A Proofs

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

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

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

From the relation of Q^π and V^π, we have

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

□

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

$$\sup(\pi(\cdot|s))\subseteq\mathrm{Argmax}_{a\in\mathcal{A}}Q^*(s,a). \tag{6}$$

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

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

where the last inquality uses (6), which implies that inner product is non-negative for every $s\in\mathcal{S}$.

Now consider the special policy

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

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

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

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

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

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

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

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

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 $\mathrm{Softmax}$ at point x as $\mathcal{J}_x$. Simple calculation yields that

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

From the intermediate value theorem, we obtain that $\|\mathrm{Softmax}(x) - \mathrm{Softmax}(y)\|_1 \leq \|\mathcal{J}_z\|_1 \|x-y\|_1$, 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

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

where inequality (a) applies Theorem 3.1. □

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

$$\begin{aligned} V^\pi(\overline{s}) &= \mathbb{E}\left[\textstyle\sum_{t=0}^\infty \gamma^t r(s_t,a_t)|s_0=\overline{s}\right] \\ &= \textstyle\sum_{s\in\mathcal{S},a\in\mathcal{A}}\sum_{t=0}^\infty \gamma^t \mathbb{P}^\pi(s_t=s,a_t=a|s_0=\overline{s})r(s,a) \\ &= \textstyle\sum_{s\in\mathcal{S},a\in\mathcal{A}}\sum_{t=0}^\infty \gamma^t \mathbb{P}^\pi(s_t=s|s_0=\overline{s})\pi(a|s)r(s,a). \end{aligned} \tag{7}$$

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

$$V^{\widetilde{\pi}}(\overline{s}) = \textstyle\sum_{s\in\mathcal{S},a\in\mathcal{A}}\sum_{t=0}^\infty \gamma^t \mathbb{P}^{\widetilde{\pi}}(s_t=s|s_0=\overline{s})\widetilde{\pi}(a|s)r(s,a). \tag{8}$$

By definition, we have the following observations,

$$\mathbb{P}^{\widetilde{\pi},t} := \mathbb{P}^{\widetilde{\pi}}(s_t=\cdot|s_0=\overline{s}) = \prod_{i=0}^{t-1}\mathbb{P}_i^{\widetilde{\pi}} e(\overline{s}),\ \ \mathbb{P}^{\pi,t} := \mathbb{P}^{\pi}(s_t=\cdot|s_0=\overline{s}) = \left(\mathbb{P}^\pi\right)^t e(\overline{s}),$$

where $\mathbb{P}_i^{\widetilde{\pi}}(s',s) := \sum_{a\in\mathcal{A}}\widetilde{\pi}_i(a|s)\mathbb{P}(s'|s,a)$, 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

$$\begin{aligned} &\left\|\mathbb{P}^{\widetilde{\pi},t} - \mathbb{P}^{\pi,t}\right\|_1 \\ =&\left\|\prod_{i=0}^{t-1}\mathbb{P}_i^{\widetilde{\pi}}e(\overline{s}) - \left(\mathbb{P}^\pi\right)^t e(\overline{s})\right\|_1 \\ \leq&\left\|\prod_{i=0}^{t-1}\mathbb{P}_i^{\widetilde{\pi}} - \left(\mathbb{P}^\pi\right)^t\right\|_1 \\ \leq&\left\|\prod_{i=0}^{t-1}\mathbb{P}_i^{\widetilde{\pi}} - \mathbb{P}^\pi\prod_{i=1}^{t-1}\mathbb{P}_i^{\widetilde{\pi}} + \mathbb{P}^\pi\prod_{i=1}^{t-1}\mathbb{P}_i^{\widetilde{\pi}} - \left(\mathbb{P}^\pi\right)^2\prod_{i=2}^{t-1}\mathbb{P}_i^{\widetilde{\pi}} + \cdots + \left(\mathbb{P}^\pi\right)^{t-1}\mathbb{P}_{t-1}^{\widetilde{\pi}} - \left(\mathbb{P}^\pi\right)^t\right\|_1 \\ \leq&\left\|\prod_{i=0}^{t-1}\mathbb{P}_i^{\widetilde{\pi}} - \mathbb{P}^\pi\prod_{i=1}^{t-1}\mathbb{P}_i^{\widetilde{\pi}}\right\|_1 + \left\|\mathbb{P}^\pi\prod_{i=1}^{t-1}\mathbb{P}_i^{\widetilde{\pi}} - \left(\mathbb{P}^\pi\right)^2\prod_{i=2}^{t-1}\mathbb{P}_i^{\widetilde{\pi}}\right\|_1 + \cdots + \left\|\left(\mathbb{P}^\pi\right)^{t-1}\mathbb{P}_{t-1}^{\widetilde{\pi}} - \left(\mathbb{P}^\pi\right)^t\right\|_1 \end{aligned} \tag{9}$$

