MODEL-FREE OPPONENT SHAPING

Christopher Lu, Timon Willi, Christian Schroeder de Witt, Jakob Foerster
Foerster AI Research
University of Oxford
christopher.lu@eng.ox.ac.uk

ABSTRACT

In general-sum games, the interaction of self-interested learning agents commonly leads to collectively worst-case outcomes, such as defect-defect in the iterated prisoner's dilemma (IPD). To overcome this, some methods, such as Learning with Opponent-Learning Awareness (LOLA), shape their opponents' learning process. However, these methods are *myopic* since only a small number of steps can be anticipated, are *asymmetric* since they treat other agents as naive learners, and require the use of *higher-order derivatives*, which are calculated through white-box access to an opponent's differentiable learning algorithm. To address these issues, we propose Model-Free Opponent Shaping (M-FOS). M-FOS learns in a *meta-game* in which each meta-step is an episode of the underlying ("inner") game. The meta-state consists of the inner policies, and the meta-policy produces a new inner policy to be used in the next episode. M-FOS then uses generic *model-free optimisation methods* to learn meta-policies that accomplish long-horizon opponent shaping. Empirically, M-FOS near-optimally exploits naive learners and other, more sophisticated algorithms from the literature. For example, to the best of our knowledge, it is the first method to learn the well-known Zero-Determinant (ZD) extortion strategy in the IPD. In the same settings, M-FOS leads to socially optimal outcomes under *meta-self-play*. Finally, we show that M-FOS can be scaled to high-dimensional settings.

1 INTRODUCTION

While much past work in multi-agent reinforcement learning (MARL) has focused on fully-cooperative learning in domains such as Dec-POMDP's (Oliehoek & Amato, 2016) or zero-sum games like Starcraft and Go Silver et al. (2017); Vinyals et al. (2019), these settings only represent a fraction of potential real-world multi-agent environments. General-sum games, which can be neither fully-cooperative nor fully-competitive, describe many domains such as agent-based modeling, social dilemmas, and systems of interacting self-interested agents like self-driving cars.

Even simple social dilemmas commonly present unique challenges that are not present in single-agent learning (Foerster et al., 2018a). For example, in the *IPD* (Axelrod & Hamilton, 1981; Harper et al., 2017), learning agents that treat their opponents as static parts of the environment typically converge on unconditional mutual defection, which is the globally worst outcome.

To avoid such catastrophic outcomes, Foerster et al. (2018a) introduce LOLA, which takes into account the opponents' learning step in order to shape their policy. In the self-play setting, LOLA was one of the first methods to discover the reciprocating tit-for-tat (TFT) strategy in the IPD.

However, LOLA and related algorithms, such as Stable Opponent Shaping (Letcher et al., 2019b, SOS) and Meta Multi-Agent Policy Gradient (Kim et al., 2021, Meta-MAPG), assume that the opponent is a naive learning (NL) agent, which is often incorrect, e.g. in self-play. Furthermore, to shape their opponents, these methods use second-order derivatives, which are typically high-variance, making learning unstable (Foerster et al., 2018a). Lastly, they are also *myopic* – they only shape the opponent's next few learning steps, not their long-term development.

To resolve *all* of these issues, we introduce **M**odel**-F**ree **O**pponent **S**haping (M-FOS). M-FOS is a general meta-learning algorithm that learns over multiple opponent-learning steps *without requiring a model of its opponent's underlying learning algorithm*.

The core of M-FOS is a *meta-game* in which each meta-step is an episode of the underlying ("inner") game. The meta-state consists of the inner policies, and the meta-policy produces a new inner policy to be used in the next episode. M-FOS then uses generic *model-free optimisation methods*, rather than approaches that require higher-order derivatives, to learn meta-policies that accomplish long-horizon opponent shaping. Furthermore, training M-FOS in *meta-self-play* allows mutual opponent shaping without causing the kind of infinite regress typically caused by ever higher-order learning awareness (Foerster et al., 2018a).

However, since M-FOS is naively model-free, the meta-self-play setting reduces to independent learning, which is highly initialisation-dependent and unstable in general-sum settings. To mitigate this, we introduce a training schedule inspired by Cognitive Hierarchies (CH) (Camerer et al., 2003). With this schedule, M-FOS learns to reciprocate with itself in the meta-game, even achieving higher scores than LOLA in self-play.

For low-dimensional games, M-FOS directly learns policy updates by taking policies as input and outputting the next policy as an action. However, directly inputting and outputting policies does not scale to higher-dimensional games. We introduce a variant of M-FOS that takes past trajectories as inputs to meta-learn across its opponent's learning steps. We then demonstrate that, even in social dilemmas with temporally-extended transition dynamics, M-FOS still manages to shape naive learners and find mutually beneficial solutions in meta-self-play.

In Section 6, we show that M-FOS can exploit naive learners much better than a set of widely used general-sum learning algorithms (Foerster et al., 2018a; Kim et al., 2021). In the IPD, M-FOS discovers a famous strategy known as ZD extortion (Press & Dyson, 2012) when playing against NL agents. Notably, unlike other algorithms, it does so *without* access to the opponent's underlying learning algorithm. M-FOS even learns to exploit other general-sum algorithms, such as LOLA.

2 Related Work

Opponent Shaping: Several methods recognise that their current actions influence the future policies of learning opponents and take advantage of this to "shape" an opponent's policy to desirable values. Most of these works assume white-box access to an opponent's learning algorithm and reward in order to take higher-order derivatives through an opponent's update (Foerster et al., 2018a; Letcher et al., 2019a; Kim et al., 2021; Willi et al., 2022). Such updates are also myopic since anticipating many steps is intractable. In self-play, these methods inconsistently assume that their opponent is a naive learner. M-FOS does not assume white-box access to an opponent's underlying learning algorithm or reward, does not require higher-order derivatives (which are often high-variance), can shape opponents across a large number of updates, and is consistent in self-play.

