negative cumulative reward  $\mathcal{L}_{\text{NCR}}$  by learning  $\{\pi_n\}_{n=1}^N$  parametrized by  $\theta = \{\theta_n\}_{n=1}^N$  with the actor-critic training paradigm. Specifically, COMA updates local policies (actors) with policy gradient

$$\nabla L_{\text{NCR}}(\theta) = \mathbb{E}_\pi \left[ \sum_{n=1}^N \nabla_\theta \log \pi^n(a^n | o^n) A^n(s, \mathbf{a}) \right], \quad (16)$$

where  $A^n(s, \mathbf{a})$  is the counterfactual baseline given by  $A^n(s, \mathbf{a}) = Q(s, \mathbf{a}) - \sum_{\tilde{\mathbf{a}}^n} \pi^n(\tilde{\mathbf{a}}^n | o^n) Q(s, (\mathbf{a}^{-n}, \tilde{\mathbf{a}}^n))$ . The critic parametrized by  $\theta^c$  is trained with  $\mathcal{L}(\theta^c) = \mathbb{E}_{\pi} \left[ \frac{1}{2} (y_t - Q_{\theta^c}(s_t, \mathbf{a}_t))^2 \right]$ , where  $y_t^n$  is the target value defined in TD( $\lambda$ ) [Sutton and Barto \[2018\]](#). In execution decisions are made with the individual policy functions  $\{\pi^n\}_{n=1}^N$ . In practice we apply ERNIE to the individual policies  $\pi^n$ .

## F Particle Environments Implementation Details

For the particle environments task, we follow the implementation of [maddpg-pytorch](#). For each task we parametrize the policy function with a three layer neural network, with 64 units hidden units. We then train for 25000 epochs (covert communication) 15000 epochs (cooperative navigation and predator prey), or 5000 epochs (tag). As we are considering the cooperative setting, we only apply ERNIE to the cooperating agents. The reward in the perturbed environments is that of the cooperative agents (note that we do not perturb the observations of the opposition agent). For adversarial regularization, we use the  $\ell_2$  norm to bound the attacks  $\delta$ . We use SGD to update parameters and use a grid search to find and  $\lambda_\pi$ .

### E.1 Mean-Field Implementation

For our mean-field implementation, we use the implementation of [Li et al. \[2021\]](#). For  $N = 3, 30$  agents we use a batch size of 32. For  $N = 6, 15$ , we use a batch size of 100. We train for 10000 episodes, and use a replay buffer of size 100. All other configurations should be the same as used in [Li et al. \[2021\]](#).

### E.2 M3DDPG Implementation

We implement our own version of M3DDPG in PyTorch [[Paszke et al., 2019](#)], as the original implementation uses Tensorflow [[Abadi et al., 2016](#)]. In each setting, we tune the attack steps size  $\epsilon \in [1e-5, 1e-2]$ .

## G ERNIE-A

We show our algorithm for solving (7). Note that  $\mathbf{a}' \cup \mathbf{a}_{i,j}$  refers to the joint action  $\mathbf{a}$  where the action of agent  $i$  is changed to  $j$ .

## H Gaussian Baseline

The baseline-Gaussian is similar to ERNIE. However, instead of generating  $\delta$  as

$$\delta = \underset{\|\delta\| \leq \epsilon}{\operatorname{argmax}} D(\pi_{\theta_k}(o_k + \delta), \pi_{\theta_k}(o_k)),$$

---

**Algorithm 1** Algorithm for solving (7)

---

**Data**  $s, \mathbf{a}, \omega, K$

**a**'  $\leftarrow \mathbf{a}$

**for**  $k \leftarrow 1$  **to**  $K$  **do**

**a**<sub>temp</sub>  $\leftarrow \mathbf{a}'$

**for**  $i \leftarrow 1$  **to**  $N$  **do**

**for**  $j \leftarrow 1$  **to**  $|A|$  **do**

**a**<sub>compare</sub>  $\leftarrow \mathbf{a}' \cup \mathbf{a}_{i,j}$

**if**  $\|Q(s, \mathbf{a}_{compare}; \omega) - Q(s, \mathbf{a}; \omega)\|_2^2 > \|Q(s, \mathbf{a}_{temp}; \omega) - Q(s, \mathbf{a}; \omega)\|_2^2$  **then**

**a**<sub>temp</sub>  $\leftarrow \mathbf{a}_{compare}$    **a**'  $\leftarrow \mathbf{a}_{temp}$

---

$\delta$  is sampled from the standard normal  $\mathcal{N}(0, I)$ . Similar to ERNIE, this baseline will ensure the policy does not change too much given Gaussian input perturbations. This baseline therefore performs well in several environments, especially those with Gaussian observation noise. However, robustness against Gaussian noise does not ensure robustness against all noise, and the Gaussian baseline may therefore fail in some perturbed environments.