CONTEXT AND HISTORY AWARE OTHER-SHAPING

Anonymous authors
Paper under double-blind review

ABSTRACT

Cooperation failures, in which self-interested agents converge to collectively worst-case outcomes, are a common failure mode of Multi-Agent Reinforcement Learning (MARL) methods. Methods such as Model-Free Opponent Shaping (M-FOS) and The Good Shepherd address this issue by shaping their co-player's learning into mutual cooperation. However, these methods fail to capture important co-player learning dynamics or do not scale to co-players parameterised by deep neural networks. To address these issues, we propose *Context and History Aware Other-Shaping* (CHAOS). A CHAOS agent is a meta-learner parameterised by a recurrent neural network that learns to shape its co-player over multiple trials. CHAOS considers both the *context* (inter-episode information), and *history* (intra-episode information) to shape co-players successfully. CHAOS also successfully scales to shaping co-players parameterised by deep neural networks. In a set of experiments, we show that CHAOS achieves state-of-the-art shaping in matrix games. We provide extensive ablations, motivating the importance of both context and history. CHAOS also successfully shapes on a complex grid-world-based game, demonstrating CHAOS's scalability empirically. Finally, we provide empirical evidence that, counterintuitively, the widely-used Coin Game environment does not require history to learn shaping because states are often indicative of past actions. This suggests that the Coin Game is, in contrast to common understanding, unsuitable for investigating shaping in high-dimensional, multi-step environments.

1 INTRODUCTION

Multi-agent learning has shown great success in strictly competitive (Silver et al., 2016) and fully cooperative settings (Foerster et al., 2019; Rashid et al., 2018). In competitive games, agents can learn competent Nash equilibrium strategies by iteratively best-responding to suitable mixtures of past opponents. Similarly, best-responding to rational co-players leads to the desirable equilibria in cooperative games (assuming joint training). In contrast, Nash equilibria often coincide with globally worst welfare outcomes in general-sum games, rendering the aforementioned learning paradigms ineffective. For example, in the iterated prisoner's dilemma (IPD) (Axelrod & Hamilton, 1981; Harper et al., 2017), naive best-response dynamics converge on unconditional mutual defection (Foerster et al., 2018).

These methods ignore a crucial factor: when multiple learning agents interact in a shared environment, the actions of one agent influence the environment and, often, the reward of other agents, which in turn influences their *learning dynamics*. For example, a car merging into the middle lane in heavy traffic makes it unattractive for fellow collision-averse motorists to move to the middle lane at the same time. Our paper investigates methods which allow agents to exploit this interconnection between their actions and the learning outcome of other agents, and leverage it to their advantage. Such "shaping" methods explicitly account for other agent's learning and have achieved promising results, e.g. discovering the famous tit-for-tat strategy in the IPD (Foerster et al., 2018; Letcher et al., 2019b; Willi et al., 2022; Balaguer et al., 2022; Lu et al., 2022). However, early shaping methods are myopic (only shape the next learning step of the co-player), require white-box access to the co-player's parameters and require higher-order derivatives. To overcome these shortcomings, both Model-Free Opponent Shaping (Lu et al., 2022, M-FOS) and The Good Shepherd (Balaguer et al., 2022, GS) frame shaping as a meta reinforcement learning problem. In these approaches, the meta-agent learns to shape others by observing full training runs of the co-players in each meta-training

episode before updating its policy. M-FOS and GS showed promising empirical success. However, both methods have shortcomings: M-FOS's meta-agent outputs a policy parameterisation for the inner-agent (similar to HyperNetworks (Ha et al., 2017)). This limits M-FOS to games where the policies can be represented compactly, such as in *infinitely-iterated* matrix games. While M-FOS does report results in a higher-dimensional game (in which the policies are represented by neural networks), it uses a hierarchical architecture to do so. GS does not output whole parameterisations but instead keeps its policy fixed during the entire duration of a trial, preventing it from using the training *context* to shape the co-player adaptively.

To address both issues, we propose Context and History Aware Other-Shaping[1] (CHAOS). In CHAOS, the meta-agent and the inner agent it controls are parameterised by a single recurrent neural network (RNN). A CHAOS agent meta-learns by retaining its hidden state throughout an entire meta-episode, similar to RL^2 (Duan et al., 2016) in single-agent RL. This hidden state enables CHAOS agents to react to two components of the co-player's learning: The *context* - inter-episode learning and the *history* - intra-episode behaviour. In shaping problems, *history* captures the co-player's current policies whilst context captures the co-player's learning rules. Together these two enable CHAOS to dynamically shape agents. Combining the meta-agent and the inner agent into one recurrent meta-learner avoids outputting policy parameterisations, unlike M-FOS.

We show that CHAOS discovers a ZD-extortion-like strategy in the *finitely-iterated* prisoner's dilemma (a more challenging setting than the *infinitely-iterated* PD where the environment is non-differentiable and where policies cannot be represented compactly). Moreover, we show that CHAOS matches or outperforms GS and M-FOS in iterated matrix games. CHAOS also matches state-of-the-art shaping against memory-based agents in the Coin Game, a grid-based environment where policies are represented by deep neural networks.

To summarise our contributions

- We introduce CHAOS, a shaping method capturing both learning context and history, suitable for high-dimensional games.
- We formalise the concept of history and context for shaping and analyse their respective roles empirically.
- We demonstrate state-of-the-art performance on a set of iterated matrix games.
- We identify a fundamental problem in the widely-used Coin Game.

2 Related Work

Opponent Shaping Many methods exist that explicitly account for their opponent's learning. Just like CHAOS, these approaches recognise that the actions of any one agent influence their co-players policy and seek to use this mechanism to their advantage (Foerster et al., 2018; Letcher et al., 2019a; Kim et al., 2021a; Willi et al., 2022). However, in contrast to CHAOS, these approaches require privileged information to shape their opponents. Finally, these models are myopic since anticipating many steps is intractable. (Balaguer et al., 2022, GS) and (Lu et al., 2022, M-FOS) solve the aforementioned issues by framing opponent shaping as a meta reinforcement learning problem, which CHAOS inherits and builds upon. The specific differences to M-FOS and GS will be the subject of Section 4.