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 Δ 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 (9), it can be rewritten and bounded as

$$\left\|(\mathbb{P}^{\pi})^{k-1}\left(\mathbb{P}^{\widetilde{\pi}}_{k}-\mathbb{P}^{\pi}\right)\prod_{i=k+1}^{t-1}\mathbb{P}^{\widetilde{\pi}}_{i}\right\|_{1}\overset{(a)}{\leq}\left\|\mathbb{P}^{\widetilde{\pi}}_{k}-\mathbb{P}^{\pi}\right\|_{1}\overset{(b)}{\leq}L_{\pi}\epsilon.$$

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

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

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

$$\left\|\mathbb{P}^{\widetilde{\pi},t}-\mathbb{P}^{\pi,t}\right\|_{1}\leq tL_{\pi}\epsilon. \tag{10}$$

Given (10), we can further obtain that

$$\begin{aligned}&\textstyle\left|\sum_{t=0}^{\infty}\gamma^{t}\sum_{s\in\mathcal{S}}\mathbb{P}^{\pi}(s_t=s|s_0=\overline{s})\sum_{a\in\mathcal{A}}\pi(a|s)r(s,a)-\right. &(11)\\ &\textstyle\qquad\left.\sum_{t=0}^{\infty}\gamma^{t}\sum_{s\in\mathcal{S}}\mathbb{P}^{\widetilde{\pi}}(s_t=s|s_0=\overline{s})\sum_{a\in\mathcal{A}}\pi(a|s)r(s,a)\right| \\ &\textstyle\leq\sum_{t=0}^{\infty}\gamma^{t}\left\|\mathbb{P}^{\pi,t}-\mathbb{P}^{\widetilde{\pi},t}\right\|_{1} \\ &\textstyle\leq\sum_{t=0}^{\infty}\gamma^{t}\cdot tL_{\pi}\epsilon\leq\frac{L_{\pi}\epsilon}{(1-\gamma)^{2}}. &(12)\end{aligned}$$

In addition, it is also clear that

$$\textstyle\left|\sum_{t=0}^{\infty}\gamma^{t}\sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}}\mathbb{P}^{\widetilde{\pi}}(s_t=s|s_0=\overline{s})(\pi(a|s)-\widetilde{\pi}_t(a|s))r(s,a)\right|\leq\frac{L_{\pi}\epsilon}{1-\gamma}. \tag{13}$$

Hence by combining (7), (8), (12) and (13), we conclude that

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

From the relation between Q^{π} and V^{π}, and the above inequality, we have

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

□

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

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

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

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

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

Proof. Theorem A.1 can be proved based on [48], where under the same condition, they show

$$\left\|\widetilde{f}-f\right\|_{W^{1,\infty}(\Omega)}\leq\epsilon.$$

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

$$\|\nabla f^{*}\|_{2}\leq 1\quad\text{and}\quad\left\|\nabla\widetilde{f}-\nabla f^{*}\right\|_{\infty}\leq\epsilon,$$

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

$$\begin{aligned}\|\widetilde{f}\|_{\mathrm{Lip}}\leq\sup_{\Omega}\left\|\nabla\widetilde{f}\right\|_{2}&\leq\sup_{\Omega}\left\|\nabla f^{*}+\nabla\widetilde{f}-\nabla f^{*}\right\|_{2}\leq\sup_{\Omega}\|\nabla f^{*}\|_{2}+\sqrt{d}\left\|\nabla\widetilde{f}-\nabla f^{*}\right\|_{\infty}\\ &\leq 1+\sqrt{d}\epsilon\leq 2.\end{aligned}$$

□

B 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 [49].