Opponent Modeling: Much work in MARL has focused on the idea of *opponent modeling* in which an agent attempts to model some aspect of the policy of other agents in the environment. This includes explicitly modeling opponent policies (Mealing & Shapiro, 2017), modeling opponent intentions (Raileanu et al., 2018), classifying opponent strategies (Weber & Mateas, 2009; Synnaeve & Bessière, 2011), and modeling an opponent's nested beliefs (Wen et al., 2019). LILI (Xie et al., 2020) models an opponent's high-level latent strategy from local observations with a latent dynamics model rather than explicitly modeling the opponent's policy. However, these methods are not capable of actively *shaping* their opponents' learning dynamics, thus they do not address the same issues as M-FOS.

Multi-Agent Meta-Learning: M-FOS is a form of multi-agent meta-learning where the meta-policy is parameterized by a neural network. Existing multi-agent meta-learning methods, such as Meta-Policy Gradient (Meta-PG) (Al-Shedivat et al., 2018), Meta-MAPG (Kim et al., 2021), and Learning to Exploit (L2E) (Wu et al., 2021) instead parameterize the meta-policy using a method similar to that of Model-Agnostic Meta-Learning (MAML) (Finn et al., 2017), in which they learn initial parameters and meta-learn across their own gradient updates. While this type of meta-learning can adapt to any task at test time in single-agent settings (Xiong et al., 2021), in multi-agent settings, the calculated gradient may not correspond to a direction of improvement as the updates of other agents change the underlying dynamics. Rather than being restricted to a gradient update within the episode, M-FOS allows for arbitrary meta-policies that can carry out long horizon opponent shaping.

Algorithm 1 General M-FOS

Initialize M-FOS parameters θ.
while true **do**
  Initialize agents' parameters $\phi_0^i, \boldsymbol{\phi}_0^{-i}$.
  for $t = 0$ **to** T **do**
    Reset environment
    Gather trajectories $\tau_{\boldsymbol{\phi}}$ given $\phi_t^i, \boldsymbol{\phi}_t^{-i}$
    Update $\boldsymbol{\phi}_{t+1}^{-i}$ according to respective learning algorithms
    Update ϕ_{t+1}^i according to meta-policy π_θ
  end for
  Update θ
end while

3 BACKGROUND

A **partially observable stochastic game** (Kuhn, 1953, POSG) consists of a tuple $\mathcal{M}_n = \langle \boldsymbol{\mathcal{I}}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \boldsymbol{\Omega}, \mathcal{O}, \mathcal{P}, \boldsymbol{\mathcal{R}}, \gamma \rangle$, where $\boldsymbol{\mathcal{I}} = \{1, \ldots, n\}$ denotes a set of n agents, $\mathcal{S}$ denotes the state space, $\boldsymbol{\mathcal{A}} = \times_{i \in \mathcal{I}} \mathcal{A}^i$ represents the joint action space, $\boldsymbol{\Omega} = \times_{i \in \mathcal{I}} \boldsymbol{\Omega}^i$ the joint observation space, $\mathcal{P} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \mapsto \mathcal{S}$ denotes the transition probability function, $\mathcal{O} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\Omega} \to [0, 1]$ is the observation function, $\boldsymbol{\mathcal{R}} = \times_{i \in \mathcal{I}} \mathcal{R}^i$ represents the set of reward functions of all agents, and $\gamma \in [0, 1)$ denotes the discount factor. At each timestep t, every agent samples an action from its stochastic policy, $a_t^i \sim \pi^i \left(\cdot \mid o_t^i, \phi^i \right)$, where the joint actions at timestep t are $\boldsymbol{a_t} = \left\{ a_t^i, \boldsymbol{a_t}^{-i} \right\}$ and $-\boldsymbol{i}$ stands for all agents except i. The policy is parameterized by ϕ^i. Given the joint actions and the current state, each agent receives their respective reward $r_t^i = \mathcal{R}^i \left(s_t, \boldsymbol{a_t} \right)$. Finally, a new state is sampled $s_{t+1} \sim \mathcal{P} \left(\cdot \mid s_t, \boldsymbol{a_t} \right)$.

Popular special cases of POSGs are *fully observable* stochastic games where all agents observe the full state at each time step or single-player, i.e. $\boldsymbol{\mathcal{I}} = \{1\}$, partially observable Markov decision processes (POMDPs), and MDPs, where the single player observes the full state at each time step.

4 MODEL-FREE OPPONENT SHAPING

Typically opponent-shaping methods are based on MAML-like approaches Foerster et al. (2018a); Letcher et al. (2019b); Kim et al. (2021) and use higher-order derivatives to directly shape the opponents' parameter update, which requires white-box access to their differentiable learning algorithm. Furthermore, opponent shaping typically creates a conceptual problem: To shape an opponent, an algorithm needs to specify the learning behaviour of other agents in the environment, e.g. by treating them as *naive learners*, as is done in LOLA (Foerster et al., 2018a). This leads to a fundamental inconsistency in self-play when two of these agents are training together. Even though they are both *opponent shaping* they treat each other as *naive learners*, which can lead to undesired outcomes (Letcher et al., 2019a). Lastly, most opponent-shaping methods only shape the next learning steps instead of considering longer horizons.

Opponent shaping can be formulated as a meta-game, in which the meta-state consists of the policies of all agents, a meta-step is an inner episode, the reward is the inner return, and the meta-action is choosing the next inner policy, where "inner" refers to the underlying game. The *key insight* underlying Model-Free Opponent Shaping (M-FOS) is that we can resolve all of the issues above by directly training meta-policies using model-free optimisation methods that are appropriate for sequential settings, rather than relying on MAML-like approaches.

