COMPETITIVE MULTI-AGENT REINFORCEMENT LEARNING WITH SELF-SUPERVISED REPRESENTATION

DiJia Su, Jason D. Lee, John M. Mulvey, H. Vincent Poor

Princeton University

ABSTRACT

We present MASRL: Competitive **M**ulti-**A**gent **S**elf-supervised representations for **R**einforcement **L**earning in the multi-agent competitive environment. MASRL introduces a simple but effective self-supervised task: predicting a learning agent's opponent's future move. In doing this, the agent learns a stronger representation from this additional signal, focusing not only on itself but also on its opponent. By understanding and anticipating the opponent's future moves, MASRL allows the learning agent to develop effective strategies for opponent exploitation. Our method stabilizes training, improves sample efficiency, and allows the agent to generalize and adapt its playing strategy to other unseen expert opponents. On the Multi-Agent Atari benchmark, MASRL achieves remarkable performance, outperforming other strong baselines. Examples of demo videos can be found at: https://sites.google.com/view/compmarl

1 Introduction

Training reinforcement learning (RL) agents on high-dimensional observations such as pixel images has been a difficult problem. With the recent advancements in deep learning, researchers have shown that combing powerful neural networks, long term credit assignment, and effective RL algorithm, agents can learn to solve complicated tasks such as Atari [1, 2], Go [3], and more [4]. In financial applications, Aiden [5], an automated trading system, employs reinforcement learning for order execution. The scope of RL in finance is very large because the algorithm needs to make rapid and intelligent decisions while accounting for market conditions and other competing players. Despite many successes, researchers have observed that training RL agent on high-dimensional pixel input is sample inefficient [6, 7]. This problem is exacerbated in the multi-agent setting since agents not only need to interact with the environment, but also with other agents.

On the other hand, self-supervised representation learning methods have shown tremendous success and significantly improve the data efficiency especially in the fields of computer vision and natural language processing. Self-supervised methods improve sample efficiency by taking advantage of additional training signals from an almost limitless supply of auxiliary tasks generated on-the-fly. Examples of task include reconstructing masked image-patches [8].

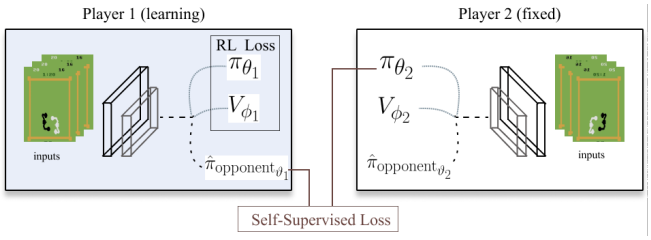

Fig. 1. MASRL training process with self-play. Figure shows that, at the current iteration, player 1 is the learning agent while player 2 is fixed. Input images are stack together and pass through the IMPALA-CNN which is shared between its own policy head θ, the value head ϕ, and the opponent strategy prediction head ϑ. The RL loss comes from π_1, V_1 while the self-supervised strategy loss comes from the cross-entropy between the predicted opponent action ($\hat{\pi}_{\vartheta_2}$) and the actual opponent's action (π_{θ_2}). At every m^{th} training step, the training alternates. Player 2 will become the learning agent with player 1 becomes fixed.

Motivated by successes in self-supervised learning, we propose MASRL: competitive Multi-agent Self-supervised representations for RL. MASRL introduces a simple but effective self-supervised task. During the training process, a learning agent predicts its opponent's future moves as an auxiliary task. In doing this, the agent learns a stronger representation from this additional signal, focusing not only on itself but also on its opponent. By understanding and anticipating the opponent's future moves, MASRL allows the learning agent to develop an effective strategy for opponent exploitation. MASRL is sample efficient and capable of generalizing and adapting its strategy to exploit other unseen expert opponents.

2 Related Works

Self-Supervised Learning for RL comes in the form of auxiliary tasks. Conditioned on the current observations, the auxiliary tasks require the learning agent to perform prediction either in the pixel space or in the latent space [9, 10]. This mechanism has been shown to have the capability to improve sample efficiency in most model-free RL algorithms. An example of a pixel-based auxiliary task is [9]. The task involves

a learning agent to control the pixels on the screen and also to predict the onset of immediate rewards from the historical context. An example of latent space prediction is [10] in which the agent predicts the future in the latent space using auto-regressive models. Unlike any of the aforementioned works, our work introduces a novel self-supervised task which is directly predicting the opponent's future move.