Opponent Modelling Similarly to our work, opponent modelling tries to disentangle some aspects of other agents' policies from the environment (Mealing & Shapiro, 2017; Raileanu et al., 2018; Tacchetti et al., 2018). In contrast to our work, these approaches do not consider agents as learners. Furthermore, they do not observe agents at different stages of learning and thus, whilst modelling as non-stationary, do not observe learning dynamics (Synnaeve & Bessière, 2011). Finally, (Balaguer et al., 2022, GS), and (Lu et al., 2022, M-FOS) do not explicitly model any aspect of the opponent.

Multi-Agent Meta-Learning Multi-agent meta-learning approaches have also shown success in mixed-games with other learners (Al-Shedivat et al., 2018; Kim et al., 2021b; Wu et al., 2021). Similar to CHAOS, they take inspiration from meta reinforcement learning - their approach is to

[1] "Other" breaks with the line of seminal work on *opponent shaping*, but highlights the general-sum aspect

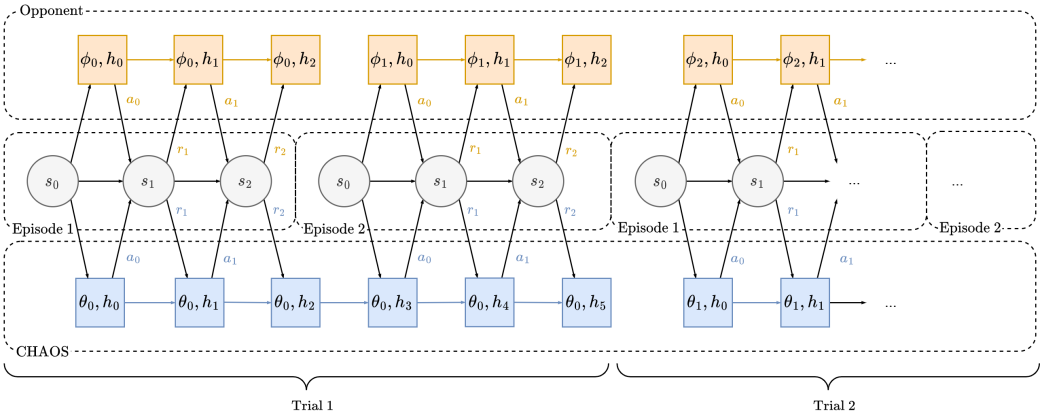


Figure 1: Example meta-learning interaction. CHAOS θ (blue) resets its hidden state h at the beginning of a trial, updates its hidden state after each environment interaction, then updates its parameters at the end of a trial. The opponent ϕ (orange) resets its parameters at the beginning of a trial, then updates its parameters at the end of each episode. Optionally, if the opponent uses memory, its hidden state h is reset at the beginning of a trial and updated after each environment interaction.

learn the optimal initial parameterisation for the shaper (Finn et al., 2017). In contrast, CHAOS uses an approach similar to RL^2, which trains an RNN-based agent to implement efficient learning for its next task. Furthermore, CHAOS is optimised using evolutionary strategies, allowing it to consider much longer time horizons than policy-gradient metods (Schulman et al., 2017).

3 Background

Partially Observable Stochastic Game (POSG) A POSG is given by the tuple $\mathcal{M} = \langle n, \mathcal{A}, \mathcal{O}, S, \mathcal{T}, \mathcal{I}, \mathcal{R}, \gamma \rangle$, where $\mathcal{A}$, $\mathcal{O}$, and S denote the action, observation, and state space, respectively. These parameters can be distinct at every time step and also incorporated into the transition function $\mathcal{T} : S \times \mathbf{A} \to \Delta S$, where $\mathbf{A} \equiv \mathcal{A}^n$ is the joint action of all agents. Each agent draws individual observations according to the observation function $\mathcal{I} : S \times N \to \mathcal{O}$ and obtains a reward according to their reward function $\mathcal{T} : S \times \mathbf{A} \times N \to \mathbb{R}$ where $N = \{1, \ldots, n\}$. POSGs can represent general-sum games. The single player case, $\mathcal{I} = \{1\}$, of POSGs are Partially Observable Markov Decision Processes (POMDPs).

RL^2 CHAOS takes an RL^2-like approach to meta-learning. RL^2 is a single-agent meta-reinforcement learning method where both meta-agent and inner agent are parameterised by a recurrent neural network (ϕ, h), where ϕ are the network parameters and h the hidden state. RL^2 samples MDPs from a distribution $\rho_{\mathcal{P}} : \mathcal{P} \to \mathbb{R}_+$ and interacts with each sample MDP for a number of episodes E, called a trial. Importantly, the RL^2 agent retains its hidden state h across all episodes in a trial and resets only when it faces a new trial. The objective is to maximise the expected total discounted reward over a trial t instead of a single episode

$$\mathbb{E}_{\phi^t}\left[\sum_{l=0}^{L} \gamma^l r\left(s_l, a_l\right)\right]$$

where $L = K * E$ and K is the episode length. The RL^2 agent is thus encouraged to use all the information captured in its hidden state h. RL^2 agents have been shown to scale to high-dimensional problem settings.

Good Shepherd GS formalises shaping as a meta-learning problem over a sequence of trials of length T. Each trial contains E episodes. In each trial, GS shapes a new co-player in a POSG $\mathcal{M}$, where $(\phi_i, \boldsymbol{\phi}_{-i})$ correspond to GS' and the co-players' parameters, respectively. During a trial t, GS uses a fixed policy ϕ_i^t. At the end of each inner episode, the co-players update their parameters with

