Multi-Agent Reinforcement Learning using Centralized Critics in Collaborative Environments

ANDREI-IOAN MIJA
Supervisor(s): Robert Loftin, Frans Oliehoek
EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

A Dissertation Submitted to EEMCS faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

Abstract

Agents trained through single-agent reinforcement learning methods such as self-play can provide a good level of performance in multi-agent settings and even in fully cooperative environments. However, most of the time, training multiple agents together using single-agent self-play yields poor results as each agent tries to learn how to perform their task while their teammates are also learning. Thus, training models to reach an optimal behaviour in such situations becomes a challenging, if not impossible issue to overcome. One possible solution to deal with this problem is to facilitate a centralized training process in which the policies of all agents are evaluated by a *centralized critic* that has access to the observations and actions of all the agents in the environment. By using this approach, the environment becomes stationary and the agents learn in a similar way to using a single-agent algorithm in settings where only one agent needs to be trained. In this paper, we test whether by using a multi-agent reinforcement learning algorithm with centralized critics, as opposed to single-agent ones, we would obtain an agent that generalizes better to new partners in a collaborative environment such as Overcooked, where coordination is critical for good performance. The results display a similar performance between the two algorithms when evaluated through self-play and slightly better or worse results when paired with the human model, representing a mediocre agent, depending on the map. Thus, the multi-agent, centralized critics algorithm used in this study did not train agents that generalize better to new partners. However, the training metrics clearly indicate that the centralized critics method makes the agents learn and converge twice as fast as its single-agent version.

1 Introduction

Reinforcement learning (RL) has become an effective approach to build artificially intelligent systems capable of performing various complex tasks. For example, agents were trained to play video games such as Go [1] and Dota [2] [3] at a level exceeding the skills of the current world champions. It also shows promise in revolutionizing multiple industries by successfully training physical systems to perform a variety of human tasks such as autonomous cars [4].

The simplest and most popular method used to train such agents is self-play (SP), where an agent learns to perform the task at hand by playing with past iterations of itself. However, most of the implementations which make use of SP are designed for single-agent environments and are not always successful when training multiple agents at the same time, as results in [5] show. This is due to a recurrent problem in multi-agent settings, as described in [6], where by using single-agent reinforcement learning methods, the changing policies of other players make the environment non-stationary from each agent's perspective. One solution to this issue is using a multi-agent reinforcement learning algorithm in which one entity, called the centralized critic, evaluates the performance of each agent while having access to the observations and actions of all agents in the environment. Thus, by using a centralized training process in which information is shared between learners, in the form of a common value function, the policies of the other agents become stationary from all agents' points of view by becoming part of the environment.

Good results have been published regarding the high performance of single-agent self-play, when trained and evaluated with other instances of themselves [5]. Nevertheless, the aforementioned paper evaluates the effectiveness of a model trained in a pair through single-agent self-play with a mediocre human model resulting in a very poor performance as opposed to

evaluating the agent with its training partner. At the same time, when first training an agent with a similar human model and then in a pair through self-play, it displayed much better performance than the previously mentioned experiment. Thus, a pattern emerges showing that agents trained through single-agent algorithms using self-play in collaborative environments perform well when paired with the agents they trained with and poorly otherwise, displaying a low level of generalization.

In this study we aim to test whether using a multi-agent reinforcement learning algorithm with centralized critics in a multi-agent environment where coordination and collaboration are critical, such as the game Overcooked [7], we would obtain an agent that would generalize better to new partners than single-agent SP or it would become overfit to its training partner. At the same time, we are also interested in the difference in training performance between the two approaches.

As such, to provide a set of significant results, the research made use of a simplified version of the game Overcooked, where agents were trained through a single-agent deep reinforcement learning algorithm, namely PPO, and its multi-agent version which was implemented by replacing the DDPG[8] algorithm in MADDPG [6] with PPO [9]. The two approaches are compared in terms of training performance by observing the speed at which they each converged to their optimal value of the reward. We also tested the level of generalization to new partners by pairing them with a mediocre agent that they have not seen during training and recording their performance.

The following sections provide details on the study at hand, starting with showcasing and analyzing work related to the problem at hand in Section 2. Next, Section 3 provides an overview of the necessary concepts which are relevant to the study as well as the algorithm used to conduct it. Then, Section 4 details various aspects related to the manner in which the hypothesis was tested such as the environment, agents and experimental setup. The results are provided and discussed in Section 5, while Section 6 addresses the reproducibility aspect of the research. Finally, in Section 7, conclusions are derived from the results and process of the study and recommendations are made for future work.

