Fictional Co-Play for Human-Agent Collaboration

Nathan Ordonez
Supervisor(s): Robert Loftin, Frans Oliehoek
EEMCS, Delft University of Technology, The Netherlands

June 19, 2022

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

Abstract

A longstanding problem in the area of reinforcement learning is human-agent collaboration. As past research indicates that RL agents undergo a distributional shift when they start collaborating with human beings, the goal is to create agents that can adapt. We build upon research using the two-player Overcooked environment to reproduce a simplified version of the Fictitious Co-Play algorithm in order to confirm past found improvements at a smaller scale of training and using Self-Play and Population-based trained algorithms as the baselines for comparison. We find that the agent on average slightly outperforms both baseline algorithms when evaluated using a human proxy. We also find high cross-seed variance in performance, indicating the potential for further hyperparameter tuning.

1 Introduction

Generating artificial agents that can play videogames is a longstanding goal in the field of artificial intelligence [1], and it's an important way to explore novel techniques that might be useful for solving important real-life tasks using AI agents. Similarly, having agents solve tasks by collaborating with humans is also a longstanding problem [2], and the hope is that useful agents will increasingly help human beings in achieving their own tasks. One paper that combines agents that can play games, and those same agents collaboratively solving tasks with a human being, is [3]. It presents a framework emulating the Overcooked videogame [4], where multiple agents (and humans) collaborate in order to complete some cooking tasks. The motivation for choosing this game as inspiration, according to the authors [3], is that whereas many other games allow agents to perform quite well by simply developing good individual gaming skills, this one instead yields the largest amount of performance through teamwork.

This framework has been experimented on in many ways, using many different approaches [5] [6] [7], and one particularly well-performing reinforcement learning method building on the basic method known as "population-based training" is shown in [8]. The method, which they coined "Fictitious Co-Play", is primarily motivated by the idea that in order to train agents that can adapt to playing with human beings, they should be trained with training partners that are diverse in their ways of playing. What is then special about their approach is that they seem to achieve this without the need for any actual human beings being involved in the training, or even any human data. Instead, the diverse set of training partners is generated synthetically by using an older reinforcement learning training technique known as self-play. In essence, the novel method takes advantage of the fact that the old method creates specialized, non-generalized agents, in order to create a diverse set of training partners.

While the stated results of the Fictitious Co-Play paper suggest that the new method presents a significant improvement on previous methods (more than doubled performance, see figure 5 [8]) , there are a few remaining questions relating to the conditions under which their paper was written. Since the authors have not released the code used to produce the paper as of this writing, and since they are using large amounts of computing power in the training of their agents, two things are unclear. First, we do not know if the primary insight of their method is actually enough to improve upon the techniques tested in the original overcooked paper [3]. Second, we do not know if the positive results they got are also significant in smaller-scale experiments.

Thus, this paper investigates the following question: **can the FCP method generate**

agents that adapt well to human beings? To do this, we focus on reproducing the state-of-the-art technique "Fictitious Co-Play" and a subset of experiments from [3] in order to find out whether a minimal implementation of their method, trained and evaluated using more accessible levels of computation (consumer-grade, single-GPU training) is able to show results that help confirm the hypothesis that the agents can indeed adapt to playing well with human beings. This will be our main research question and contribution to the field. As a secondary goal, by our attempt to answer the main question, we also address the gaps in the RL literature following the original FCP paper concerning the performance of its primary insight, and whether the previously shown performance gains persist in smaller-scale experiments.

The agent is therefore implemented using the Overcooked code from [3] as a basis, and evaluated using their human proxy agent, due to logistical reasons not allowing tests to be performed on real human beings. Then, the evaluation is compared to two other algorithms (Population-based training, Self-Play) used as a baseline by the Fictitious Co-Play paper. The results show that our agent performs slightly better than the baseline methods, which is very different from the results found by [8]. Further, we see that the variance of our agent is the highest among the three, and that there is much room for improvement by doing hyperparameter tuning.

This paper will use the following structure: First, in section 2 we will talk about related work as it relates to the problem our method is trying to solve. Then, in section 3, background will be given on the FCP method, and a short introduction to the main, important concepts will be included as well. The methodology will be described for how our experiments were designed in chapter 4, and the design choices made will be motivated. Next, the results of the experiments will be shown and discussed in section 5, followed by section 6 on performing responsible research. Finally, we will conclude the paper in section 7 with a short summary of our main conclusions, and make some suggestions for future work that could be based on this project.

2 Related Work

Diverse training partners An idea found in reinforcement learning is that diversity in training partners can help an agent adapt better. We will mention two relevant papers on this subject. One [9] shows that by using diversity as a reward function for an RL agent, it can develop entire skills (such as walking) on its own. Second is [5], which builds on this idea by developing another agent for the Overcooked environment. In their method, a group of agents are trained using population-based training, and the agents are partly rewarded by how different they are to the rest of the group. This results in a group of agents that has developed more different behaviors (which can be seen as analogue to the "skills" developed by the agents of this subsection's first paper). They also compared their agent to a method known as *TrajeDi* from [10], which introduces a generalised measure of similarity between two distributions, based on the *Jensen-Shannon Divergence* [11] which is then used as an objective for generating agent policies that work best for a maximal number of possible partners. Both of their results showed good performance, even when compared to the method used in this paper.

human-agent interaction The original overcooked paper [3] posits that when an agent plays with a human after training without any humans, it undergoes a distributional shift.