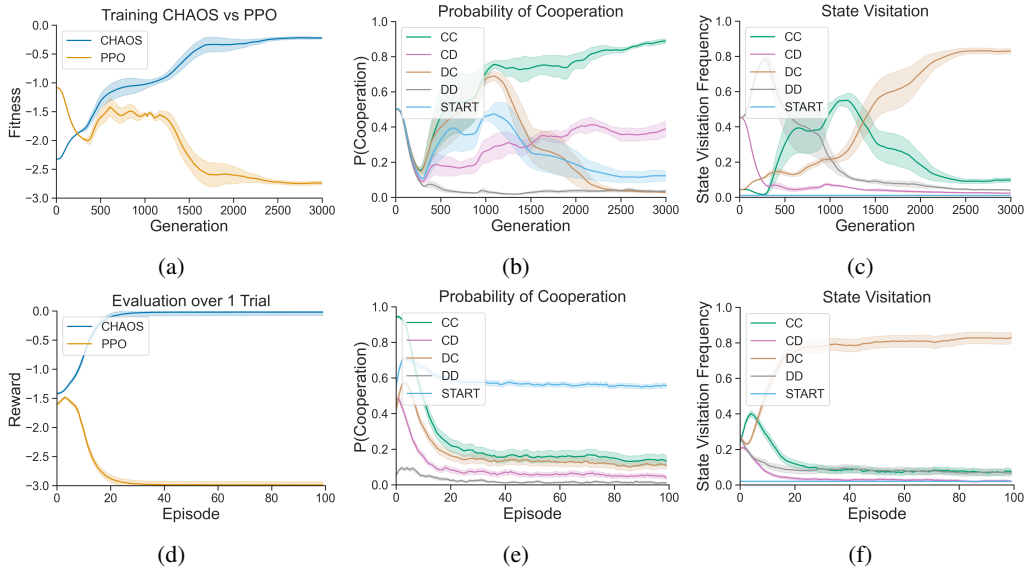


Figure 2: Training results in the finite IPD over 5 seeds for CHAOS. The upper row displays the (a) fitness, (b) conditional probability of cooperation, and (c) state visitation. The state visitations for DC reach 80% after 2000 generations of training, indicating that CHAOS has learned to extort its opponent. The lower row is evaluation results over a single trial composed of 100 inner episodes over 20 seeds, where (d) shows the reward, (e) CHAOS's conditional probability of cooperation, and (f) state visitation.

respect to the episodic return $J^e_{-i} = \sum_{k=0}^{K} r^k_{-i}(\phi^t_i \phi^e_{-i})$, where K is the length of an episode. For example, if the co-players were Naive Learners, the update looks as follows:

$$\boldsymbol{\phi}^{e+1}_{-i} = \boldsymbol{\phi}^{e}_{-i} + \alpha \nabla_{\boldsymbol{\phi}^{-}_{e}{}^{i}} J^e \left(\phi^i_t, \boldsymbol{\phi}^{-i}_e\right),$$

where α is the learning rate. GS optimises the meta-return $\bar{J} = \sum_e^E J^e_i$ (summed over all episodes) at the end of a trial using Evolutionary Strategies. The policy is parameterised by a feed-forward network, thus lacking both history and context memory. GS, thus, cannot adapt to changing learning dynamics of the co-player. We show this to be detrmin our results (see Section 6) that this is detrimental in some games.

Model-Free Opponent Shaping M-FOS frames opponent shaping as a meta reinforcement task. More specifically, the meta-task is formulated as a POMDP $\langle \overline{\mathcal{S}}, \overline{\mathcal{A}}, \Omega, \overline{\mathcal{O}}, \overline{\mathcal{P}}, \overline{\mathcal{R}}, \bar{\gamma} \rangle$ over an underlying general-sum game, represented by a POSG $\mathcal{M}$. In the POMDP, the meta-state $\overline{\mathcal{S}}$ spans the policies of every player in the underlying POSG: $\bar{s}_e = \left(\phi^i_{e-1}, \boldsymbol{\phi}^{-i}_{e-1}\right) \in \mathcal{S}$. The meta-action space $\overline{\mathcal{A}}$ consists of the policy parameterisation of the underlying inner agent playing the game for the meta-agent. At each meta-episode, conditioned on both agent's policies, M-FOS outputs parameters of the next inner-agents (similar to a HyperNetwork (Ha et al., 2017)), i.e., $\bar{a}_e = \phi^i_e \sim \pi_\theta\left(\cdot \mid \bar{o}_e\right)$, where θ is the parameters of the M-FOS agent. The meta-reward is the return of the inner agent over one inner episode, $\bar{r}_e = \sum_{k=0}^{K} r^i_k \left(\phi^i_e, \boldsymbol{\phi}^{-i}_e\right)$. This Hypernetwork-like approach is M-FOS' main shortcoming - it is difficult to scale to complex inner-agent policy parameterisations. To scale to complex inner-agent policy parameterisations that are used in the Coin Game environment, they use a hierarchical architecture in which the meta-agent instead outputs a *conditioning vector* for the inner agent that contains context information. M-FOS is optimised using Evolution Strategies (Salimans et al., 2017) for the iterated matrix games and uses PPO (Schulman et al., 2017) for the Coin Game environment.

Evolution Strategies (ES) CHAOS uses Evolution Strategies (Salimans et al., 2017, ES) to optimise the meta-agent. ES is a model-free optimisation method. Let $F : \mathbb{R}^d \to \mathbb{R}$ be some function we want to optimise over. Instead of optimising the objective directly, ES blurs $F(\mathbf{x})$ with Gaussian

noise

$$\mathbb{E}_{\epsilon\sim N(\mathbf{0},I_d)}[F(\mathbf{x}+\sigma\epsilon)]\,,$$

where σ is a hyper-parameter controlling how much Gaussian noise is added. This allows using the following simple gradient estimator:

$$\nabla_{\mathbf{x}}\mathbb{E}_{\epsilon\sim N(\mathbf{0},I_d)}[F(\mathbf{x}+\sigma\epsilon)] = \mathbb{E}_{\epsilon\sim N(\mathbf{0},I_d)}\left[\frac{\epsilon}{\sigma}F(\mathbf{x}+\sigma\epsilon)\right]\,,$$

allowing the optimization of non-differentiable functions using gradient descent methods. ES bypasses the credit assignment problem by directly optimizing in the parameter space of the model instead of the policy space and is thus suitable for long-time horizon problems (Salimans et al., 2017).

4 METHOD

We assume a set of POSGs M and a distribution we can sample from ρ_M. We also assume a set of initial learners ϕ_0 and a corresponding distribution ρ_ϕ, as shaping acknowledges other learners within the environment, in contrast to RL^2. Just like in GS, we define a *inner episode* to be a finite sequence of interactions within a fixed POSG and fixed initial learner, and a *trial* to be a sequence of inner episodes.

Algorithm 1: General CHAOS

```
Initialise Shaper parameters θ.
while true do
    for n ∈ {1, ..., N} do
        Sample shaper parameters θ_m from ES
        Initialise co-player parameters φ_0.
        for t = 0 to T do
            Reset Environment
            Gather trajectories τ_t given θ_n, φ_t
            Update φ_{t+1} according to respective
            learning algorithm
        end for
        Calculate fitness F_n for trial
    end for
    Update φ with ES(F)
end while
```

Figure 1 illustrates the interaction between agents and the environment. At the start of a trial, co-players $\boldsymbol{\phi}_{-i}$ ρ_ϕ and a new game (POSG) $\rho_{\mathcal{M}}$ are drawn. The shaping agents' policy ϕ_i and hidden state h are initialised. During an episode of length K, upon receiving a state, agents take their respective actions, a_i^k. At each time step in the episode, the internal state of the shaping agent is updated: $h_{k+1} = f(h_k)$. On receiving actions, the POSG returns rewards r_i^k, new observations o_{k+1}^i and a done flag d, indicating if an episode has ended.