2 Related Work

The work in [5] focuses on the poor performance of AI agents when paired with human agents in the highly cooperative and collaborative environment based on the popular video game Overcooked [7]. Numerous agents are trained or implemented using different methods such as population-based (PBT) [10] training, in which a population of agents whose policies are parameterized by neural networks are trained with each other, having the least performant agents replaced by mutated versions of the strongest ones. Other agents with similar performance explored in the aforementioned paper are coupled planning with re-planning which consists of a computing near-optimal joint plan for each agent in order to achieve the best possible rewards and re-planning after every joint action. Behaviour cloning (BC) is explained in the Section 3 and it represents the human model which is sometimes used for training, but mainly to test the performance of the other agents.

The results show that, when agents play with other instances of themselves, their performance is very high, whereas when paired with the human model agent, they perform very poorly. This clearly shows the lack of generalization to new partners of single-agent methods in a highly collaborative, multi-agent environment.

OpenAI's Five [3] is an AI which defeated the world champions of the popular e-sports game Dota [2]. The policy is trained via the PPO algorithm and through the SP method. The most important part of this research is that an AI designed for competitive settings displayed the ability to collaborate well with humans. However, this can be due to the increased performance of the AI, which can account for the lack of skill displayed by the human player. Another reason could be that the human players who were paired with the AI were close in skill to it and were thus able to collaborate well.

The results in this paper indicate that training agents in collaborative settings using single-agent reinforcement learning approaches can sometimes result in effective teamwork with new partners and thus, a good level of generalization. Nonetheless, it is unclear whether collaboration was achieved because of the technique used to train the agents or the settings of the environment.

Fictitious Co-Play (FCP) [11] proposes an idea through which an agent that can efficiently cooperate with a human is trained without using any user data. The main point behind the above-mentioned method is that, in order to create an agent that generalizes well to new partners, it should be trained with multiple, diverse partners. Thus, several agents are trained through self-play using different initialization seeds for their neural network, in order to account for symmetry. At the same time, at various checkpoints during the game, snapshots of the agents are saved such that, in the end, one agent is trained through PBT with all the SP agents and their snapshots as part of the population.

This algorithm provided better results in terms of evaluation performance and human preference when compared to simple behaviour cloning (BC), SP or PBT. At the same time, it shows an effective approach to using single-agent algorithms to reach a high level of generalization.

In **K-level Reasoning for Zero-Shot Coordination in Hanabi** [12], the researchers employ the use of the cognitive hierarchies [13] (CH) framework, more precisely, the K-level reasoning [14] (KLR) instance which was difficult to scale in the context of coordination problems, but, by applying two innovations, this approach becomes scalable. Firstly, each level would be trained synchronously rather than sequentially, making them run in parallel and thus reducing the clock time and stabilizing training. Secondly, a best response (BR) agent is trained at the same time on the whole hierarchy with a special focus on the two upper levels. All of the above results in a new state of the art performance when tested together with a proxy human policy which displays a reasonable level of generalization.

Naturally, various multi-agent reinforcement learning algorithms have been designed specifically for situations where more than one agent needs to be trained at the same time. One such algorithm, the multi-agent deep deterministic policy gradient (MADDPG)[6], consists of using a centralized critic, which has access to the observation and value space of all agents, to evaluate the policies of all models during training. MADDPG has presented good results in various experiments such as the "physical deception task" where using the single-agent version, DDPG[8], would result in the agents not completing a task, while using the centralized critics algorithm did not only enable the agents to complete their task but also to do so in an efficient manner. The most relevant experiment for the current work is "cooperative navigation", as it is the only one that does not contain a set of adversaries, being only based on cooperation. The experiments consist of 3 agents which have to occupy 3 different landmarks efficiently and MADDPG has performed twice as well as its independent learning counterpart.

Another, more complex, multi-agent RL algorithm called QMIX [15] follows a different approach. Instead of learning decentralized policies using a centralized value function, Q_{tot}, or multiple, independent value functions for each agent a, Q_a, it combines the two approaches to learn a factored Q_{tot}, similarly to value decomposition networks (VDN)[16]. As opposed to VDN which learns Q_{tot} by summing the individual Q_a of each agent, QMIX uses a mixing network to combine the individual value functions in a non-linear manner, thus representing a richer class of action-value functions while making efficient use of the extra state information. The algorithm was tested against VDN and individual Q learning (IQL)[17], as well as various ablations on obtaining the centralized Q_{tot} from the individual Q_a and has consistently achieved better results on the complex and highly cooperative game called Starcraft II [18].

3 Background

This section first describes relevant concepts for the work at hand in Subsection 3.1, such as a multi-agent Markov Decision Process, the policy gradient method, the actor-critic architecture, and the centralized critic. Subsection 3.2, provides an overview of the algorithms used to train the agents which were later used in the experiments namely PPO and MAPPO, as well as an algorithm that failed to be trained in the environment called MADDPG.