Multi-Agent RL algorithms fall under the following two categories: centralized [11] or decentralized learning [12]. In between the two categories, there are value decomposition, centralized training and decentralized execution (CTDE) [13].

Another line of related work is *opponent modeling*. Prior works in opponent modeling include: exploiting opponent learning dynamics [14, 15], modeling the intention of opponent [16, 17], and incorporating opponent's behavior as features which can be either handcrafted [18] or learned [19]. The closest related work is [19] in which the authors proposed an unsupervised encoder-decoder framework for learning policy representation (as an embedding) across all agents. However, the proposed method: 1) requires interaction with a variety type of opponents throughout the training process; 2) requires additional fine-tuning procedure using either supervised or RL method; and 3) is complicated and requires agent-interaction graph to record the interaction history. Our method is novel because our strategy: 1) doesn't require interaction of any other types of opponent; 2) complete end-to-end training; 3) is a lot more simpler and readily integrate onto any RL framework; and 4) our learned agent can adapt its playing strategy automatically and generalize to other unseen opponents.

3 Background

We consider the following setting of a two player Markov game based on partially observable Markov decision process (POMDP). We denote state space with S , action spaces with A_1, A_2 and observation spaces O_1, O_2 for agent 1 and agent 2, respectively. At each time step t, both agents execute their action $a_1^{(t)} \in A_1, a_2^{(t)} \in A_2$ simultaneously by following each individual's policy $\pi_i : O_i \times A_i \to [0, 1]$. After executed their actions, each agent receives a reward $r_i^{(t)} : S \times A_i \to R$. We denote the state transition functions $T : S \times A_1 \times A_2 \to S$. In the RL setting, the objective of an i_{th} learning agent is to optimize a policy $\pi_i(a_i|o_i)$ such that the cumulative discounted future reward is maximized: $\sum_t^{\infty} \gamma^t E_{a_i^t, a_j^t, s^t}[r_i^{(t)}|s_0 = s, \pi_i, \pi_j]$ with discount factor γ, for $i \neq j$. In the zero-sum competitive game setting, agent j simply receives the reward that is the negative of agent i.

Proximal Policy Optimization. PPO [20] is a state-of-the art on-policy method for learning a policy $\pi_\theta(a|o)$ (with parameter θ) by maximizing $\mathcal{L}_\pi$:

$$\mathcal{L}_\pi(\theta) = \min(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), \\ \text{clip}(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon) A^{\pi_{\theta_k}}(s,a)) \quad (1)$$

with ϵ as the hyper-parameter that determines the distance constraint between and old and the new policy, and A as the advantage function. The PPO also fits a value function by regressing toward the discounted returns: $\mathcal{L}_V(\phi) = \mathbb{E}[(V_\phi(o_t) - V_t^{\text{targ}})^2]$ with ϕ as the parameters for learning the value function. We estimate the advantage function with generalized advantage estimation (GAE). To summarize, the actual gradient update is a combination of the policy loss, value loss and entropy bonus S_π. With c_1 and c_2 as the tune-able constants:

$$\mathcal{L}_{\text{PPO}} = \mathbb{E}[\mathcal{L}_\pi(\theta) + c_1 \mathcal{L}_V(\phi) - c_2 S_\pi(o_t)]. \quad (2)$$

4 Self-Supervised Representation Learning

In the multi-agent setting, we make use of the opponent's future moves as an additional learning signal for enhancing the representation. Our hypothesis is simple. In predicting the opponent's next move, the agent can grasp a strong learning representation by focusing not only on itself but also on its opponent. By understanding, reasoning, and anticipating the opponent's future moves, the learning agent can develop better defensive or attacking strategy and to exploit its opponent.