When a inner episode terminates, the learner takes a gradient update maximising total *episode* return, $J_{-i}^e = \sum_{k=0}^K r_{-i}^k(\phi_i^t, \boldsymbol{\phi}_{-i}^e)$. The updated learner $\boldsymbol{\phi}_{-i}^{e+1}$ and the shaper's hidden state h_K are passed to the next episode. This process is repeated over E episodes in a trial. When a trial terminates, the shaper's policy is updated, maximising total *trial* reward, $\bar{J} = \sum_e^E J_i^e$, via an evolutionary strategy.

We use Evolution Strategies (Salimans et al., 2017) to maximise the expected fitness across a population. At the start of each generation, a population of CHAOS agents of size M and N Naive Learners (NL) are initialised. During a generation, the population plays against copies of the Naive Learner in parallel for a series of E inner episodes. At the end of an inner episode, each copy of the Naive Learner performs a gradient update. At the end of the trial, CHAOS performs a gradient update in the direction of the maximum expected fitness, and the next generation begins.

5 EXPERIMENTS

5.1 ENVIRONMENTS

Iterated Prisoner's Dilemma The prisoner's dilemma is a well-known and widely studied general-sum game illustrating that two rational agents may not cooperate even if it is globally optimal. The players choose to either cooperate (C) or defect (D) and receive a payoff according to Table 3. In the *iterated* prisoner's dilemma (IPD), the agents repeatedly play the prisoner's dilemma and can observe the previous decisions.

Past research has used the infinitely IPD in their experiments (Foerster et al., 2018; Letcher et al., 2019b; Willi et al., 2022; Lu et al., 2022; Balaguer et al., 2022). In the infinite version, players submit

Table 1: Converged Reward in the (a) IPD and (b) IMP

(a)

	PPO
CHAOS	**(-0.13 ± 0.02, -2.84 ± 0.05)**
M-FOS	(-0.60 ± 0.14, -2.34 ± 0.14)
GS	(-0.97 ± 0.03, -1.26 ± 0.10)
PPO	(-2.00 ± 0.00, -2.00 ± 0.00)

(b)

	PPO
CHAOS	**(0.86 ± 0.02, -0.86 ± 0.02)**
M-FOS	(0.83 ± 0.09, -0.83 ± 0.09)
GS	(-0.01 ± 0.01, 0.01 ± 0.01)
PPO	(0.00 ± 0.00, 0.00 ± 0.00)

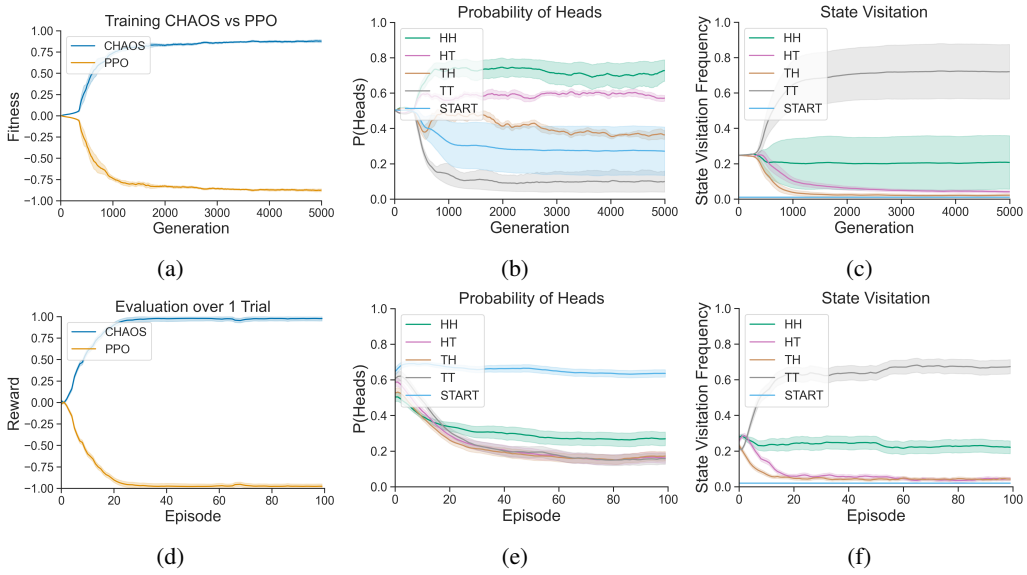


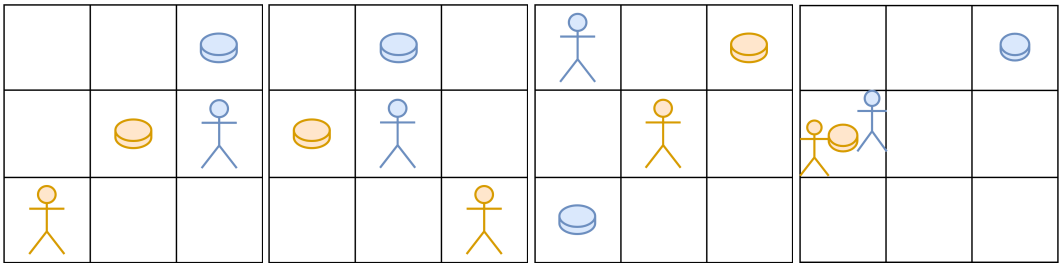
Figure 3: Training results in the IMP over 5 seeds for CHAOS. (a) Fitness (b) Empirical probability of Cooperative action conditioned by state and (c) state visitation. Evaluation results over a single trial composed of 100 inner episodes over 20 seeds. (d) Average Reward (per step) (e) CHAOS's probability of cooperation conditioned on state and (f) state visitation.

a policy represented by five parameters, where each parameter is the probability of cooperation after each state given a one-step history $(CC, CD, DC, DD, \text{start})$. Press & Dyson (2012) showed that having access to the last state is sufficient for acting optimally. The infinite version is a differentiable game Balduzzi et al. (2018) as the exact value function can be calculated directly from the policies, thus accessing exact gradients is possible and optimization tractable. In our work, we consider the finitely IPD (f-IPD), and we do not take advantage of exact value functions, resulting in a version of the game that is more similar to current reinforcement learning environments. In the f-IPD, the agents do not submit a full strategy but take an individual action (either C or D) at each timestep.