3.1 Preliminaries

Multi-agent Markov Decision Process. A multi-agent Markov Decision Process (MDP) [19] is defined by a set of finite states S, a reward function R which returns a real-value as a result and has the form $R : S \to \mathbb{R}$, α representing a finite set of agents and a finite set of action for each agent i, namely A_i. Finally, a transition function used to go from one state to another $\tau : S \times A_1 \times A_2 \times ... \times A_n \times S \to [0, 1]$. Thus, the multi-agent MDP can be written as a quintuple of all the aforementioned elements $< S, \alpha, \{A_{i\in\alpha}\}, \tau, R >$.

Policy gradient represents a popular reinforcement learning approach in which the parameters θ of a policy are optimised in order to maximize the objective function $J(\theta) = \mathbb{E}_\pi[R_{t+1}]$ by moving in the direction of $\nabla_\theta J(\theta)$ (Equation 1) [6] through gradient ascent.

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s, a)], \tag{1}$$

where $Q^\pi(s, a)$ represents the long-term rewards with the following formula, as per [6]:

$$Q^\pi(s, a) = \mathbb{E}_{s'}[r(s, a) + \gamma \mathbb{E}_{a'\sim\pi}[Q^\pi(s', a')]] \tag{2}$$

As opposed to value based reinforcement learning, a policy gradient algorithm is more effective and efficient for continuous action space while also converging quicker. However, this method suffers from high variance gradient estimate which is intensified in multi-agent settings as mentioned in [6].

The **actor-critic architecture**, as stated in [20], employs the use of two networks as opposed to one used by policy gradients: actor and critic. The actor network operates as the policy which the agent uses to select an action, while the critic network represents the value function used to evaluate, or criticize, the action taken by the agent.

As described in [21], the actor-critic architecture can help reduce the variance caused by policy gradient algorithms by employing the use of an advantage function for the critic network which subtracts a baseline from the estimated value function. By using a state value function, $V^\pi(s)$, as the baseline, the estimate of the advantage function becomes $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$. Replacing $Q^\pi(s,a)$ with $A^\pi(s,a)$, the new formula becomes:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) A^\pi(s,a)], \tag{3}$$

where p^π is the state distribution.

The **centralized critic** architecture builds on top of the actor-critic method by only using one common critic to evaluate the policies of all actors. This approach solves a recurrent problem found in multi-agent settings where the changing policies of other players make the environment non-stationary from each agent's perspective [6]. Thus, by allowing the agents to share information amongst themselves during training, the policies become part of the environment which, in turn, becomes stationary. It is important to note that the centralized critics are used only during training while the execution is kept decentralized.

Formally, following the centralized critic definition from [6] and adapting it to use the advantage function instead of the Q function, we establish a game with N agents with the set $\pi = \pi_1, \pi_2, ..., \pi_N$ representing the policies of all agents, parameterized by $\theta = \theta_1, \theta_2, ..., \theta_N$. Then, the expected return for each agent is expressed as $J(\theta_i) = \mathbb{E}[R_i]$ and its gradient as:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s\sim p^\pi, a_i\sim\pi_i}[\nabla_{\theta_i} \log \pi_i(a_i|o_i) A_i^\pi(x, a_1, ..., a_N)], \tag{4}$$

where $A_i^\pi(x, a_1, ..., a_N) = Q_i^\pi(x, a_1, ..., a_N) - V_i^\pi(x)$ represents the centralized advantage function, or centralized critic, as it takes as input the actions of all N agents and state information x which, in the current case, consists of the observations of all the agents in the environment.

3.2 Algorithms

Multi **A**gent **DDPG** (**MADDPG**)[6] represents the multi-agent version of DDPG[8], a single-agent RL algorithm which uses the actor-critic architecture. To obtain the MADDPG algorithm, DDPG's critic is enhanced with more information regarding the other agents' policies, similarly to Equation 4 but replacing A_i^π with Q_i^π, thus bringing a very simple extension of actor-critic policy gradient methods. The Q-function in this case is a centralized action-value function that takes as input the actions of all agents together with some state information that results from the observations of the agents within the game. This ultimately means that for each agent, all the other agents become part of the environment, thus making it stationary even as the policies change.

Proximal **P**olicy **O**ptimization (**PPO**)[9] is currently one of the state of the art approaches for policy learning within reinforcement learning. It is a policy gradient algorithm that follows the *Trust region* idea from TRPO [22] while providing a relatively simple implementation, understanding and parameter tuning. The objective function consists of multiplying the advantage estimate multiplied with the ratio of the probability of choosing an action given the new policy and the old policy, respectively. In order to deal with the common

policy gradient problem of updating the policy at every step and thus ruining it, the PPO algorithm uses a clipping function L^{CLIP} defined in Equation 5, to ensure that the gradient steps are not too large.