We denote this *self-supervised* loss by $\mathcal{L}_{\text{self-supervised}}$ for agent i playing against its opponent j, for $i \neq j$. Then, we characterize this loss by taking the cross-entropy between the actual opponent's next move versus the predicted opponent's next move. For agent i:

$$\mathcal{L}_{\text{self-supervised},i}(\vartheta) = -\sum_{a_j \in A_j} \pi_j(a_j|o_j) \log \hat{\pi}_{\text{opponent}_{\vartheta,i}}(a_j|o_i) \quad (3)$$

where $\hat{\pi}_{\text{opponent}_{\vartheta,i}} : O_i \times A_j \to [0, 1]$ is the agent i 's opponent action (strategy) prediction function parameterized by ϑ. With c_3 as a tune-able scalar, we augment the PPO loss eq. (2) with self-supervised loss eq. (3) as:

$$\mathcal{L}_{\text{MASRL},i} = \mathbb{E}[\mathcal{L}_{\text{PPO},i} + c_3 \mathcal{L}_{\text{self-supervised},i}] \quad (4)$$

4.1 Theoretical Analysis

Here, we derive the exact solution for getting the best response strategy with the auxiliary task of predicting opponent's future move. Assuming the existence of the best response strategy, for agent i, we denote Y_i as its best response strategy, Z_i as the auxiliary task of predicting the opponent's playing strategy (next move), and that $\psi(o_i)$ as the learned representation for the auxiliary task. Assume opponent j plays a deterministic strategy, and that the best response strategy is uniquely decomposed as an one-to-one mapping from observation o_i to action $a_i^* = y_i$, with $Y_i = f_i^*(O_1) + N$ where N has mean of 0 and σ^2 sub-Gaussian, and $f_i^* = E[Y_i|O_i]$ is the best action function for agent i. We assume non-degeneracy in $\Sigma_{O_i O_i}, \Sigma_{Y_i Y_i}$ and that O_i, Z_i, Y_i are jointly Gaussian. Then:

Proposition 1 *Assume θ is linear, if $O_i \perp\!\!\!\perp Z_i | Y_i$ (conditional independence) and $\Sigma_{Z_i Y_i}$ has rank $dim(A_j)$, then the optimal θ^* that gives the best response strategy for agent i is*

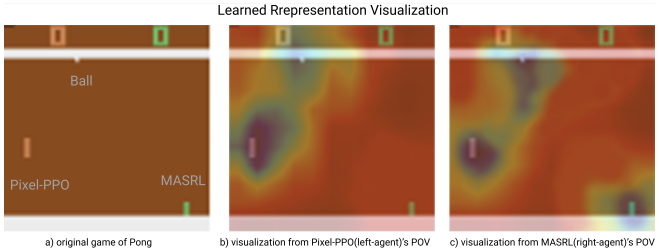

Fig. 2. Grad-CAM visualization gives the attention map of the contribution of each pixel leads to a played action strategy. Plot a) shows the original game of Pong with a Pixel-PPO (brown colored agent) locates on the left playing against with a MASRL (green colored agent) locates on the right. On the plot b), we show the visualization of attention map from the perspective of Pixel-PPO. We see that Pixel-PPO only focuses on itself and the ball, it doesn't pay any attention to its opponent. Thus its played strategy is likely to be passive: simply hit the ball back without opponent exploitation. On the plot c), we show the visualization from the perspective of MASRL. Here, we see that MASRL not only focus to itself, but also focus on its opponent. Only by knowing the location of its opponent, MASRL can predict the strategy it will play and can develop methods for opponent exploitation

$\theta^* = (\Sigma_{Y_i Y_i|O_i})(\Sigma_{Z_i Y_i|O_i})^{-1}$, *Thus,* $\theta^* \cdot \psi_i(O_i)$ *gives the best possible strategy operating on the* $\psi_i(O_i)$ *learned from the self-supervised auxiliary task.*