Definition B.1. Given $\alpha \geq 0$ and domain $\Omega \subset \mathbb{R}^d$, we define the Sobolev space $W^{\alpha,\infty}(\Omega)$ as

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

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

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

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

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

$$f(x) = W_L \cdot \sigma(\cdots \sigma(W_1 s + b_1) \cdots) + b_L, \tag{14}$$

where $W_1, \ldots, W_L$ and $b_1, \ldots, b_L$ 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:

$$\begin{aligned}\mathcal{F}(L, p) =& \big\{ f \mid f(x) \text{ in the form (14) with } L\text{-layers} \\ & \text{and width bounded by } p \big\}. \end{aligned} \tag{15}$$

We next present the function approximation results.

Theorem B.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 ϵ, there exists a neural network $\widetilde{f} \in \mathcal{F}(L, p)$ with $L = \widetilde{O}(\log(\epsilon^{-1}))$ and $p = \widetilde{O}(\epsilon^{-\frac{d}{\alpha-1}})$, such that

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

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

For reinforcement learning, f^* in Theorem B.1 can be viewed as either the near-optimal smooth policy π^* 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 $\widetilde{f}$ such that $\|\widetilde{f}\|_{\mathrm{Lip}}$ is below 2, given a sufficient width $p = \widetilde{O}\left(d^{\frac{d\alpha}{2(\alpha-1)^2}}\right)$. This result indicates that when training policies with the ERNIE algorithm, we should use wide neural networks.

C ERNIE for Mean-Field MARL

As mentioned in 4.4, to learn robust policies we aim to use the regularizer

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

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

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

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 [50].

D 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 ℓ_2 norm to bound the attacks δ.

D.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.

D.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.

D.3 Computing resources

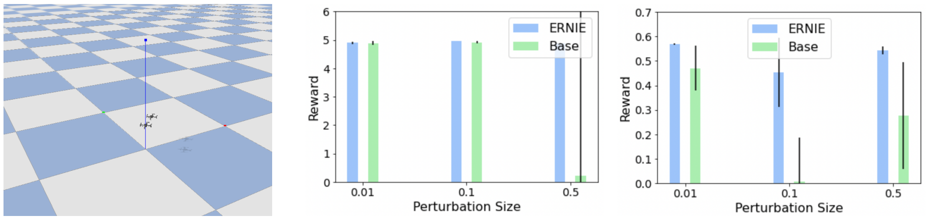
Experiments were run on Intel Xeon 6154 CPUs and Tesla V100 GPUs.

D.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 [51] to update parameters and use a grid search to find λ_Q and λ_π.

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]



(a) Environment Image (b) Agent 1 Robustness (c) Agent 2 Robustness

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.

E 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 additional environment changes.

E.1 Multi-Agent Drone Control

To evaluate the performance of ERNIE in multi-agent robotics environments, we use the multi-agent drone environment [52]. We find that ERNIE can indeed provide enhanced robustness against input perturbations. The results can be found in Figure 6.

E.2 ERNIE for COMA (Traffic Light Control)

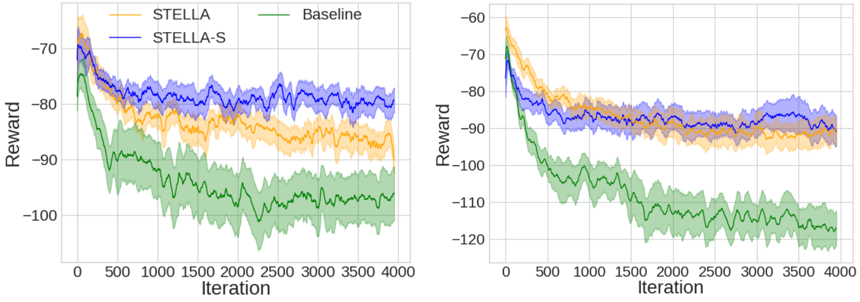
We apply ERNIE to improve the robustness COMA for traffic light control. Figure 7 shows the performance of COMA with and without ERNIE on various environment changes. From Figure 7 we can see that the ERNIE and ERNIE w/o ST frameworks are able to outperform the baseline in all of the perturbed environments, 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/0 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.