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{5}$$

Multi **A**gent **PPO** (**MAPPO**) is a multi-agent version of the aforementioned PPO algorithm which was implemented for the purpose of this research. The PPO algorithm uses a generalized advantage estimation which runs the policy for T timesteps before updating it. By augmenting the advantage function in a similar style to MADDPG and following the centralized critic formula depicted in Equation 4 from Subsection 3.1, the PPO algorithm can be successfully converted into a centralized critic algorithm. More precisely, this is done by concatenating the agent's observations with the observations and actions of the other agents, then providing the resulting vector as input for the policy neural network.

4 Methodology

In order to test whether a centralized critics algorithm generalizes better to new partners than a single-agent approach, we used various tools from the study done in [5]. This includes the simplified version of the game Overcooked [7] and various algorithms such as self-play using PPO and behaviour cloning using human data. The setup is explained in detail in the next paragraphs.

4.1 The Overcooked environment

The environment used to conduct this research is the same as the one used in [5] and it represents a simplified version of the game Overcooked [7]. In the game, players need to prepare and deliver as many dishes as possible in a given amount of time in order to maximize their score.

The authors of [5] found the setting of the game as being challenging for collaboration, but especially for coordination which not many environments are designed for. Given the complexity of the real game, it would be computationally expensive to train agents. As such, the simplified version only contains onions, pots, dishes and delivery zones. The agents need to pick up three onions from the onion dispenser, put them in a pot and wait for 20 timesteps for the soup to cook. Then, they need to use a dish to pick up the soup and deliver it at the designated location. Upon delivery, the agent will receive a reward.

There are multiple layouts available that require various levels of coordination and collaboration from *Asymmetric Advantages* where coordination would improve efficiency to *Forced coordination* where, without collaboration, no soups can be delivered. Different layouts can have a different number of dispensers, pots and delivery zones. At the same time, the possible actions that the agents can take are moving to the tile above, below, to their left and right as long as there is still available space to move in the chosen direction. The players can also *interact* with the tile which they are facing or stay at the current location and do nothing.

The final goal of the agents is to learn how to deliver dishes in an efficient manner by collaborating with their teammates and taking advantage of the map layout. The environment is designed as a Markov Decision Process to facilitate the usage of reinforcement learning algorithms.

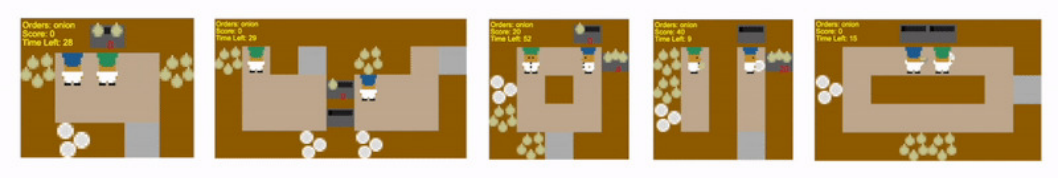

Figure 1: Room layouts as described in [5]. From left to right: *Cramped Room* where both players have access to all resources but they can easily collide. *Asymmetric Advantages* eliminates any collision problems and allows both players to work independently. Nevertheless, working together will greatly improve efficiency as the green hat player has the delivery zone closer to the pot, whereas the blue hat one has the onion dispenser closer to the pot. *Coordination Ring* requires players to coordinate when moving around the map as collisions can easily take place. *Forced Coordination* restricts the players from preparing and delivering a dish by themselves and, as such, forces them to coordinate. *Counter circuit* contains an inapparent coordination strategy, where, instead of going around the counter to get the onions and place them in a pot, one agent can pass them over the counter while another one pots them.

4.2 Agents

This subsection presents the methods chosen to train agents which were used to evaluate the performance of the algorithms during training as well as their level of generalization.

Behaviour Cloning [23] is a type of imitation learning. The specific model used in this research is the same as the one used in [5]. As such, it uses supervised learning methods to learn a policy based on recorded gameplay by learning how to map observations to actions.

The data used by the BC-trained agents to learn was collected during the [5] study through Amazon Mechanical Turk. It has been filtered depending on the optimal trajectories and returned reward resulting in approximately 16 human-human trajectories per layout. Each trajectory is split into two single-agent trajectories. In the aforementioned work, the researchers used half of the data to create a model used to aid the training of other agents and a half to create an agent to test against. However, given that we only train our agents through self-play, all the collected data was used to train a single model for testing purposes. Lastly, the data used in this research is from 2020 whereas the one used in the previously mentioned paper is from 2019.