Proof. We denote the $\mathbb{E}^L[Y|X]$ as $\mathbb{E}^L[Y|X = x] \coloneqq W^*x - b^*$, where $W^*, b^* \coloneqq \arg\min_{W,b}[\|Y - WX - b\|]$. This term represents best linear predictor of Y given X. Adapted from [21], we expand $\mathbb{E}^L[Z_i|O_i]$ as the following:

$$\begin{aligned} \mathbb{E}^L[Z_i|O_i] &= \mathbb{E}^L[\mathbb{E}^L[Z_i|O_i, Y_i]|O_i] \\ &= AO_i + B\mathbb{E}^L[Y_i|O_i] \end{aligned} \tag{5}$$

and that $A = \Sigma_{O_i Z_i|Y_i}(\Sigma_{O_i,O_i|Y_i})^{-1}$,
and that $B = \Sigma_{Z_i,Y_i|O_i}(\Sigma_{Y_i Y_i|O_i})^{-1}$. Here, we see that $\psi_i(o_i) = Ao_i + Bf^*(o_i)$, where $f^*(\cdot) = \mathbb{E}^L[Y|O_i]$. Thus, in the case of conditional independence, $A = 0$ and that if $\Sigma_{Z_i Y_i|O_i}$ has a same rank as $dim(A_j)$ [1], then the optimal $\theta_i^* = (\Sigma_{Y_i Y_i|O_i})(\Sigma_{Z_i Y_i|O_i})^{-1}$. Here, we have denoted the best possible θ^* as the one that outputs the closest possible action wrt the optimal Y_i at each O_i.

5 Training Details

We instantiate two separate neural networks for agent i and agent j. Both agents are trained by playing against each other. To stabilize training and to combat against the non-stationary issue [22] which arises at the MARL setting, while training agent i with eq. (4), we fix its opponent j (fig. 1). At an interval of m training steps, we alternate this training process for agent j by applying eq. (4) while fixing agent i. In doing this, both agents evolve at an interleave manner under the same training methodology. During the evaluation time, we take one of the agents and let it play against with an expert opponent trained from a different method.

[1]Note that in Atari setting, the dimensions of A_i, A_j, and Y_i are the same.

6 Experimentation

6.1 Setup

We used the same IMPALA-CNN architecture (15 convolutional layers) as in [23] and we instantiate two separate IMPALA-CNN networks for player 1 and player 2, respectively. The IMPALA-CNN is responsible for extracting features from the input images, and is shared between its own policy head θ, value head ϕ, and opponent prediction head ϑ (as in fig. 1). We used the image size of 84x84 with $\gamma = 0.99$, $c_2 = 0.1$, $c_1 = 0.5$, $m = 25$, PPO clip to be 0.2, and Adam optimizer with learning rate of 5e-4 and a c_3 =1e-9. Following [24], we trained the agents on 20M frames. Upon finish training the agents, we take the last iteration agent for evaluation. For all results, we averaged over five random seeds.

We examine our methods in the following 14 **multi-agent** Atari environments [25]. We evaluate each methods by playing against one another for 1000 rounds of competition. We report the averaged winning rate.

6.2 Baselines

To establish benchmarks, we have the following baselines:

- MASRL: this is our proposed MASRL agents trained by self-supervised loss as in (eq. (4)).
- Pixel-PPO: this is the default PPO agents (by applying eq. (2) to both agents at an interleave manner) without self-supervised strategy loss.
- MAPPO [26]: this is the multi-agent version of PPO with a centralized value function.
- MAACER, MAACKTR, MAA2C: this is the multi-agent version of ACER [7], ACKTR [27], and A2C, respectively, with a centralized value function.

6.3 The effects on the learned features representation

To understand the effect of MASRL on the learned feature representation, we visualize the spatial attention map from the output of the last convolutional layer using Gradient-weighted Class Activation Mapping (Grad-CAM) [28]. Here, we take the output of the action probability and visualize the contribution of each pixel leading to a played action strategy. We give an example of an expert Pixel-PPO playing against an expert MASRL on game of Pong. In the left plot of fig. 2, we show the original game of Pong with Pixel-PPO as the browned-colored agent (left), and MASRL as the green-colored agent (right). Then in the middle plot of fig. 2, we gave the attention visualization from the perspective of Pixel-PPO (left-agent).