Over repeated interactions, IPD produces a spectrum of interesting behaviours. In particular, two cases are of interest in this work: 1) Cooperation, and 2) Zero-Determinant (ZD) Extortion strategies. In Cooperation, agents are shaped sufficiently to CC and choose not defect, even though this would increase short-term rewards. In ZD-Extortion (Press & Dyson, 2012), co-players cooperate while allowing the shaper to enforce a linear relationship between their own payoff and that of the co-player, thus inducing behaviours that are more favourable than mutual cooperation.

Iterated Matching Pennies The Iterated Matching Pennies (IMP) is an iterated matrix game like the IPD. The players choose either heads (H) or tails (T), and receive a payoff according to the choices of both players. In contrast to the IPD, which is a general-sum game, IMP is a zero-sum game. The only equilibrium strategy for each one-memory agent is to play a random policy, resulting in an expected joint payoff of (0,0). It is only with intra-episode memory that a shaper can observe a co-player's policy and begin shaping.

Coin Game The Coin Game (Lerer & Peysakhovich, 2017) is a multi-agent, wrap-around grid-world environment that simulates social dilemmas (like the IPD) with high-dimensional states and



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

Figure 4: Illustration of (a) a state observed by both agents in the non-egocentric Coin Game and (b)-(c) the same state observed by the blue and orange agent, respectively, in the egocentric Coin Game. (d) Illustration of a special state where memory-less agents can infer conventions.

multi-step actions. Two players – blue and orange – move around a grid and pick up blue and orange coloured coins. When a player picks up a coin of its own colour, the player receives a reward of $+1$. When a player picks up a coin of the co-player's colour, the player also receives a reward of $+1$ and the co-player receives a reward of -2. If a coin gets picked up, a new coin of the same colour appears in a random location on the grid. If both agents reach a coin simultaneously, then both agents pick up that coin (the coin is duplicated). The episodes are of fixed length. When both players pick up coins without regard to colour, the expected reward is 0. In contrast to the IPD, the Coin Game requires learning from high-dimensional states, a task that current shaping methods struggle to learn.

5.2 Baseline Comparisons

We compare our method against three baselines: a Naive Learner, M-FOS and GS. A Naive Learner (NL) does not account for the learning of the co-player. It updates at the end of each inner-episode with learning rate α:

$$\phi^i_{t+1} = \phi^i_t + \alpha \nabla_{\phi^i_t} \mathcal{R}^i(\phi^i_t, \phi^{-i}_t) \tag{1}$$

In the IPD, our NL is parameterised as a tabular policy trained using PPO (Schulman et al., 2017). In the Coin Game, the NL is parameterised by a recurrent neural network trained using PPO. The specific implementation details are provided in Appendix B.

For M-FOS and GS, we optimise both methods using Evolution Strategies (Salimans et al., 2017), in line with the original implementations (Lu et al., 2022; Balaguer et al., 2022). For M-FOS, we use the hierarchical architecture used in its Coin Game results since we are using neural networks for these environments instead of simple tabular policies. The implementation details for M-FOS and GS are provided in Appendix B and Appendix C

In every game, CHAOS is parameterised as a recurrent neural network and is trained using Evolution Strategies (Salimans et al., 2017). We used the Jax library (Bradbury et al., 2018) with the Haiku framework (Hennigan et al., 2020) to implement our neural networks. For the Evolution strategies, we relied on the Evosax library (Lange, 2022). Our experiments were performed on NVIDIA A40 and V100 GPUs. Addditional implementation details and hyperparameters for each game are provided in Appendix C. Furthermore, the whole codebase will be released upon acceptance.

5.3 Ablations

In order to evaluate CHAOS' effectiveness we apply the following ablations.

The Hardstop Challenge - during a trial, after k episodes the co-player no longer takes learner updates. In the situation where the co-player no longer updates, optimal behaviour would be to exploit this fixed policy (effectively stop shaping). We choose $k = 2$ to be less than the number of episodes required for a shaper to reach to ZD- Extortion-like policies. This challenge tests if shapers' can 1) identifying the sudden change in an co-player's learning dynamics 2) react and deploy a more suitable exploitative policy. We evaluate CHAOS against GS, to compare context and context-less shaping methods.

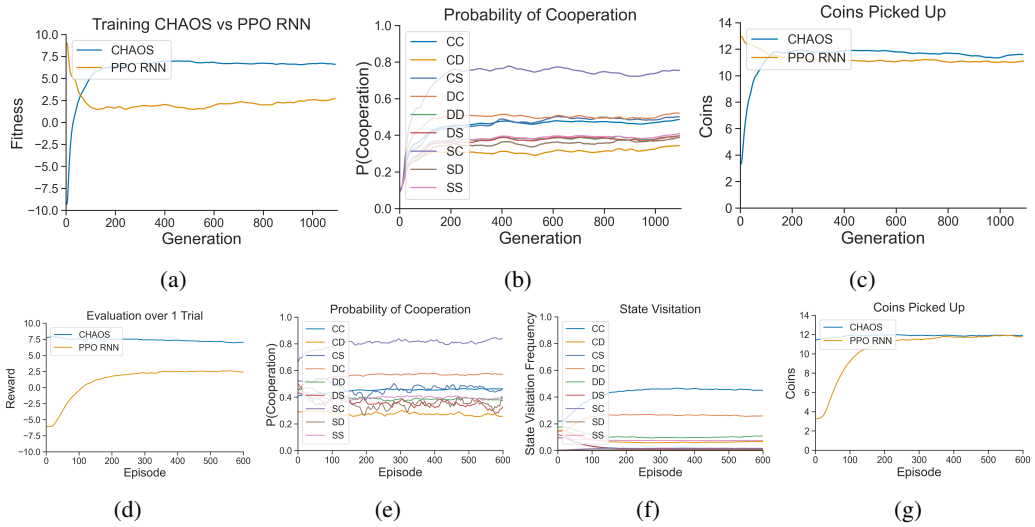


Figure 5: Training results of CHAOS vs. PPO RNN in the egocentric Coin Game. (a) Fitness, (b) both agent's frequency of picking up its own colour coin, (c) the number of coins picked up per episode. Evaluation run of a single trial of 600 inner episodes. (d) Reward, (e) both agent's frequency of picking up its own color coin, (f) state visitation, and (g) the number of coins picked up per episode.