We formally construct the meta-game as a POMDP $\langle \bar{\mathcal{S}}, \bar{\mathcal{A}}, \Omega, \bar{\mathcal{O}}, \bar{\mathcal{P}}, \bar{\mathcal{R}}, \bar{\gamma} \rangle$ over an underlying POSG $\mathcal{M}_n$. The meta-game is partially observable because we do not assume full access to the opponents' parameters. The M-FOS meta-agent controls agent $i \in \boldsymbol{\mathcal{I}}$ in the underlying POSG $\mathcal{M}_n$. The state space $\bar{S}$ of the meta-game consists of the policy parameters of the agents in the underlying POSG, $\bar{s}_t = (\phi_{t-1}^i, \boldsymbol{\phi}_{t-1}^{-i}) \in \bar{\mathcal{S}}$. The meta-agent's action space consists of agent i's policy, for example outputting a conditioning vector or setting agent i's policy parameters directly, $\bar{a}_t = \phi_t^i \sim \pi_\theta(\cdot \mid \bar{o}_t)$. Here the meta-policy is parameterized by θ. The meta-agent receives observation $o_t \in \Omega$ with

probability $\bar{\mathcal{O}}(\bar{o}_t \mid \bar{s}_t, \bar{a}_t)$. After each meta-episode, the scalar reward is $\bar{r}_t = \sum_{k=0}^{K} r_k^i(\phi_t^i, \boldsymbol{\phi}_t^{-i})$, where K is the length of the inner episode (i.e. the *reward* in the meta-game at each step is the inner *return*). Finally, a new meta-state is sampled from a stochastic transition probability function, $\bar{s}_{t+1} \sim \bar{P}(\cdot \mid \bar{s}_t, \bar{a}_t)$. $\bar{P}(\bar{s}_t, \bar{a}_t)$ is stochastic since, in general, the update function for any agent can be stochastic, $\phi_{t+1}^j \sim h(\cdot \mid \phi_t^j)$. For example, when agent j updates their parameters with policy gradients. Consequently, the trajectory is denoted as $\bar{\tau}_\theta := (\bar{o}_0, \bar{a}_0, \bar{r}_0, \ldots, \bar{r}_T)$, where T is the length of the meta-episode. We train the meta-policy to maximise the expected return per meta-episode $J = \sum_{t=0}^{T} \bar{r}_t^i(\phi_t^i, \boldsymbol{\phi}_t^{-i})$. Crucially, rather than relying on higher-order derivatives, M-FOS uses *model-free optimisation* methods to directly train a meta-policy. In the Section 6 we show that PPO (Schulman et al., 2017; Barhate, 2021) and Genetic Algorithms (Such et al., 2017) work well in this general meta-learning framework.

4.1 M-FOS Self-Play

By doing model-free optimisation in the meta-game, we no longer require higher-order derivatives and also can learn strategies that engage in long-horizon opponent shaping. Next, we also address the issue of symmetry and consistency by introducing *meta-self-play*.

When using MAML-like approaches for opponent shaping, attempts of consistent self-play lead to *infinite recursions*, since each agent differentiates through the learning step of the other agent and so on. In contrast, since M-FOS is entirely model-free, *meta-self-play* between two M-FOS agents simply corresponds to learning in a general-sum game, where model-free methods can be applied without causing infinite regress.

One challenge is that independent learning in general-sum settings is highly initialisation dependent and unstable, which is undesirable for a principled method. To overcome this, we take inspiration from the Cognitive Hierarchies framework (Camerer et al., 2003) and introduce an *implicit hierarchy*. This is implemented via a parameter λ that corresponds to the probability of an M-FOS agent being paired with a naive learner rather than another M-FOS agent. By setting $\lambda = 1$ at the beginning of training, we ground the training to an approximate best-response to NL, while annealing it to $\lambda = 0$ allows us to transition to self-play over the course of training gradually. Suppose λ is annealed slowly enough, such that the M-FOS agents are always playing near optimally for the given distribution. In that case, this process mirrors an infinite-depth cognitive hierarchy, overcoming the stability issues of multi-agent learning.

5 Experimental Setup

5.1 Environments

IPD: The iterated prisoner's dilemma is one of the most widely-studied and important general-sum games, with applications in evolutionary biology, economics, politics, sociology, and other fields (Rapoport et al., 1965). In the iterated prisoner's dilemma, agents can choose to cooperate (C) or defect (D) against each other, with the payouts of the result being presented in Table 5a. The game is played repeatedly, with players able to observe their opponent's past decisions. Axelrod (Axelrod & Hamilton, 1981) famously held an IPD tournament where a strategy known as TFT, in which a player copies the other player's last move, was popularized.

Despite decades of previous study of the IPD, Press & Dyson (2012) made a surprising mathematical discovery that dramatically changed our understanding of the game: There exist fixed policies, called ZD extortion strategies, that dominate any learning opponent. More specifically, ZD extortion enforces a linear relationship between the two agents' rewards that *disproportionately* benefits the extortioner (see Figure 1a). However, it is still in a learning agent's best interest to cooperate against extortion despite the fact that it benefits the extortioner more.

Iterated Matching Pennies: Iterated Matching Pennies (IMP) is an iterated matrix game like the IPD but is zero-sum. Agents can choose "Heads" or "Tails" and get payouts according to Table 5b.

Chicken Game: The Chicken Game is a stochastic matrix game. Agents can either Swerve (C) or head Straight (D). While agents can gain a small reward by heading straight against a swerving opponent, they incur a large negative cost if they both head straight. It is often used in political

science and economics to describe brinksmanship scenarios in which there is a threat of mutually assured destruction Rapoport & Chammah (1966).