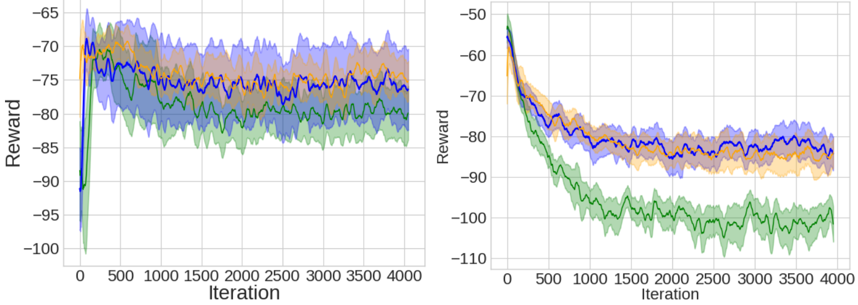
Algorithm	*Train*	*Obs. Noise (0.1)*	*Obs. Noise (1.0)*	*Speed (30 m/s)*
ERNIE	$-\mathbf{78.39}(\mathbf{3.12})$	$-86.37(7.1)$	$-102.45(3.45)$	$-91.73(6.84)$
ERNIE w/o ST	$-83.07(3.52)$	$-\mathbf{84.66}(\mathbf{6.15})$	$-\mathbf{101.37}(\mathbf{3.05})$	$-\mathbf{91.69}(\mathbf{8.82})$
Baseline	$-93.97(7.34)$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

E.3 Additional Results on Changed Networks

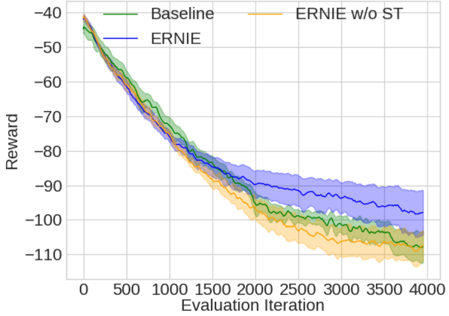
In addition to evaluating the performance of ERNIE and the baseline algorithms on the 4×4 network, we evaluate the performance of these algorithms on a 6×6 network. The results shown in table 3



(a) Different Speed (40 m/s) (b) Gaussian Noise (0.01)



(c) Different Traffic Flow (Flow-3) (d) Irregular Grid Topology (COMA)



(e) Larger Grid Topology (COMA)

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

shows that ERNIE and ERNIE-S again outperform the baseline algorithm in the changed environment, indicating increased robustness.

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

Algorithm	*4* × *4*	*6* × *6*
Baseline (QCOMBO)	−401.64(22.25)	−320.66(40.80)
ERNIE w/o ST	−221.24(13.88)	−213.20(14.04)
ERNIE	**−217.21(8.36)**	**−152.60(3.91)**
Baseline (COMA)	−384.17	−330.55(5.70)
ERNIE	−394.14(1.29)	−337.25(3.86)
ERNIE w/o ST	**−369.40(6.04)**	**−319.16(3.95)**

We also evaluate the performance of ERNIE in another irregular traffic network from Atlanta. This grid can be see in Figure 8, and the performance of ERNIE and the baselines can be seen in table 4. As with the other environment changes, we can see that the ERNIE framework exhibits increased robustness over the baseline algorithms.

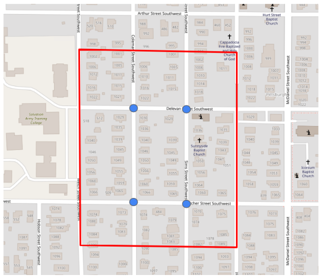


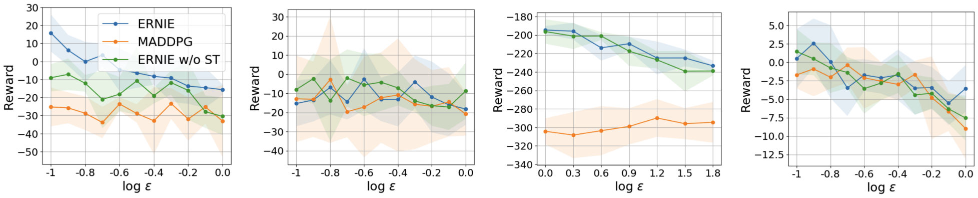
Figure 8: Irregular 2×2 traffic network from Atlanta.

Table 4: Evaluation rewards and standard deviation on irregular networks

Algorithm	*Atlanta*
Baseline (QCOMBO)	$-435.69(27.09)$
ERNIE w/o ST	$-339.48(28.98)$
ERNIE	$-\mathbf{285.84}(\mathbf{28.44})$
Baseline (COMA)	$-477.54(4.41)$
ERNIE w/o ST	$-\mathbf{402.12}(\mathbf{5.67})$
ERNIE	$-432.87(5.43)$

E.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 (Stackleberg 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.