The Only-History Challenge - we reset the hidden state of shapers between episodes, removing their ability to use *context* to shape. In this challenge, shapers must infer co-player's current policy by only using the history. We evaluate the shaper within the IMP environment, over different episode lengths, to limit the relative strength of *history*.

6 RESULTS

In this section we report and compare the results between CHAOS and our baselines on our environments. Firstly we find that CHAOS shapes achieves state-of-the-art results in the IPD. On the IMP game, CHAOS achieves state-of-the-art results, outperforming M-FOS, GS and other baselines. Next we demonstrate that CHAOS is scalable, as it achieves comparable shaping in the Coin Game. Finally through a series of ablations we demonstrate the importance of context and history for effective shaping.

Iterated Prisoner's Dilemma In the IPD, CHAOS shapes its co-player more aggressively than the baselines (see Table 1a), achieving an average return of -0.13 per episode. However, all shaping baselines reach ZD-extortion-like policy. CHAOS switches policies during an co-player's learning, switching from cooperation to exploitation (see Figure 2). In Figure 2f, we display the state visitation over one trial. It shows that a fully trained CHAOS agent pursues a tit-for-tat like-strategy to encourage cooperation within the first 5 episodes before pursing an excessively exploitative policy. At this point, the co-player is shaped, as is it unable to move to another equilibrium.

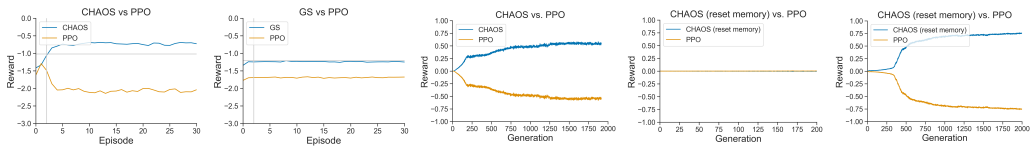
Iterated Matching Pennies In the IMP, CHAOS exploits its opponent to achieve a score of $(0.80, -0.80)$ (see Table 1b). As expected, GS cannot shape the opponent, achieving a score close to the Nash Equilibrium. Without having any context, it is not possible to shape the opponent because the opponent can also switch to a random strategy to at least achieve a score of 0. Thus dynamic-shaping, is required to find exploitative strategies. We find that M-FOS another dynamic shaping method is able to find exploitation too.

Coin Game We see that CHAOS outperforms M-FOS in Table 2. This provides evidence that CHAOS is scalable to more complex co-players policies. Both GS and CHAOS demonstrate shaping, as co-player picks up more of its own coin than the shaper (see Figure 5).

Table 2: Head-to-head results in the *egocentric* Reward per episode and the standard deviation (over 5 seeds) for Coin Game.

	PPO RNN
CHAOS	$(6.51 \pm 0.46, 2.71 \pm 0.11)$
M-FOS	$(2.67 \pm 0.52, 3.94 \pm 0.15)$
GS	**(6.72 ± 0.72, 2.39 ± 0.10)**

We found GS produces comparable results to CHAOS. At first this is surprising, since GS is a feedforward network and does not have access to the history. Therefore, in principle it should not be able to retaliate against a defecting agents since it has no memory of their past actions. However, close investigation of the problem setting shows that due to the specific environment dynamics, the *current state* is often indicative of *past actions*. For example, when the two agents are on the same square, in all likelihood one of the agents defected (see Figure 4d). Similarly, if the agents are currently standing a on a coin of a given colour, this coin was picked up on the last time step. This illustrates that Coin Game allows for simple shaping strategies that do not require *context* or *history*, limiting its utility as a benchmark for investigating these aspects .



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

Figure 6: The Hardstop Challenge: Average reward for a single trial (a) CHAOS and (b) GS against a naive learner in the IPD. Note that once co-player stops learning, CHAOS is able to dynamically switch to much more exploitative behaviour (reward=-0.5) whereas GS's policy remains fixed. The Only-History Challenge: Training curves in the IMP with episode length = 2 for (c) CHAOS and (d) CHAOS without memory. Note that in short time-spans, where history can not be used context can enable shaping. Additionally (e) CHAOS without history in IMP with episode length = 100 shows with sufficient timespans, history can be used to shape.

Ablations CHAOS outperforms GS on the hardstop challenge. CHAOS demonstrates dynamic shaping by switching strategies at $t = 2$ episodes, when the hardstop is triggered (see Fig. 6c). In contrast, GS's policy is fixed throughout a trial and thus can not exploit the hardstop. In the Only History challenge, CHAOS (reset memory) learns to exploit its opponent either with *context* or, with a long enough episode length, *history* (see Fig. 6).

7 Conclusion

In this paper, we introduce CHAOS, a shaping method capturing both learning context and history, suitable for high-dimensional games. We formalise the concept of history and context for shaping and analyse their respective roles empirically. We demonstrate state-of-the-art performance on a set of iterated matrix games. We identify a fundamental problem in the widely-used Coin Game.

When multiple agents interact in a shared environment, the actions of each any one agent influence the rewards and environment faced by others, and through their learning ultimately affect their behaviour. Shaping, i.e. constructing agents that can effectively leverage this interconnection, has emerged as a sub-field in Mult-agent Reinforcement learning and has received considerable attention in recent years. CHAOS substantially expands the current capabilities of shaping agents by allowing them to react to changes to the co-players' learning dynamics as well as to predictable patterns in their within-episode behaviour, thus resulting in significantly more effective shaping.