Coin Game: Coin Game is a multi-agent grid-world environment that simulates social dilemmas like the IPD but with high dimensional dynamic states first proposed by Lerer & Peysakhovich (2017). We provide more details in Figure 6.

5.2 Baseline Comparisons

Naive Learning (NL): Naive learners assume that other agents are part of the environment and are static between episodes. Thus, between each episode, naive learners perform the following update 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 reinforcement learning, this is often approximated with a sample-based approach. In our experiments, in the Coin Game, the NL uses PPO, Schulman et al. (2017) which modifies this by clipping the update. In matrix games, we can directly perform gradient ascent without sampling because the exact value $\mathcal{R}^i$ is differentiable.

Learning with Opponent Learning Awareness (LOLA): LOLA assumes that other agents are naive learners and perform the gradient step performed above. LOLA takes a gradient through the opponent's update function to shape the opponent.

$$\begin{aligned} \phi^i_{t+1} &= \phi^i_t + \alpha^i \nabla_{\phi^i_t} \mathcal{R}^i(\phi^i_t, \phi^{-i}_t + \Delta\phi^{-i}_t) \\ \Delta\phi^{-i}_t &= \alpha^{-i} \nabla_{\phi^i_t} \mathcal{R}^{-i}(\phi^i_t, \phi^{-i}_t) \end{aligned} \tag{2}$$

Multiagent Model-Agnostic Meta-Learning: We introduce a new baseline, Multiagent MAML (M-MAML), which is inspired by Meta-Multiagent Policy Gradient (Kim et al., 2021, Meta-MAPG). Meta-MAPG and M-MAML operate in a similar setting to M-FOS in that they meta-learn over multiple opponent learning updates. However, instead of learning an update function, they learn initial parameters. They then meta-learn over their own gradient updates (much like MAML Finn et al. (2017)) as well as the gradient updates of their opponents. Meta-MAPG and M-MAML optimize the following:

$$\begin{aligned} &\max_{\phi^i_0} \mathbb{E}_{p(\phi^{-i}_0)}[\sum_{t=0}^{t=T} \mathcal{R}^i(\phi^i_t, \phi^{-i}_t)], \\ &\phi^i_{t+1} = \phi^i_t + \alpha^i \nabla_{\phi^i_t} \mathcal{R}^i(\phi^i_t, \phi^{-i}_t) \\ &\phi^{-i}_{t+1} = \phi^{-i}_t + \alpha^{-i} \nabla_{\phi^{-i}_t} \mathcal{R}^{-i}(\phi^i_t, \phi^{-i}_t) \end{aligned} \tag{3}$$

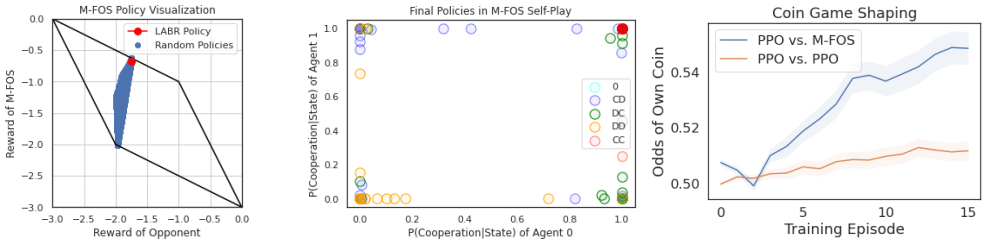
I.e., the methods only optimize initial policy parameters, assuming that all agents are naive learners.

Meta-MAPG expands the objective into multiple learning terms to perform policy-gradient updates. However, we do not directly compare to Meta-MAPG because it only scales to $T = 7$ meta-steps in the IPD, not $T = 100$. Instead, we use the exact value function and exact gradients allowing our baseline (M-MAML) to scale to meta-episodes consisting of 100 inner episodes.

5.3 M-FOS Implementation Details

Matrix Games: In the matrix game environments we allow M-FOS to observe the full state, which is the concatenation of the policies played last timestep $o_t = s_t = (\phi^i_{t-1}, \phi^{-i}_{t-1})$. Because all of our evaluated opponents (including M-FOS itself) only make updates according to the current state, this turns the induced POMDP into an MDP. Because the inner policy can be fully expressed with very few parameters, we can directly output the parameters, turning the MDP into a basic continuous control problem. Because of this, we model the M-FOS meta-agent as a simple feed-forward neural network parameterized by θ that takes in the state and outputs a distribution over the next policy.

$$\begin{aligned} &\max_{\theta} \mathbb{E}_{p(\phi^{-i}_0, \phi^i_0)}[\sum_{t=0}^{t=T} \mathcal{R}^i(\phi^i_t, \phi^{-i}_t)], \\ &\phi^i_{t+1} \sim \pi_\theta(\cdot \mid \phi^i_t, \phi^{-i}_t) \\ &\phi^{-i}_{t+1} = f(\phi^i_t, \phi^{-i}_t) \end{aligned} \tag{4}$$



(a) (b) (c)

Figure 1: (a) Visualisation of M-FOS v. Look-Ahead Best Response in the IPD. Note that the payoff between the two agents is near-linear and favors the M-FOS agent, indicating ZD extortion. (b) Visualisation of 32 Final Episode Policies in M-FOS v. M-FOS in the IPD (c) Probability of the PPO agent picking up its own coin across the inner episodes. Note that it is shaped into picking up more of its own coins against the M-FOS agent.

We optimize the meta-policy using both Genetic Algorithms Such et al. (2017), and PPO Schulman et al. (2017); Barhate (2021), and report the best of both. A detailed breakdown of the performance of each can be found in the Appendix B.

