chitecture (Raileanu et al. 2018) or auxiliary policy features (Grover et al. 2018; Hong et al. 2018). MBOM (Yu et al. 2021b) uses the (provided) environment model to simulate multiple best response policies as opponents and then mixes them according to the similarity with the real behaviors of opponents. However, MBOM were examined only in two-player games. In contrast with this set of methods that introduce prediction models to (over)fit an opponent's trajectories, in PERLA agents communicate their policy parameters (and local observations) to each other during training. With this and using our novel *policy embedding procedure* allows the agents to immediately form best responses to the joint policies of other agents.

3 PERLA framework

We formulate the MARL problem as a Markov game (MG) (Deng et al. 2021), which is represented by a tuple $\mathfrak{G} = \langle \mathcal{N}, \mathcal{S}, (\mathcal{A}_i)_{i\in\mathcal{N}}, P, R_i, \gamma\rangle$ where $\mathcal{S}$ is the finite set of states, $\mathcal{A}_i$ is an action set for agent $i \in \mathcal{N}$, and $R_i : \mathcal{S}\times\boldsymbol{\mathcal{A}} \to \mathcal{P}(D)$ is the reward function that agent i seeks to maximise (where D is a compact subset of $\mathbb{R}$), and $P : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \to [0, 1]$ is the probability function describing the system dynamics where $\boldsymbol{\mathcal{A}} := \times_{i=1}^{N}\mathcal{A}_i$. We consider a partially observable setting, in which given the system state $s^t \in \mathcal{S}$, each agent $i \in \mathcal{N}$ makes local observations $\tau_i^t = O(s^t, i)$ where $O : \mathcal{S} \times \mathcal{N} \to \mathcal{Z}_i$ is the observation function and $\mathcal{Z}_i$ is the set of local observations for agent i. To decide its actions, each agent $i \in \mathcal{N}$ samples its actions from a *Markov policy* $\pi_{i,\boldsymbol{\theta}_i} : \mathcal{Z}_i \times \mathcal{A}_i \to [0, 1]$, which is parameterised by the vector $\boldsymbol{\theta}_i \in \mathbb{R}^d$. Throughout the paper, $\pi_{i,\boldsymbol{\theta}_i}$ is abbreviated as π_i. At each time $t \in 0, 1, \ldots$, the system is in state $s^t \in \mathcal{S}$ and each agent $i \in \mathcal{N}$ takes an action $a_i^t \in \mathcal{A}_i$, which together with the actions of other agents $\boldsymbol{a}_{-i}^t := (a_1^t, \ldots, a_{i-1}^t, a_{i+1}^t, \ldots, a_N^t)$, produces an immediate reward $r_i \sim R(s^t, \boldsymbol{a}_i^t)$ for agent $i \in \mathcal{N}$. The system then transitions to a next state $s^{t+1} \in \mathcal{S}$ with probability $P(s^{t+1}|s^t, \boldsymbol{a}^t)$ where $\boldsymbol{a}^t = (a_1^t, \ldots, a_N^t) \in \boldsymbol{\mathcal{A}}$ is the *joint action* which is sampled from the *joint policy* $\boldsymbol{\pi} := \prod_{i=1}^N \pi_i$. The goal of each agent i is to maximise its expected returns measured by its VF $v_i(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s^t, \boldsymbol{a}^t)|s_0 = s\right]$ and the action-value function for each agent $i \in \mathcal{N}$ is given by $Q_i(s, \boldsymbol{a}) = \mathbb{E}[\sum_{t=0}^{\infty} R_i(s^t, \boldsymbol{a}^t)|\boldsymbol{a}^0 = \boldsymbol{a}]$, where $-i$ denotes the tuple of agents excluding agent i. Likewise, we denote $\prod_{j=1,j\neq i}^N \pi_j$ as $\boldsymbol{\pi}_{-i}$. In the fully cooperative case all agents share the same goal: $R_1 = \ldots R_N := R$.

In the CL paradigm, given a state $s \in \mathcal{S}$ and the joint action $\boldsymbol{a} \in \boldsymbol{\mathcal{A}}$, each agent $i \in \mathcal{N}$ computes its action-value function $Q_i(s, \boldsymbol{a})$. The action-value function provides an estimate of the agent's expected return using its policy given the behaviour of all other agents $\mathcal{N}/\{i\}$ for a given action $a_i \in \mathcal{A}_i$. Therefore, $Q_i(s, \boldsymbol{a})$ seeks to provide an estimate of the agent's own action, accounting for the actions of others. MARL agents use stochastic policies to explore — each joint action $\boldsymbol{a} \sim \boldsymbol{\pi}$ may contain exploratory actions that included in agent policy updates.