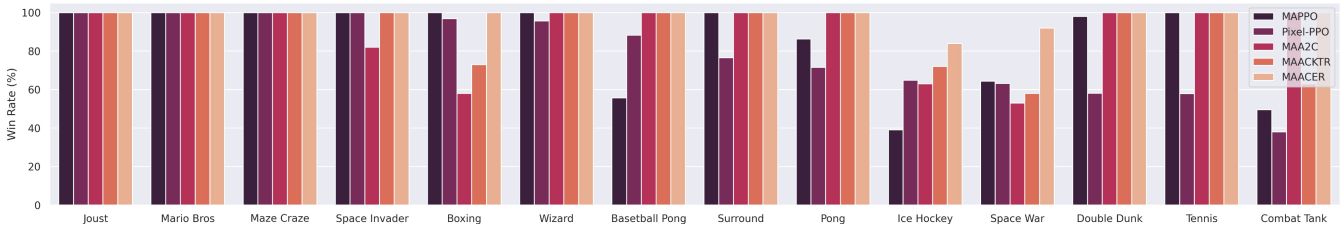

Fig. 3. MASRL vs {Pixel-PPO, MAPPO, MAA2C, MACKTR, MAACER} with y-axis shows the winning rate for MASRL averaged over 1000 rounds of competitions. A percentage greater than 50% indicates that MASRL is a better player. MASRL wins 13/14 games with average win rate of 79.33% (against Pixel-PPO), and wins 12/14 games with average win rate of 85.2% against MAPPO, and wins 14/14 games with average win rate of 89.7%, 90.43%, 99.3% when playing against MAA2C, MACKTR, and MAACER baselines, respectively.

Here, we see that Pixel-PPO only focuses on itself and the ball, and doesn't pay too much attention to its opponent. Ideally, in order to exploit the opponent, we will need to know opponent's current position and/or future moves. Because of Pixel-PPO doesn't pay too much attention to its opponent, it is likely to play a more passive strategy: simply catch and hit the ball back without any opponent exploitation. On the right plot of fig. 2, we show the visualization from the perspective of MASRL (right-agent). Here, we see that it is not only attenting to itself and the ball, but also attenting to its opponent. "To beat your enemy, you must know your enemy", by knowing the location of its opponent, MASRL can predict the strategy it will play and can develop methods for opponent exploitation.

6.4 Full Experimentation Results

Our full empirical results are summarized below. On fig. 3, we show the performance of MASRL playing against several baselines. Starting with result of MASRL playing aginst Pixel-PPO.On the y-axis, we show the winning rate for MASRL with the game name labeled on the x-axis. A winning rate greater than 50% indicates that MASRL is a better player. As is shown in the figure, out of 14 games, MASRL wins 13 of them when playing against Pixel-PPO. The largest winning margin is equal (or almost equal) to 100% in Joust, Mario Bros, Maze Craze, and Space Invaders. Although on Double-Dunk and Tennis, the improvement is not that significant, it still achieves nearly a 60% winning rate. Only on Combat-Tank, we see MASRL loses to Pixel-PPO. On average, we see a 79.33% winning percentage.

Next, on fig. 3 we show the performance of MASRL versus MAPPO. Out of 14 games, MASRL wins on 12 of them, with a tie on Combat-Tank, and a loss on Ice-Hockey. On average, we see MASRL achieves 85.2% winning percentage against MAPPO. Lastly, MASRL wins 14/14 games with average win rate of 89.7%, 90.43%, 99.3% when playing against MAA2C, MACKTR, and MAACER baselines, respectively.

6.5 Best Response.

We study the best response on MASRL. To measure best response, we take a trained Pixel-PPO and fixed it as an opponent to play against with. Then, we instantiate a new agent learning from scratch. Thus, the learning curve shows the degree of exploitation by this new learning agent against a trained Pixel-PPO. In Figure.fig. 4, we instantiate MASRL and training it playing against a fixed and trained Pixel-PPO for 20M frames. We compare the learning curve with Pixel-PPO. Both methods are training from scratch. This learning curve shows the "best response": how good can the agent takes advantage of its opponent? Armed with the capability to predict opponent's next move, MASRL is capable of learning faster and gaining a larger cumulative rewards. This shows that MASRL is developing better opponent exploitation.