M-FOS in Coin Game: Here, M-FOS does not directly observe the opponent's policy parameters but only the effects of their past actions. The opponent is parameterized by a convolutional neural network and, as a naive learner, is trained using PPO. M-FOS's inner policy is parameterized by a convolutional recurrent neural network that takes in an observation as input along with a conditioning vector from the meta-policy. We require the inner policy to be recurrent to respond to and shape the opponent's policy. The hidden state of the recurrent neural network is reset each episode. M-FOS's meta-policy is parameterized by a convolutional recurrent neural network that processes the batch of trajectories from the last episode and outputs a conditioning vector, used in the next episode. Using PPO, the inner policy and the meta-policy parameters are trained end-to-end to maximise the expected discounted meta-return.

6 Results

6.1 Matrix Games

IPD: In a round-robin tournament in which algorithms train against each other in a head-to-head matchup, M-FOS vastly outperforms all other learning methods in the IPD in Table 1. Notably, it is the only algorithm to achieve scores better than mutual cooperation (-1), and it does so against all opponents, excluding itself. Similarly, it is the only algorithm for which one of its opponents performs worse than mutual defection (-2), and it does so against both naive learners and LOLA.

Table 1: Head-to-head rewards of each learning algorithm in the Iterated Prisoner's Dilemma.

	M-FOS	NL	LOLA	M-MAML
M-FOS	**-1.01**	**-0.51**	**-0.73**	**-0.67**
NL	-2.14	-1.98	-1.52	-1.28
LOLA	-2.09	-1.30	-1.09	-1.04
M-MAML	-1.86	-1.25	-1.15	-1.17

M-FOS v. Naive Learner: Against an NL agent, M-FOS gets an average score of -0.51, while the NL agent gets an average score of -2.14. This is a far more advantageous result than LOLA achieves (-1.30/-1.52), even though LOLA has a perfect learning model of its opponent and can take the derivative through its update step and the environment. In Figure 2 we show that this is because LOLA is a myopic one-step learner whereas M-FOS considers the discounted returns far in the future.

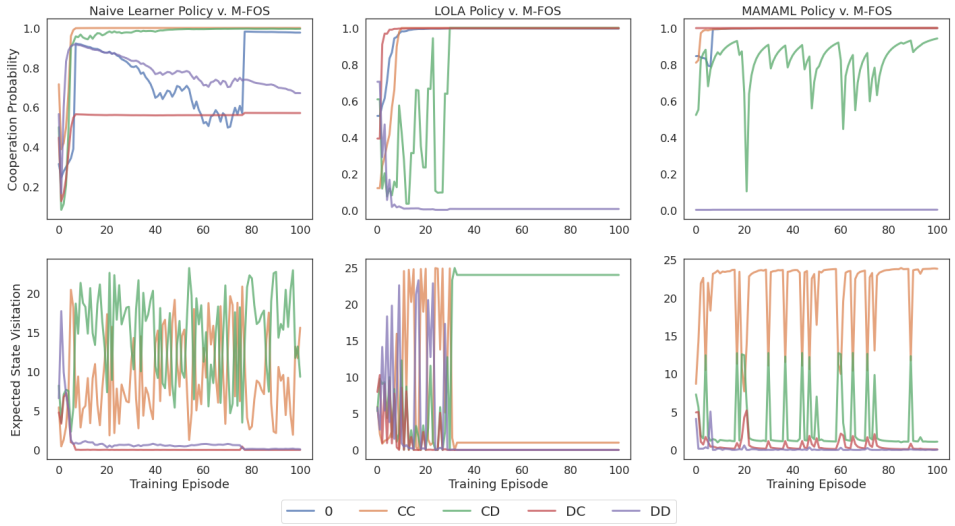


Figure 2: Visualisations of a run of a meta-episode of each learner against M-FOS. Notice how the opponents' policies are shaped into cooperating, resulting in state visitations that are beneficial to the M-FOS agent.

Also, note that the NL agent achieves a total score lower than -2. This is a lower score than ZD extortion can theoretically make its opponent achieve since blind defection at worst achieves a score of -2. Figure 3 shows how M-FOS achieves this.

M-FOS v. Look-Ahead Best Response: To demonstrate the above point, we train M-FOS against a variant of a naive learner that can observe its opponent's next policy and then plays the best response to it (which is calculated by performing a thousand steps of gradient ascent). Despite the game being symmetric, M-FOS extorts this Look-Ahead Best Response (LABR) agent, achieving an average score of -0.71. Figure 1a shows that the policy M-FOS outputs approximates ZD extortion. To the best of our knowledge, M-FOS is the first learning algorithm to discover ZD extortion.

M-FOS v. LOLA: Foerster et al. (2018a) write that 2nd-order LOLA, which is an agent that takes the derivative through the opponent's LOLA update, does not achieve any incremental gains against an opposing LOLA agent. In other words, a LOLA agent achieves a better score against another LOLA agent than a 2nd-order LOLA agent would, implying that it is difficult to exploit LOLA. However, M-FOS manages to find a dominating strategy against LOLA (-0.73 / -2.09). To the best of our knowledge, M-FOS is the first learning algorithm to exploit the LOLA update.

M-FOS v. M-MAML: M-MAML seems to have generally learned to initialize with values close to TFT (see Figure 5). This initialisation allows it to achieve favorable results against all algorithms except M-FOS (-0.67 / -1.86), which learns to exploit it in Figure 2.

M-FOS v. M-FOS: We arrive at a cooperative score when M-FOS is trained against other M-FOS agents using the meta-self-play training scheme from above. When viewing the final policies played against each other, we observe that M-FOS has largely arrived at TFT, as seen in Figure 1b. To the best of our knowledge, M-FOS is the first learning algorithm to arrive at TFT in the IPD against itself without using higher-order derivatives, access to the opponent's rewards, or specific hand-coding of TFT-like behaviour.