Self-play is a reinforcement learning method in which an agent learns by playing with different instances of itself. The self-play method is used to train multiple agents through various deep reinforcement learning algorithms such as PPO, MADDPG and a multi-agent version of PPO which we refer to as MAPPO.

Normally, it is expected that the multi-agent algorithms would perform better than their single-agent counterpart considering that, during training, the critic has access to the obser-

vations and actions of all agents [6]. However, the multi-agent version of the PPO algorithm implemented in this work, MAPPO, does not show any considerable increases in performance over its single-agent counterpart.

4.3 Experimental setup

In order to test our hypothesis, three agents are trained, one through a single-agent RL algorithm, another through multi-agent RL and the last one using behaviour cloning on human data.

For the single-agent RL algorithm, PPO was chosen since it was already implemented within the Overcooked environment, showcasing a good performance. Nonetheless, it is important to note that in [5], the PPO algorithm uses an observation space that consists of a featurization of the Overcooked state environment into a stack of boolean masks that are compatible with a CNN. Since the NeurIPS submission in 2019 which corresponds to the **neurips2019 branch**[1], the library used to train the agent was changed from stable-baselines [24] to RLlib [25] and thus, the underlying implementation of PPO changed as well. This means that the hyperparameters listed in the aforementioned paper were no longer valid and new ones needed to be tuned. Thus, to ease the hyperparameter optimisation process, we used the same hand-crafted observation space as the BC agent.

Multiple issues were present while integrating the multi-agent algorithm in the Overcooked environment. Initially, MADDPG was chosen as there was already an implementation available from RLlib. However, as stated in the description of the algorithm [26], MADDPG is very hard to get to work and combined with the fact that new algorithms are hard to optimise in the Overcooked environment and the lack of time available for conducting this research, training agents through it was not successful.

The next multi-agent algorithm considered which was successfully integrated within the environment is a centralized version of PPO which we previously referred to as MAPPO. However, there is one important factor to note which is that the underlying model of the neural network in MAPPO is different than the one we use in PPO, as the version from [5] used a custom model, whereas the MAPPO uses the default model provided by RLlib, both of which are detailed in Appendix A. Various attempts were made at adapting the centralized critic version to use the custom PPO model but eventually proved unsuccessful. Nonetheless, the same set of hyperparameters obtained from tuning PPO provide similar results when applied to MAPPO.

5 Results

The first experiment consists of training a PPO and a MAPPO pair of agents and comparing their performance over the learning process. Both agents used the same set of hyperparameters which are also listed in Appendix A and converged to almost the same reward level when trained in self-play. However, the MAPPO pair converged twice as fast as the PPO agents as illustrated in Figure 2, where averaged rewards during training are shown from 3 different maps of 3 seeds each. These results prove that the centralized critic algorithm

[1] `https://github.com/HumanCompatibleAI/human_aware_rl/tree/neurips2019`

learns faster than the single-agent version and is thus more efficient in terms of data and computational efficiency.

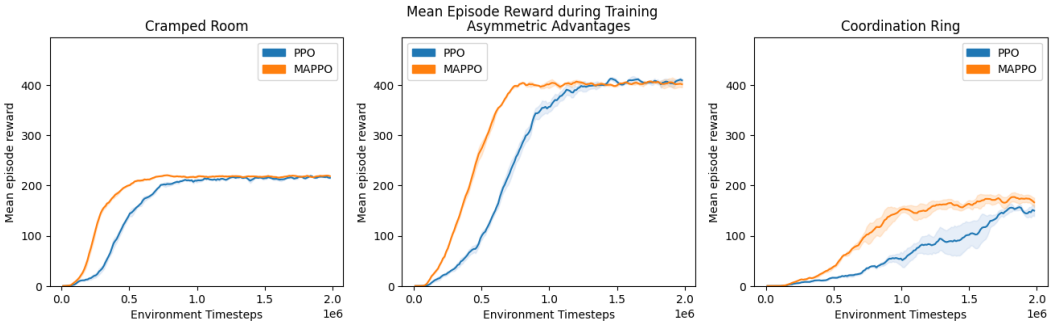

Figure 2: Average rewards per episode (400 timesteps per episode) registered during self-play training for different pairs of agents. In each graph, the orange line represents the performance of the MAPPO agent while PPO is displayed in blue. In the *Cramped Room* layout, MAPPO seems to be converging at around 0.5 timesteps, whereas PPO converges at approximately 0.75 million timesteps. In the *Asymmetric Advantages* map, MAPPO stabilizes at around 0.6 million timesteps and PPO at around 1.2 million. Finally, in *Coordination Ring*, MAPPO converges at approximately 1 million timesteps while PPO only does so at 2 million. Overall, a general trend is observed in the sense that MAPPO converges to an optimal policy, on average, twice as fast as PPO.