REFERENCES

Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. In *International Conference on Learning Representations*, 2018. 2

Robert Axelrod and William D Hamilton. The evolution of cooperation. *science*, 211(4489):1390–1396, 1981. 1

Jan Balaguer, Raphael Koster, Christopher Summerfield, and Andrea Tacchetti. The good shepherd: An oracle agent for mechanism design. 2022. 1, 2, 5, 7

David Balduzzi, Sébastien Racanière, James Martens, Jakob N. Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 363–372, 2018. 6

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`. 7

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl$ˆ2$: Fast reinforcement learning via slow reinforcement learning. arXiv preprint arXiv:1611.02779, 2016. 2

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, 2017. 3

Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130, 2018. 1, 2, 5

Jakob Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 1942–1951. PMLR, 2019. 1

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *5th International Conference on Learning Representations*, 2017. 2, 4

Marc Harper, Vincent Knight, Martin Jones, Georgios Koutsovoulos, Nikoleta E. Glynatsi, and Owen Campbell. Reinforcement learning produces dominant strategies for the iterated prisoner's dilemma. *PLOS ONE*, 12(12):e0188046, 2017. 1

Tom Hennigan, Trevor Cai, Tamara Norman, and Igor Babuschkin. Haiku: Sonnet for JAX, 2020. URL `http://github.com/deepmind/dm-haiku`. 7

Dong-Ki Kim, Miao Liu, Matthew Riemer, Chuangchuang Sun, Marwa Abdulhai, Golnaz Habibi, Sebastian Lopez-Cot, Gerald Tesauro, and Jonathan P. How. A policy gradient algorithm for learning to learn in multiagent reinforcement learning. In *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5541–5550, 2021a. 2

Dong Ki Kim, Miao Liu, Matthew D Riemer, Chuangchuang Sun, Marwa Abdulhai, Golnaz Habibi, Sebastian Lopez-Cot, Gerald Tesauro, and Jonathan How. A policy gradient algorithm for learning to learn in multiagent reinforcement learning. In *International Conference on Machine Learning*, pp. 5541–5550. PMLR, 2021b. 2

Robert Tjarko Lange. evosax: Jax-based evolution strategies, 2022. URL `http://github.com/RobertTLange/evosax`. 7

Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *CoRR*, abs/1707.01068, 2017. 6

Alistair Letcher, David Balduzzi, Sébastien Racanière, James Martens, Jakob N. Foerster, Karl Tuyls, and Thore Graepel. Differentiable game mechanics. *J. Mach. Learn. Res.*, 20:84:1–84:40, 2019a. 2

Alistair Letcher, Jakob N. Foerster, David Balduzzi, Tim Rocktäschel, and Shimon Whiteson. Stable opponent shaping in differentiable games. In *7th International Conference on Learning Representations*, 2019b. 1, 5

Christopher Lu, Timon Willi, Christian A Schroeder De Witt, and Jakob Foerster. Model-free opponent shaping. In *International Conference on Machine Learning*, pp. 14398–14411. PMLR, 2022. 1, 2, 5, 7, 12

Richard Mealing and Jonathan L. Shapiro. Opponent modeling by expectation-maximization and sequence prediction in simplified poker. *IEEE Trans. Comput. Intell. AI Games*, 9(1):11–24, 2017. 2

William H. Press and Freeman J. Dyson. Iterated Prisoner's Dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, 109(26): 10409–10413, 2012. ISSN 0027-8424. 6

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, pp. 4257–4266. PMLR, 2018. 2

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pp. 4295–4304. PMLR, 2018. 1

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864, 2017. 4, 5, 7

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017. 3, 4, 7

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016. 1

Gabriel Synnaeve and Pierre Bessière. A bayesian model for opening prediction in RTS games with application to starcraft. In *IEEE Conference on Computational Intelligence and Games*, pp. 281–288, 2011. 2

Andrea Tacchetti, H Francis Song, Pedro AM Mediano, Vinicius Zambaldi, Neil C Rabinowitz, Thore Graepel, Matthew Botvinick, and Peter W Battaglia. Relational forward models for multi-agent learning. *arXiv preprint arXiv:1809.11044*, 2018. 2

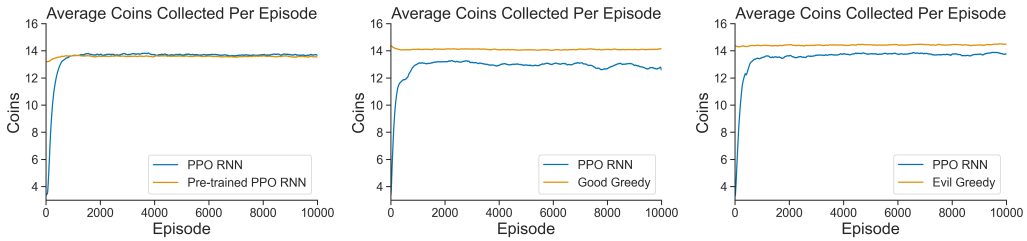
Timon Willi, Alistair Letcher, Johannes Treutlein, and Jakob N. Foerster. COLA: consistent learning with opponent-learning awareness. In *International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23804–23831, 2022. 1, 2, 5

Zhe Wu, Kai Li, Enmin Zhao, Hang Xu, Meng Zhang, Haobo Fu, Bo An, and Junliang Xing. L2E: learning to exploit your opponent. arXiv preprint arXiv:2102.09381, 2021. 2

A Appendix

A.1 Experimental Protocol

We define an experimental protocol for producing the correct conditions for shaping



(a) (b) (c)

Figure 7: Results of the experimental protocol verifying that PPO RNN learns to play the egocentric Coin Game against (a) pre-trained PPO RNN (b) Good Greedy (c) Evil Greedy. Notice that the agent learns to pick up roughly the same number of coins per episode as its competent opponents.

Experimental Protocol We verify that an agent can learn against three competent opponents. The first agent is *Good* Greedy, picking up coins indiscriminately but prioritizing its own colour coin if it is equidistant from two coins; the second agent is *Evil* Greedy, picking up coins indiscriminately but prioritizing its opponent's coin if it is equidistant from two coins; the third agent is the current agent pre-trained to competency via self-play. After verifying that an agent learns competency against these three opponents, we set the parameters of the trial to reflect the time and scale required for the agent to become competent against a competent opponent.

Sanity We verify that PPO RNN can learn against competent opponents in the Coin Game in Figure 7. PPO RNN learns to pick up the same number of coins as a pre-trained PPO RNN, Good Greedy, and Evil Greedy. These hyperparameters are the same ones used during meta-learning.

Coin Game Adjustments In the Coin Game, agents struggle to learn (via reinforcement learning) when trained against a pre-trained opponent. On inspection of trajectories, we found that competent agents removed a sufficient amount of coins to restrict reinforcement learners ability to capture signal from the game.

To address this, we show that adjusting the observations such that an agent receives to an *egocentric* viewpoint (i.e. an agent always observes that it is in the centre of the grid) leads to competency against a competent opponent. In this case, we measured competency as an agent's ability to pick up coins. *Competent*agents were those who picked up a similar number of coins to those trained against a stationary agent. Throughout the rest of the paper, we refer to the original version of Coin Game as *non-egocentric* Coin Game and the modified observation version as *egocentric* Coin Game. In addition, we deviate from the original 5 by 5 version of Coin Game to a 3 by 3 version, following the setting used in (Lu et al., 2022).