(a) Covert Comm. (b) Tag (c) Navigation (d) Predator Prey

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

E.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.

F Baseline Algorithms

In this section we describe the baseline algorithms in detail.

F.1 QCOMBO

QCOMBO [45] 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 $y_t^n = r_t^n + \gamma \max_{\widehat{a}^n} Q^n(o_{t+1}^n, \widehat{a}^n; \theta^n), \;\; \forall n \in [N]$, and $\theta = [\theta^1, \ldots, \theta^n]$ 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 ω, and minimize the approximate Bellman residual

$$\mathcal{L}(\omega) = \mathbb{E}\big[\frac{1}{2}(y_t - Q(s_t, \mathbf{a}_t; \omega))^2\big], \tag{16}$$

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

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

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 λ_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.

F.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_ϕ is used to update a deterministic policy μ_θ. 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 $y_t = r_t + Q'(s_{t+1}, \mu'_\theta(s_{t+1}))$ and $Q'_\phi(\cdot), \mu'_\theta(\cdot)$ are target networks. The policy function is updated with the policy gradient taking the form

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

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_ϕ and independent actor functions $\{\mu_{\theta_1}, ..., \mu_{\theta_N}\}$. 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 $y_t = r_t + Q'(s_{t+1}, \mu'_{\theta_1}(o_{1,t+1}), ..., \mu'_{\theta_N}(o_{N,t+1}))$ and $Q'_\phi(\cdot), \mu'_{\theta_1}(\cdot), ..., \mu'_{\theta_N}(\cdot)$ are target networks. Each policy function μ_{θ_i} is updated with the policy gradient

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

where ρ 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 μ_θ.

F.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

$$\nabla L_{\text{NCR}}(\theta) = \mathbb{E}_\pi\big[\sum_{n=1}^{N} \nabla_\theta \log \pi^n(a^n|o^n) A^n(s, \mathbf{a})\big], \tag{17}$$

where $A^n(s, \mathbf{a})$ is the counterfactual baseline given by $A^n(s, \mathbf{a}) = Q(s, \mathbf{a}) - \sum_{\widetilde{a}^n} \pi^n(\widetilde{a}^n|o^n) Q(s, (\mathbf{a}^{-n}, \widetilde{a}^n))$. The critic parametrized by θ^c is with trained with $\mathcal{L}(\theta^c) = \mathbb{E}_\pi\big[\frac{1}{2}(y_t - Q_{\theta^c}(s_t, \mathbf{a}_t))^2\big]$, where y_t^n is the target value defined in TD(λ) [53]. In execution decisions are made with the individual policy functions $\{\pi^n\}_{n=1}^N$. In practice we apply ERNIE to the individual policies π^n.

G 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 ℓ_2 norm to bound the attacks δ. We use SGD to update parameters and use a grid search to find and λ_π.

G.1 Mean-Field Implementation

For our mean-field implementation, we use the implementation of Li et al. [40]. 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. [40].

G.2 M3DDPG Implementation

We implement our own version of M3DDPG in PyTorch [54], as the original implementation uses Tensorflow [55]. In each setting, we tune the attack steps size $\epsilon \in [1e-5, 1e-2]$.

H ERNIE-A

We show our algorithm for solving (5). 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.

Algorithm 1 Algorithm for solving (7)

```
Data $s, \mathbf{a}, \omega, K$
$\mathbf{a}' \leftarrow \mathbf{a}$
for $k \leftarrow 1$ to $K$ do
  $\mathbf{a}_{temp} \leftarrow \mathbf{a}'$
    for $i \leftarrow 1$ to $N$ do
        for $j \leftarrow 1$ to $|A|$ do
        $\mathbf{a}_{compare} \leftarrow \mathbf{a}' \cup \mathbf{a}_{i,j}$
            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
          $\mathbf{a}_{temp} \leftarrow \mathbf{a}_{compare}$
          $\mathbf{a}' \leftarrow \mathbf{a}_{temp}$
```

I Gaussian Baseline

The baseline-Gaussian is similar to ERNIE. However, instead of generating δ as

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

δ is sampled from the standard normal $\mathcal{N}(0, I)$. Similar to ERNIE, this baseline will ensure the policy does not change to 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.