The second experiment entails comparing the generalization level between the two algorithms. To test this, a BC agent was trained using the 2020 human data recorded after the study in [5]. Then, the MAPPO and PPO agents were each paired with the BC agent in order to register the average reward over 100 rollouts of 400 timesteps each, thus providing a reliable metric for the level of generalization between the two approaches. Figure 3 clearly shows that when paired with themselves, the agents display a similar level of performance which is the same as the one registered during training after each algorithm converged. Similarly, when each agent is paired with the human model, only a relatively small difference can be observed between the reward gained by the team with the PPO agent and the one with the MAPPO model. However, one noteworthy fact to mention is that the pairs which contain a MAPPO trained agent tend to have a smaller variance and can thus be considered to be more stable across rollouts.

Furthermore, whenever the self-play team using one algorithm performed better on a layout, the team consisting of the human model and that algorithm also performed slightly better than its counterpart on that same map. Moreover, even though the difference in performance is small, the MAPPO agents proved to be slightly more effective on maps where agents are likely to collide such as *Cramped Room* and *Coordination Ring*. The PPO trained agents perform slightly better on the *Asymmetric Advantages* layout where agents have all the tools necessary to complete and deliver dishes by themselves and can thus complete the game while never collaborating.

One last point to mention is that the performance of self-play agents is considerably higher than the one of the team consisting of the human model and the self-play trained agent. This

is consistent with the findings in [5], emphasising the idea that agents trained through self-play usually perform poorly in collaborative environments when paired with new partners.

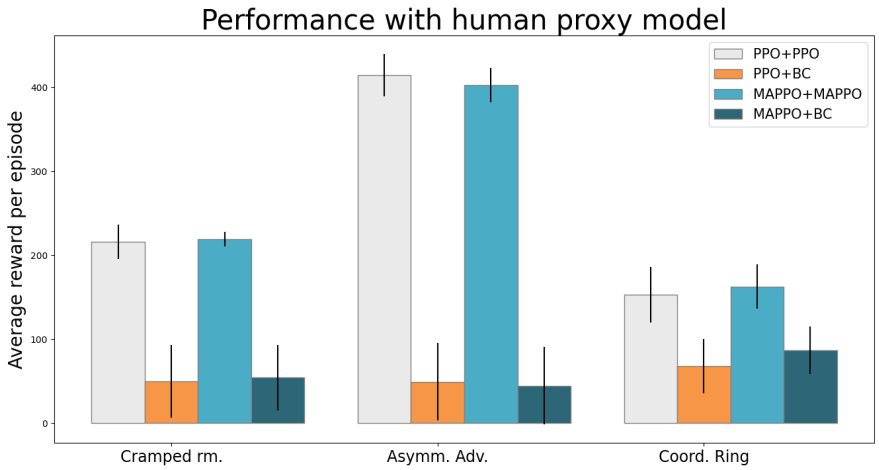

Figure 3: Average rewards per 100 rollouts of 400 timesteps each, for different pairs of agents. The light grey bar represents the performance of PPO when playing with itself. In orange we register the average rewards per episode of PPO paired with the human agent trained through BC, while the 2 blue plots represent the performance of a pair of MAPPO (light blue) and MAPPO playing with the same BC agent as PPO. When evaluated both in self-play and with the human model, both PPO and MAPPO show similar performance across all 3 layouts when averaged across 3 seeds (per layout).

6 Responsible Research

To facilitate the process of responsible research, two main aspects need to be discussed, namely the ethical implications of the study as well as the reproducibility factor. The former is not a concern for this study as the research is conducted within a virtual environment and the main research question to answer is whether one approach trains agents that generalize better to new partners than another. The only element which could represent an ethical concern is the privacy factor of the human data used to train the behaviour cloning agent to obtain the human model. However, this data is the 2020 version of the one used during the work in [5] and it is widely available at the repository associated with their work. At the same time, the data only contains the trajectories which the humans created while playing the game, without any explicit or implicit reference that could lead to finding the identity of the people who took part in the study.

The reproducibility aspect has been accounted for since the beginning of the research by thoroughly recording the process and the results obtained throughout the study to eventually offer a comprehensive overview of how the work has been conducted and what the findings are. The code is built on top of the **master branch**[2] of the repository used for the study in

[2] `https://github.com/HumanCompatibleAI/human_aware_rl`

[5]. Various additions were made such as an evaluation setup inspired by their **neurips2019 branch**[3] which was used for the final submission of the aforementioned paper, as well as implementing the necessary models for the centralized critics algorithm. The version of the code used to obtain the results from Section 5 is available in a GitHub repository[4], together with instructions on how to run the experiments. Furthermore, the seeds used to conduct the study as well as the hyperparameters used to train the agents are present in the Appendix A.