B Matrix Game Hyper-parameters

	C	D
C	(-1,-1)	(0, -3)
D	(-3, 0)	(-2, -2)

Table 3: Payoff matrix of IPD.

	H	T
H	(1,-1)	(-1, 1)
T	(-1, 1)	(1, 1)

Table 4: Payoff matrix of IMP.

We present the hyper-parameters used for training in both the iterated prisoner's dilemma and the matching pennies game.

Hyperparameter	Value
Number of Actor Hidden Layers	1
Number of Critic Hidden Layers	1
Torso GRU Size	[25]
Length of Meta-Episode	100
Length of Inner Episode	100
Number of Generations	5000
Batch Size	100
Population Size	1000
OpenES sigma init	0.04
OpenES sigma decay	0.999
OpenES sigma limit	0.01
OpenES init min	0.0
OpenES init max	0.0
OpenES clip min	-1e10
OpenES clip max	1e10
OpenES lrate init	0.01
OpenES lrate decay	0.9999
OpenES lrate limit	0.001
OpenES beta 1	0.99
OpenES beta 2	0.999
OpenES eps	1e-8

Table 5: Hyperparameters for CHAOS in Iterated Prisoner's Dilemma

Hyperparameter	Value
Number of Minibatches	4
Number of Epochs	2
Gamma	0.96
GAE Lambda	0.95
PPP clipping epsilon	0.2
Value Coefficient	0.5
Clip Value	True
Max Gradient Norm	0.5
Entropy Coefficient Start	0.02
Entropy Coefficient Horizon	2000000
Entropy Coefficient End	0.001
Learning rate	1
ADAM epsilon	1e-5

Table 6: Hyperparameters for Tabular-PPO in Iterated Prisoner's Dilemma

Hyperparameter	Value
Number of Actor Hidden Layers	2
Number of Critic Hidden Layers	2
Network Hidden Size	[16, 16]
Length of Meta-Episode	100
Length of Inner Episode	100
Number of Generations	5000
Batch Size	100
Population Size	1000
OpenES sigma init	0.04
OpenES sigma decay	0.999
OpenES sigma limit	0.01
OpenES init min	0.0
OpenES init max	0.0
OpenES clip min	-1e10
OpenES clip max	1e10
OpenES lrate init	0.01
OpenES lrate decay	0.9999
OpenES lrate limit	0.001
OpenES beta 1	0.99
OpenES beta 2	0.999
OpenES eps	1e-8

Table 7: Hyperparameters for GS in Iterated Prisoner's Dilemma

Hyperparameter	Value
Number of Actor Hidden Layers	1
Number of Critic Hidden Layers	1
Actor GRU Hidden Size	16
Critic GRU Hidden Size	16
Meta Agent Gru Hidden Size	16
Hidden Layer Size	16
Length of Meta-Episode	100
Length of Inner Episode	100
Number of Generations	5000
Batch Size	100
Population Size	1000
OpenES sigma init	0.04
OpenES sigma decay	0.999
OpenES sigma limit	0.01
OpenES init min	0.0
OpenES init max	0.0
OpenES clip min	-1e10
OpenES clip max	1e10
OpenES lrate init	0.01
OpenES lrate decay	0.9999
OpenES lrate limit	0.001
OpenES beta 1	0.99
OpenES beta 2	0.999
OpenES eps	1e-8

Table 8: Hyperparameters for MFOS in Iterated Prisoner's Dilemma

C COIN GAME HYPERPARAMETERS

Hyperparameter	Value
Number of Actor Hidden Layers	1
Number of Critic Hidden Layers	1
Torso Gru Size	[16]
Length of Meta-Episode	600
Length of Inner Episode	16
Number of Generations	3000
Batch Size	100
Population Size	4000
OpenES sigma init	0.04
OpenES sigma decay	0.999
OpenES sigma limit	0.01
OpenES init min	0.0
OpenES init max	0.0
OpenES clip min	-1e10
OpenES clip max	1e10
OpenES lrate init	0.01
OpenES lrate decay	0.9999
OpenES lrate limit	0.001
OpenES beta 1	0.99
OpenES beta 2	0.999
OpenES eps	1e-8

Table 9: Hyperparameters for EARL in Iterated Matching Pennies

Hyperparameter	Value
Number of Minibatches	8
Number of Epochs	2
Gamma	0.96
GAE Lambda	0.95
PPO clipping epsilon	0.2
Value Coefficient	0.5
Clip Value	True
Max Gradient Norm	0.5
Anneal Entropy	False
Entropy Coefficient Start	0.02
Entropy Coefficient Horizon	2000000
Entropy Coefficient End	0.001
LR Scheduling	False
Learning Rate	0.005
ADAM Epsilon	1e-5
With CNN	False

Table 10: Hyperparameters for PPO Memory and Tabular in the Coin Game

Hyperparameter	Value
Number of Actor Hidden Layers	1
Number of Critic Hidden Layers	1
Hidden Size	[16]
Length of Meta-Episode	600
Length of Inner Episode	16
Number of Generations	3000
Batch Size	100
Population Size	4000
OpenES sigma init	0.04
OpenES sigma decay	0.999
OpenES sigma limit	0.01
OpenES init min	0.0
OpenES init max	0.0
OpenES clip min	-1e10
OpenES clip max	1e10
OpenES lrate init	0.01
OpenES lrate decay	0.9999
OpenES lrate limit	0.001
OpenES beta 1	0.99
OpenES beta 2	0.999
OpenES eps	1e-8

Table 11: Hyperparameters for GS in Coin Game

Hyperparameter	Value
Number of Actor Hidden Layers	1
Number of Critic Hidden Layers	1
Actor GRU Hidden Size	16
Critic GRU Hidden Size	16
Meta Agent Gru Hidden Size	16
Hidden Layer Size	16
Length of Meta-Episode	100
Length of Inner Episode	100
Number of Generations	5000
Batch Size	100
Population Size	1000
OpenES sigma init	0.04
OpenES sigma decay	0.999
OpenES sigma limit	0.01
OpenES init min	0.0
OpenES init max	0.0
OpenES clip min	-1e10
OpenES clip max	1e10
OpenES lrate init	0.01
OpenES lrate decay	0.9999
OpenES lrate limit	0.001
OpenES beta 1	0.99
OpenES beta 2	0.999
OpenES eps	1e-8

Table 12: Hyperparameters for MFOS in Coin Game