Table 2: Head-to-head results of each learning algorithm in Iterated Matching Pennies.

	M-FOS	NL	LOLA	M-MAML
M-FOS	**0.0**	**0.20**	**0.19**	**0.22**
NL	-0.20	0.0	-0.02	-0.01
LOLA	-0.19	0.02	0.0	0.02
M-MAML	-0.22	0.01	-0.02	0.0

IMP: In IMP, M-FOS once again outperforms other baseline methods in Table 2. In particular, by examining how M-FOS exploits a naive learner compared to how LOLA does so, we observe that LOLA is myopic compared to M-FOS. In Figure 4, LOLA gradually approaches the nash equilibrium against a naive learner in order to avoid being exploited by its opponent. In contrast, M-FOS cyclically shapes the naive learner's policy to continuously exploit it while staying one step ahead.

Table 3: Head-to-head results of each learning algorithm in the Chicken Game. The results of an M-FOS meta-policy that learns an initial policy is in parentheses.

	M-FOS	NL	LOLA	M-MAML
M-FOS	**-0.01**	**0.97**	-0.94[**0.5**]	**0.86**
NL	-1.03	-0.0	-0.97	-0.27
LOLA	**0.87**[-1.5]	0.94	-85.96	0.40
M-MAML	-1.08	0.27	**-0.42**	-0.15

Chicken Game: M-FOS performs well against all baselines in the Chicken Game in Table 3 but achieves a lower score against LOLA. Interestingly, LOLA behaves arrogantly in the Chicken Game – it always believes it can shape its opponent by heading straight. While this works against most learning opponents, it leads to catastrophic results in self-play (-85.96).

Because M-MAML selects its initial policy, it can shape LOLA from the first time step, preventing LOLA from immediately heading straight after its first update. M-FOS, in contrast, is by default forced to a random initialisation. However, if we allow M-FOS also to learn an initial policy, it achieves a much higher score against LOLA (0.5), far outperforming M-MAML (-0.42).

6.2 Coin Game

Table 4: Head-to-head results of M-FOS and PPO in the Coin Game.

	M-FOS	PPO
M-FOS	**20.56**	**44.26**
PPO	-24.62	4.25

Prior work (Yu et al., 2021) has shown that LOLA-DiCE (Foerster et al., 2018b) and Meta-MAPG (Kim et al., 2021) do not achieve significant results in a *simplified* version of coin game with a fully cooperative reward. Because of this, we do not compare to these baselines. We also observe that M-FOS outperforms PPO in head-to-head training in Figure 1c while still achieving good performance in self-play. Meanwhile, PPO agents, when trained together, pick up each other's coins indiscriminately, leading to 0 expected reward.

7 Conclusion & Future Work

In this paper, we presented Model-Free Opponent Shaping (M-FOS) as a simple model-free alternative to popular MAML-like opponent shaping methods, such as LOLA and MMAPG. Although M-FOS does not use higher-order derivatives and does not have white-box access to its opponent's learning model, it vastly outperforms all tested baselines across several matrix games.

More specifically, in the IPD, M-FOS achieves several notable results. First, to the best of our knowledge, it is the first learning algorithm to discover ZD extortion, the first learning algorithm that exploits LOLA, and the first learning algorithm to achieve cooperation in self-play without using higher-order derivatives or inconsistent models. Furthermore, it achieves a score higher than mutual cooperation against all tested opponents, while none of the baselines could do so against any single opponent. We also show that M-FOS can scale to more complex, high-dimensional games and achieve similar results.

In the future, we could generalize M-FOS beyond social dilemmas. For example, M-FOS could shape another learning agent over a cheap talk channel. Such a method could then be used to shape models trained on real-world data in an adversarial manner.

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 *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Sk2u1g-0-.

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

Nikhil Barhate. Minimal pytorch implementation of proximal policy optimization. https://github.com/nikhilbarhate99/PPO-PyTorch, 2021.

Colin Camerer, Teck Ho, and Juin-Kuan Chong. A cognitive hierarchy theory of one-shot games and experimental analysis. *SSRN Electronic Journal*, 09 2003. doi: 10.2139/ssrn.411061.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, 2017.

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, 2018a.

Jakob N. Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric P. Xing, and Shimon Whiteson. Dice: The infinitely differentiable monte carlo estimator. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1524–1533. PMLR, 2018b. URL http://proceedings.mlr.press/v80/foerster18a.html.

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.

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 *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5541–5550, 2021.

H. W. Kuhn. *Extensive games and the problem of information*. Princeton University Press, Princeton, NJ, 1953.

Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *CoRR*, abs/1707.01068, 2017. URL http://arxiv.org/abs/1707.01068.

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.

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.

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.

Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer International Publishing, 2016. ISBN 978-3-319-28927-4. doi: 10.1007/978-3-319-28929-8. URL https://www.springer.com/gp/book/9783319289274.

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.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4254–4263. PMLR, 2018. URL http://proceedings.mlr.press/v80/raileanu18a.html.

Anatol Rapoport and Albert M. Chammah. The game of chicken. *American Behavioral Scientist*, 10 (3):10–28, 1966. doi: 10.1177/000276426601000303. URL https://doi.org/10.1177/000276426601000303.

Anatol Rapoport, Albert M Chammah, and Carol J Orwant. *Prisoner's dilemma: A study in conflict and cooperation*, volume 165. University of Michigan press, 1965.

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

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nat.*, 550(7676):354–359, 2017.

Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint arXiv:1712.06567, 2017.

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.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354, 2019.

Ben George Weber and Michael Mateas. A data mining approach to strategy prediction. In *Proceedings of the 2009 IEEE Symposium on Computational Intelligence and Games*, pp. 140–147, 2009.

Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *7th International Conference on Learning Representations*, 2019.

Timon Willi, Johannes Treutlein, Alistair Letcher, and Jakob Foerster. COLA: consistent learning with opponent-learning awareness. *CoRR*, abs/2203.04098, 2022. doi: 10.48550/arXiv.2203.04098. URL https://doi.org/10.48550/arXiv.2203.04098.

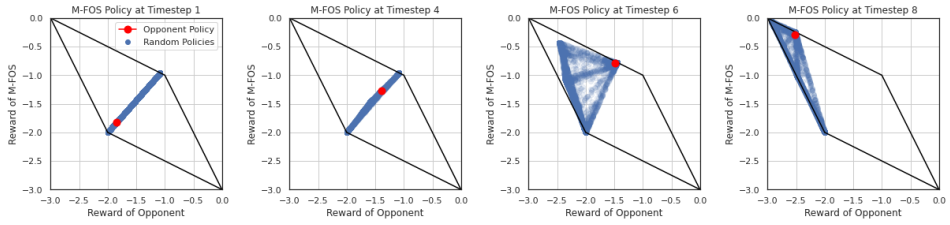
Zhe Wu, Kai Li, Enmin Zhao, Hang Xu, Meng Zhang, Haobo Fu, Bo An, and Junliang Xing. L2e: Learning to exploit your opponent, 2021.

Annie Xie, Dylan P. Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. Learning latent representations to influence multi-agent interaction. In *4th Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 575–588, 2020.

Zheng Xiong, Luisa M. Zintgraf, Jacob Beck, Risto Vuorio, and Shimon Whiteson. On the practical consistency of meta-reinforcement learning algorithms. *CoRR*, abs/2112.00478, 2021. URL https://arxiv.org/abs/2112.00478.

Xiaopeng Yu, Jiechuan Jiang, Haobin Jiang, and Zongqing Lu. Model-based opponent modeling. arXiv preprint arXiv:2108.01843, 2021.

A ADDITIONAL PLOTS



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

Figure 3: Visualisations of M-FOS shaping a naive learner. The area denoted by the black lines represents the episode's possible rewards. The blue points represent the possible payoffs of a naive learner against the M-FOS policy at that timestep. (a)-(b) M-FOS begins by playing TFT until the opponent is sufficiently cooperative. (c)-(d) M-FOS then repeatedly switches between an extortion-like policy (c) and a defecting policy (d), making the NL oscillate.

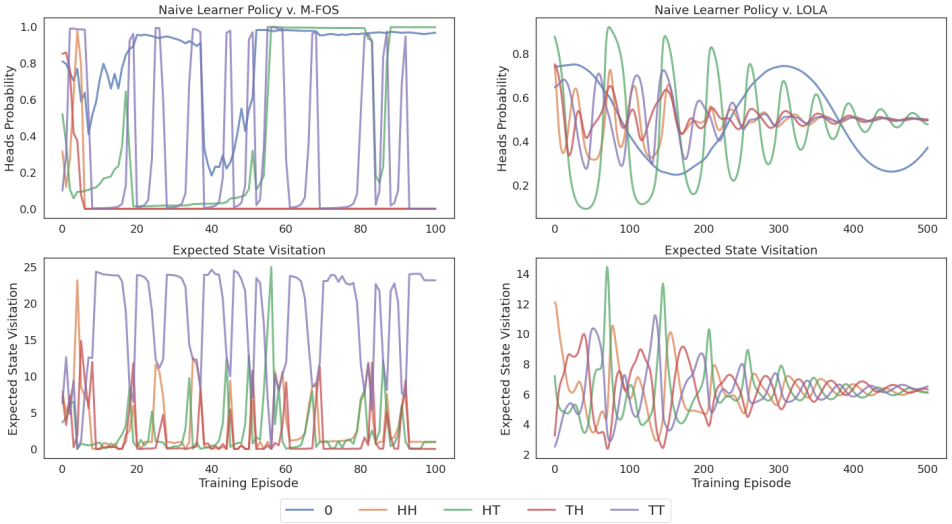


Figure 4: Visualisation of M-FOS's long-term shaping LOLA's and myopic strategy in the Iterated Matching Pennies environment. Note how LOLA converges to the nash equilibriuum, resulting in zero reward for both agents, while M-FOS continually drags the naive learner's policy to exploitable states.

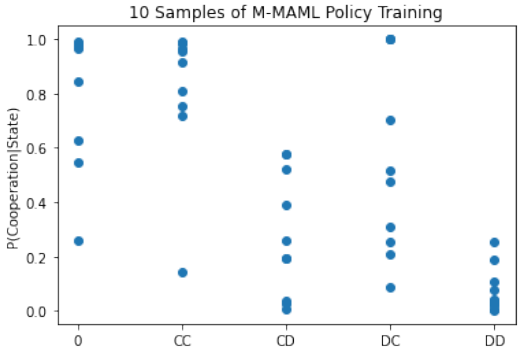


Figure 5: The distribution of probabilities in each state after training 10 different instances of M-MAML.

B Detailed Results

Each experiment is run 10 times. The inner batch size of each experiment for the matrix games is 4096.

Table 5: Payoff Matrix for (a) the Prisoner's Dilemma, (b) Matching Pennies, and (c) Chicken Game

(a)

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

(b)

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

(c)

	C	D
C	(0, 0)	(-1, +1)
D	(+1, -1)	(-100, -100)

Table 6: Head-to-head results of each learning algorithm in IPD, results reported for M-FOS PPO.

	M-FOS	NL	LOLA	M-MAML
M-FOS	-1.01	-0.51	-1.03	-0.84
NL	-2.14	-1.98	-1.52	-1.28
LOLA	-1.02	-1.30	-1.09	-1.04
M-MAML	-1.52	-1.25	-1.15	-1.17

Table 7: Head-to-head results of each learning algorithm in IPD, results reported for M-FOS GA.