The core component of the PERLA framework is a Critic Policy Embedding procedure. As the name suggests, the procedure embeds the agents' policies within the critic, enabling us to construct a critic whose input takes only the agent's own action and the state while factoring in the joint behaviour of other agents. This is done by jointly marginalising the action-value function $\tilde{Q}$, as defined below:

$$\tilde{Q}_i(s, a_i) := \mathbb{E}_{\boldsymbol{\pi}_{-i}}\left[Q_i(s, \boldsymbol{a})\right]; \quad \boldsymbol{a} \equiv (a_i, \boldsymbol{a}_{-i}) \in \boldsymbol{\mathcal{A}}, \quad (1)$$

where $s \in \mathcal{S}, a_i \sim \pi_i(\cdot|\tau_i), \boldsymbol{a}_{-i} \sim \boldsymbol{\pi}_{-i}(\cdot|\tau_{-i})$. This object requires some explanation: as with Q_i, the function $\tilde{Q}_i$ seeks to estimate the expected return following agent i taking action a_i. However, unlike Q_i, $\tilde{Q}_i$ builds in the distribution of the actions played by other agents *under their current policy*. This gives less weight to low probability actions and conversely for high probability actions. A key aspect is that $\hat{Q}$ depends only on the agent's own action (and state) which acquires the scalability benefit of IL while factoring in the behaviour of other agents. As in practice, it may be impossible to analytically calculate 1, to approximate $\tilde{Q}_i(s, a_i)$, for any $s \in \mathcal{S}$ and any $a_i \in \mathcal{A}_i$ we construct the object $\hat{Q}$:

$$\hat{Q}_i(s, a_i) = \frac{1}{k}\sum_{j=1}^{k} Q_i(s, a_i, \boldsymbol{a}_i^{(j)}); \quad \boldsymbol{a}_{-i}^{(j)} \sim \pi(\boldsymbol{a}_{-i}|\tau_{-i}). \quad (2)$$

We now describe the implementation details of Actor-Critic algorithm using the critic policy embedding procedure, along with pseudocode in Algorithm 1:

1. Augment the critic Q to accommodate joint action, i.e., $Q(s^t, a_i^t, \boldsymbol{a}_{-i}^{t(j)})$.
2. Each agent i communicates its policy π_i and its local observation τ_i^t to other agents.
3. Upon arrival to state s^t each agent i samples other agents' actions given their local observations, k times using other agents' policies $a_j \sim \pi_j(a_j|\tau_j^t)$. This produces joint samples of the actions $\{\boldsymbol{a}_{-i}^{t(j)}\}_{j=1}^k$ for each time step t.
4. Each agent uses this to approximate the function $\tilde{Q}_i(s^t, a_i^t) = \frac{1}{k}\sum_{j=1}^k Q(s^t, a_i^t, \boldsymbol{a}_{-i}^{t(j)}) \approxeq \hat{Q}_i(s_t, a_i^t)$, embedding the behaviour of other agents.
5. Agent i uses $\tilde{Q}_i$ as the critic to update its policy using the chosen policy gradient algorithm.
6. Return to **2** and repeat until termination.

In our experiments, we use a value-function style critic, where $Q(s^t, a_i^t, \boldsymbol{a}_{-i}^{t(j)}) = r_t + \gamma V(s^t, \boldsymbol{a}_{-i}^{t(j)})$ and $V(s^t, \boldsymbol{a}_{-i}^{t(j)})$ is approximated via a deep neural network. In this case, it is enough to marginalise the next step values function. To perform policy updates, we use PPO updates (Schulman et al. 2017). We apply PERLA to two cases, one which involves a base learner whose critic input captures global information (MAPPO (Yu et al. 2021a)) and another base learner whose critic input is only local observations (IPPO) (in both cases, PERLA makes use of shared global state information during training). This gives rise to **PERLA MAPPO** and **PERLA IPPO** algorithms respectively.

4 Theoretical Analysis

In this section, we perform a detailed theoretical analysis of PERLA. Here, we derive theoretical results that establish