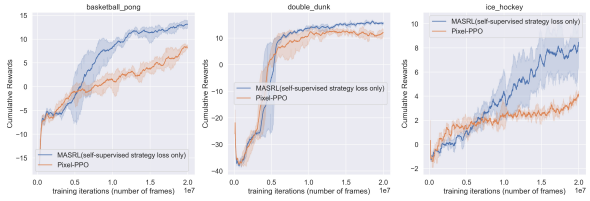

Fig. 4. Best response plot. We show the learning curve for Pixel-PPO and MASRL playing against the same expert opponent (a trained Pixel-PPO). Throughout the entire progress, the expert opponent receives no gradient update. Plot shows that MASRL is capable of learning more cumulative rewards at a faster rate. This shows that MASRL is better at opponent exploitation.

7 Conclusion

We have developed MASRL: Competitive Multi-Agent Self-supervised representations for Reinforcement Learning in the multi-agent competitive environment. Future directions include examine the algorithm in real world applications such as quantitative finance. In particular, financial trading is a tug of war among quant-driven systems (such as Quantitative Brokers). This is an interesting zero-sum multi-agent problem as each competing market player tries to gain advantage over other players. Thus, it could be a valuable area to explore.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[2] D. Su, J. Ooi, T. Lu, D. Schuurmans, and C. Boutilier, "Conqur: Mitigating delusional bias in deep q-learning," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2020, pp. 9187–9195.

[3] D. Silver, J. Schrittwieser, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, 10 2017.

[4] D. Su, J. D. Lee, J. M. Mulvey, and H. V. Poor, "Musbo: Model-based uncertainty regularized and sample efficient batch optimization for deployment constrained reinforcement learning," *arXiv preprint arXiv:2102.11448*, 2021.

[5] H. Burhani, G. W. Ding, P. Hernandez-Leal, S. Prince, D. Shi, and S. Szeto, "Aiden – reinforcement learning for order execution," Nov 2020, Accessed: 2021-08-30.

[6] L. Kaiser, M. Babaeizadeh, P. Miłos, B. Osiński, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al., "Model based reinforcement learning for atari," in *Proceedings of the International Conference on Learning Representations*, 2019.

[7] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

[8] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2536–2544.

[9] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement learning with unsupervised auxiliary tasks," *arXiv preprint arXiv:1611.05397*, 2016.

[10] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[11] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *in Proceedings of AAAI/IAAI*, vol. 1998, no. 746-752, pp. 2, 1998.

[12] M. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the 11th International Conference on Machine Learning*, 1994.

[13] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, et al., "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 2085–2087.

[14] J. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, "Learning with opponent-learning awareness," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 2018, pp. 122–130.

[15] R. Everett and S. Roberts, "Learning against non-stationary agents with opponent modelling and deep reinforcement learning," in *Proceedings of the AAAI Spring Symposia*, 2018.

[16] R. Raileanu, E. Denton, A. Szlam, and R. Fergus, "Modeling others using oneself in multi-agent reinforcement learning," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2018, pp. 4257–4266.

[17] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, S. A. Eslami, and M. Botvinick, "Machine theory of mind," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2018, pp. 4218–4227.

[18] H. He, J. Boyd-Graber, K. Kwok, and H. Daumé III, "Opponent modeling in deep reinforcement learning," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2016, pp. 1804–1813.

[19] A. Grover, M. Al-Shedivat, J. Gupta, Y. Burda, and H. Edwards, "Learning policy representations in multiagent systems," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2018, pp. 1802–1811.

[20] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[21] J. D. Lee, Q. Lei, N. Saunshi, and J. Zhuo, "Predicting what you already know helps: Provable self-supervised learning," *arXiv preprint arXiv:2008.01064*, 2020.

[22] G. Papoudakis, F. Christianos, A. Rahman, and S. V. Albrecht, "Dealing with non-stationarity in multi-agent deep reinforcement learning," *arXiv preprint arXiv:1906.04737*, 2019.

[23] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al., "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *Proceedings of the International Conference on Machine Learning*. PMLR, 2018, pp. 1407–1416.

[24] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[25] J. K. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente, et al., "Pettingzoo: Gym for multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[26] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of mappo in cooperative, multi-agent games," *arXiv preprint arXiv:2103.01955*, 2021.

[27] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *Advances in Neural Information Processing Systems*, vol. 30, pp. 5279–5288, 2017.

[28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, Oct 2019.