	M-FOS	NL	LOLA	M-MAML
M-FOS	–	-0.745	-0.73	-0.67
NL	-1.69	-1.98	-1.52	-1.28
LOLA	-2.09	-1.30	-1.09	-1.04
M-MAML	-1.86	-1.25	-1.15	-1.17

Table 8: Head-to-head results of each learning algorithm in IMP, results reported for M-FOS PPO.

	M-FOS	NL	LOLA	M-MAML
M-FOS	0.0	0.20	0.19	0.22
NL	-0.20	0.0	-0.02	-0.01
LOLA	-0.19	0.02	0.0	0.02
M-MAML	-0.22	0.01	-0.02	0.0

Table 9: Head-to-head results of each learning algorithm in IMP, results reported for M-FOS GA.

	M-FOS	NL	LOLA	M-MAML
M-FOS	–	0.13	0.10	0.17
NL	-0.13	0.0	-0.02	-0.01
LOLA	-0.10	0.02	0.0	0.02
M-MAML	-0.17	0.01	-0.02	0.0

Table 10: Head-to-head results of each learning algorithm in the Chicken Game. The results of an M-FOS meta-policy that learns an initial policy is in parantheses. Results reported for M-FOS PPO.

	M-FOS	NL	LOLA	M-MAML
M-FOS	-0.01	0.97	-0.94[0.5]	0.85
NL	-1.03	-0.0	-0.97	-0.27
LOLA	0.87[-1.5]	0.94	-85.96	0.40
M-MAML	-1.11	0.27	-0.42	-0.15

C Hyperparameter Details

We report our hyperparameter values that we used for each of the methods in our experiments:

Table 11: Head-to-head results of each learning algorithm in the Chicken Game. The results of an M-FOS meta-policy that learns an initial policy is in parantheses. Results reported for M-FOS GA.

	M-FOS	NL	LOLA	M-MAML
M-FOS	–	0.97	-0.94[0.5]	0.86
NL	-1.03	-0.0	-0.97	-0.27
LOLA	0.91[-1.5]	0.94	-85.96	0.40
M-MAML	-1.08	0.27	-0.42	-0.15

C.1 M-FOS

Hyperparameter	Value
Number of Actor Hidden Layers	1
Size of Actor Hidden Layers	[256]
Number of Critic Hidden Layers	1
Size of Critic Hidden Layers	[256]
Length of Meta-Episode T	100
Batch Size B	4096
Adam Step Size	0.0002
Number of Epochs	4
Outer Discount Factor γ	0.99
PPO Clipping ϵ	0.2
Entropy Coefficient	0.01

Table 12: PPO for IPD, IMP, and Chicken Game

Hyperparameter	Value
Number of Hidden Layers	1
Size of Hidden Layers	[256]
Number of Species N	2048
Batch Size B	128
Length of Meta-Episode T	100
Noise Std Dev σ	2.0
Number of Elites E	1

Table 13: Genetic Algorithm for IPD, IMP, and Chicken Game

Hyperparameter	Value
Number of Conv Layers	2
Output Channels of Conv Layers	[16, 16]
Kernel Sizes of Conv Layers	[[3, 3], [3, 3]]
Strides of Conv Layers	[1, 1]
Number of Linear Layers	1
Size of Linear Layer	[16]
Number of GRUs	1
Size of GRUs	[16]
Length of Meta-Episode T	16
Length of Inner Episode	16
Batch Size B	512
Adam Step Size	0.0002
Number of Epochs	16
Outer Discount Factor γ	0.99
PPO Clipping ϵ	0.2
Entropy Coefficient	0.01

Table 14: PPO For Coin Game. The Actor, Critic, and Meta-Policy have the same network architecture but do not share weights.

C.2 Environments

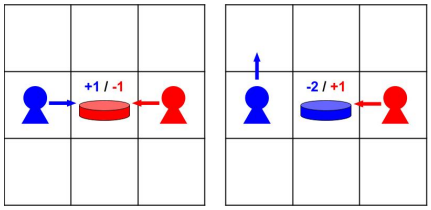


Figure 6: Illustration of Coin Game. The game consists of two players, labeled red and blue, who are tasked with picking up coins, also labeled red and blue, in a 3x3 grid. If a player picks up any coin by moving into the same position as the coin, they receive a reward of $+1$. However, if they pick up a coin of the other player's color, the other player receives a reward of -2. Thus, if both agents play greedily and pick up every coin, the expected reward for both agents is 0.

Hyperparameter	Value
Inner Gamma γ	0.96
Learning Rate α	1
M-MAML Adam Learning Rate	0.05

Table 15: Hyperparameters for IPD Environment

Hyperparameter	Value
Inner Gamma γ	0.96
Learning Rate α	0.1
M-MAML Adam Learning Rate	0.05

Table 16: Hyperparameters for IMP Environment

Hyperparameter	Value
Learning Rate α	1
M-MAML Adam Learning Rate	0.05

Table 17: Hyperparameters for Chicken Environment

Hyperparameter	Value
Number of Conv Layers	2
Output Channels of Conv Layers	[16, 16]
Kernel Sizes of Conv Layers	[[3, 3], [3, 3]]
Strides of Conv Layers	[1, 1]
Number of Linear Layers	1
Size of Linear Layer	[16]
Adam Step Size	0.005
Number of Epochs	80
PPO Clipping ϵ	0.2
Entropy Coefficient	0.01
Discount Factor γ	0.96
Length of Inner Episode	16

Table 18: Hyperparameters for Coin Game Environment and Naive Learner. The Actor and Critic share the same architecture but do not share weights.