In order to provide a set of reliable results the agents were trained on multiple layouts and various seeds. Moreover, all displayed data represents an averaged return over the evaluation of multiple seeds that were used during training.

7 Conclusions and Future Work

Summary. Using a multi-agent, centralized critic algorithm to train agents in a collaborative environment did not provide models that generalize better to new partners than the single-agent version of the algorithm. However, the centralized critic approach aids the agents to train almost twice as fast as the single-agent counterpart, while keeping the same level of generalization in the simplified version of the game Overcooked. At the same time, in various layouts where coordination was more relevant than in others, the multi-agent architecture resulted in agents that perform slightly better on average and more consistently as they present lower variance over the rollouts. Nevertheless, the increase in performance can potentially be attributed to the randomness of the initialization seeds as the difference between the results displayed by each algorithm is relatively small and there is not one single algorithm always outperforming the other.

Limitations and future work. Firstly, as mentioned in Subsection 4.3, the observation space used to train the PPO and MAPPO agents is the same as the handcrafted one used to train the human model. As such, it contains heuristics which makes it easier for the agent to learn. Initially, this observation space was used while attempting to train agents through MADDPG. The centralized critic algorithm was designed to concatenate the observation space of the two agents and the larger, more detailed observation space which was used by PPO was not compatible with that architecture. When MADDPG failed to train, it was decided to keep the BC observation space as we could easily customize PPO to use the same architecture as MADDPG. At the same time, the BC observation space is easier to train as it has been crafted to facilitate the learning process in this environment, and, due to the time constraints, that was an added bonus. Nonetheless, the BC observation space might not be as expressive as the observation space used by PPO in [5]. As such, for future work, one can apply the centralized critics using the more expressive and general observation space in the attempt to get the agent to generalize better.

Secondly, one successful approach discussed in [5] to train agents that would generalize well to new partners, more specifically humans, consists of training an agent with a human model and then linearly increase the number of episodes in which the agent trains against itself, until only training through self-play. Using this strategy, one can test whether a centralized critic algorithm would generalize to new partners better than a single-agent one when first training with a mediocre partner and then through self-play.

[3] `https://github.com/HumanCompatibleAI/human_aware_rl/tree/neurips2019`
[4] `https://github.com/andrei-07/rp-overcooked-centralized-critics`

While the performance of the multi- and single-agent approaches is similar in our study, it is not yet known how the behaviour of the agents changes. There is a possibility that the centralized critic has a better way to avoid collision whereas the single-agent algorithm increases the agent's performance when working separately. Nevertheless, all of these assumptions are hard to test when the agents cannot be observed playing the game. Thus, one can adapt the visual representation of the game from the neurips2019 branch used for the work in [5] to be able to use it with the agents in this study. By having the ability to observe the behaviour of the agents in real-time, more conclusions can be drawn regarding the advantages and disadvantages of the two types of algorithms and their performance.

References

[1] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017. arXiv: 1712.01815. [Online]. Available: http://arxiv.org/abs/1712.01815.

[2] VALVE, *Dota 2*, https://www.dota2.com/home".

[3] OpenAI, *Openai five defeats dota 2 world champions*, https://openai.com/blog/openai-five-defeats-dota-2-world-champions/, Apr. 15, 2019.

[4] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 427–438, 2013. DOI: 10.1109/TII.2012.2219061.

[5] M. Carroll, R. Shah, M. K. Ho, *et al.*, "On the utility of learning about humans for human-ai coordination," *CoRR*, vol. abs/1910.05789, 2019. arXiv: 1910.05789. [Online]. Available: http://arxiv.org/abs/1910.05789.

[6] R. Lowe, Y. WU, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf.

[7] Ghost Town Games, *Overcooked on steam*, https://store.steampowered.com/app/448510/Overcooked/, 2016.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. arXiv: 1707.06347. [Online]. Available: http://arxiv.org/abs/1707.06347.

[10] M. Jaderberg, V. Dalibard, S. Osindero, *et al.*, "Population based training of neural networks," *CoRR*, vol. abs/1711.09846, 2017. arXiv: 1711.09846. [Online]. Available: http://arxiv.org/abs/1711.09846.

[11] D. Strouse, K. R. McKee, M. M. Botvinick, E. Hughes, and R. Everett, "Collaborating with humans without human data," *CoRR*, vol. abs/2110.08176, 2021. arXiv: 2110.08176. [Online]. Available: https://arxiv.org/abs/2110.08176.

[12] B. Cui, H. Hu, L. Pineda, and J. Foerster, "K-level reasoning for zero-shot coordination in hanabi," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 8215–8228. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/4547dff5fd7604f18c8ee32cf3da41d7-Paper.pdf.

[13] T. Ho, C. Camerer, and J.-K. Chong, "A cognitive hierarchy model games," *The Quarterly Journal of Economics*, vol. 119, pp. 861–898, Feb. 2004. DOI: 10.1162/0033553041502225.

[14] M. A. Costa-Gomes and V. P. Crawford, "Cognition and behavior in two-person guessing games: An experimental study," *American Economic Review*, vol. 96, no. 5, pp. 1737–1768, Dec. 2006. DOI: 10.1257/aer.96.5.1737. [Online]. Available: https://www.aeaweb.org/articles?id=10.1257/aer.96.5.1737.

[15] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, PMLR, Oct. 2018, pp. 4295–4304. [Online]. Available: https://proceedings.mlr.press/v80/rashid18a.html.

[16] P. Sunehag, G. Lever, A. Gruslys, *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *CoRR*, vol. abs/1706.05296, 2017. arXiv: 1706.05296. [Online]. Available: http://arxiv.org/abs/1706.05296.

[17] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *In Proceedings of the Tenth International Conference on Machine Learning*, Morgan Kaufmann, 1993, pp. 330–337.

[18] Blizzard Entertainment, *Starcraft II*, https://starcraft2.com/en-us/.

[19] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, ser. TARK '96, The Netherlands: Morgan Kaufmann Publishers Inc., 1996, 195â210, ISBN: 1558604179.

[20] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," in Second. The MIT Press, 2018, ch. 11.1. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html.

[21] D. Silver, *Lectures on reinforcement learning*, URL: https://www.davidsilver.uk/teaching/, 2015.

[22] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. arXiv: 1502.05477. [Online]. Available: http://arxiv.org/abs/1502.05477.

[23] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine Intelligence 15*, Oxford University Press, 1996, pp. 103–129.

[24] A. Hill, A. Raffin, M. Ernestus, *et al.*, *Stable baselines*, https://github.com/hill-a/stable-baselines, 2018.

[25] E. Liang, R. Liaw, R. Nishihara, *et al.*, "Ray rllib: A composable and scalable reinforcement learning library," *CoRR*, vol. abs/1712.09381, 2017. arXiv: 1712.09381. [Online]. Available: http://arxiv.org/abs/1712.09381.

[26] Ray, *Rllib maddpg documentation*, https://docs.ray.io/en/latest/rllib/rllib-algorithms.html#maddpg, 2018.

A Implementation details for PPO and MAPPO

Feature space. Both algorithms are trained using the same manually designed 64-dimensional featurization the state used by the BC agent in [5]. Given the limited amount of human-human data, this observation space was specifically designed to facilitate the learning of the human model trained through BC. As such, this featurization provides a quicker and easier training process, while taking away some of the performance. The observation space consists of the relative position of each player from various points of interest within the map such as the closest onion, dish, soup, onion dispenser, dish dispenser, serving location, and pot (one for each pot state: empty, 1 onion, 2 onions, cooking, and ready). It also includes boolean values that encode the agent's direction and indicate if the agent is close to empty counters. The absolute position of the agent in the layout is also incorporated in the feature space.

Reward shaping. Normally, the agent only receives a reward when delivering a dish. However, to allow for a swift training process, a dense reward is added to the environment by providing the agents with additional rewards when placing an onion in a pot (3 reward points), when picking up a dish while a soup is being cooked (3 reward points) and a reward of 5 when picking up the soup with a dish. In [5], the dense reward is reduced to 0 over time. Nonetheless, reducing the dense reward in the experiments presented in this work provided negative results.

PPO model is parameterized by 3 fully-connected layers with hidden size 64, as convolutional neural networks could not be used because of the BC feature space.

MAPPO model is parameterized by 2 fully-connected layers with hidden size 256. The centralized critic takes as input the observation of the current agent (96), the observations(96) of the opponent and the actions (6) of the opponent and concatenates them, resulting in a 198 one-dimensional vector which is fed as input to one layer of size 64, followed by one of size 16 and an output layer of size 1. All activation functions are relu.

To train the agents used for the experiments, 3 seeds (2229, 7649, 7225) used. The same hyperparameters were used for training both the PPO and the MAPPO agents across all 3 layouts and are listed below:

- Learning rate = 0.0001
- VF coefficient = 0.5
- Rew. shaping horizon
- Training batch size: 12000
- Minibatch size: 2000
- Entropy coefficient = 0.2 (linearly decreasing to 0.1),
- Gamma = 0.9,
- Lambda = 0.98
- Clipping = 0.2
- Gradient steps per minibatch per PPO step = 8