This means that in order to play well with humans, the agent must have generalised its playing ability with regards to its training partner. A recent paper [6] by some of the same authors as the FCP paper [8] focused on this concept in, among others, the Overcooked environment, and found that a heterogeneous set of training partners does indeed increase the generalisation of an agent.

ad-hoc teamplay In some environments, two agents could discuss or share their strategies before the actual task begins. Ad-hoc gameplay is when there is no such preliminary coordination. In this paper, we tackle the latter type, which means our agents have to work with an agent that they do not know anything about. One paper which directly focuses on this challenge in the Overcooked environment (among other environments) is [7]. In their work, they developed an agent with the ability to guess which task its teammate is performing. Additionally, [12] also contributed to this issue by focusing on symmetry breaking. In their method, coined "Other-Play", a self-play agent is paired with itself in the game of Hanabi, but with randomly relabeled environment elements. The goal is for the agent to learn to recognize and adapt to symmetries that they might face against any other agent. A symmetry, for example, is when two agents are blocking each-other's path, and they both need to decide who gives up on their original trajectory. Their results showed yet another improvement on the self-play method.

‘’

3 Background

3.1 Preliminaries

Decentralized Partially Observable Markov Decision Process A Dec-POMDP [13] is the most accurate way of describing the problem faced by agents in the Overcooked environment, due to the fact that each player may have different observations of the environment, potentially containing different information. The following is a detailed explanation as it pertains to its usage in this paper's agent. In the case of a pair of agents, the process consists of the following tuple:

$$\mathcal{M} = \langle I, S, \left\{A_{i\epsilon\{1,2\}}\right\}, P, R, \left\{\Omega_{i\epsilon\{1,2\}}\right\}, O, h\rangle$$

I is the set of agents, S is the finite set of states with initial state s_0; A_i is the set of actions available to the i-th agent, with $A = \mathsf{X}_i A_i$ being the set of joint actions; $P(s'|s,a)$ are the state transition probabilities, from state s to s' given agent action a. R: S $\rightarrow \mathbb{R}$ is the real-valued reward function; Ω_i is the set of observations of the i-th agent, with $\Omega = \mathsf{X}_i \Omega_i$ being the set of joint observations. O are the $O(o|s,a)$ probabilities that agents see observation o, given state s and agent actions a. h is the horizon, and when it is infinite a discount factor $0 \leq \gamma < 1$ is used.

The main objective is the following. The agents will determine the state transition probability function $\mathcal{P}$ so as to maximize the expected sum reward $\sum_t R(s_t, (a_t^1, a_t^2))$ for state s and actions a^1, a^2 for each agent, at time t. Because the objective in this case has a technically infinite horizon (although this does not have to be the case), the sum would normally be infinite. This is where we would use the discount factor γ.

Behavioral Cloning Behavioral cloning (as included in [14]) is a method for training an agent's policy based on recordings of human data. It applies a standard supervised learning method to the discrete action space of our Overcooked environment. Our model, taken from [3], turns an encoded state into a probability distribution over the possible actions, and was trained using standard cross-entropy loss. It is meant to be a simple, yet effective way of simulating an actual human partner for validation. For that reason, we use it in this paper as a way to evaluate agents and compare them to each-other.

Proximal Policy Optimization Proximal Policy Optimization (PPO) [15] is a method based on policy gradient reinforcement learning. It works by collecting data samples (by playing in the environment) and then sampling mini-batches from the data samples in order to optimize an objective function approximator (in the form of a neural network). Most training methods mentioned in this paper (PBT, FCP, SP) use a PPO implementation as the learning algorithm that is run on each individual agent.

Population-based training Population-based training (PBT) [16] is another technique used to train agents in reinforcement-learning environments. It works by initializing a population of agents with different hyperparameters which will play with each-other in random sub-groups (in our case, pairs). After a set amount of training, agents are evaluated, and a genetic algorithm is used to remove the worst-performing agents, as well as to multiply well-performing ones.

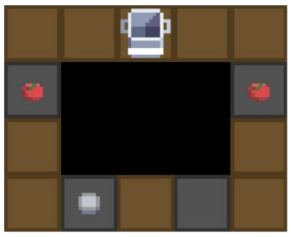

Figure 1: Cramped room, from the Overcooked environment, illustration taken from [3]. Shows the space agents can move in in black, and the counters, tomatoes, onions, etc. Sparse reward is received when an onion soup is delivered, and dense reward is given for things like picking up a plate, chopping onions, etc. An agent can either be placed on the left or the right at the beginning of each game.

3.2 Overcooked environment

The Overcooked environment, taken from [3], is used as the basis for our tests. In this environment, two players are tasked to prepare and serve as many onion soups as possible within a time limit. This means there is a chain of events that must happen, and that can sometimes be done in parallel to increase throughput of onion soups. Often, it is optimal to work as a team with your partner, and in the specific layouts included with the base environment, movement often has to be coordinated with the other agent so as to not block each other's paths. The environment is also classified game-theoretically by [3] as a

common-payoff game, as opposed to zero-sum games like Starcraft. It is also important to note that the environment includes sparse and dense rewards, which are used for the reward shaping technique [17] to bootstrap the training of an agent. In its default implementation, the environment performs reward shaping by annealing between the two kinds of reward depending on total step count. Finally, there are multiple layouts available for the players, and they are designed to test for different skills in the player's cooperative ability. For our interests, the simplest room, called "cramped" is the one chosen (illustrated in figure 1). In terms of visibility, both agents are able to see the whole map at any time, so a boolean mask is not used for observations.

3.3 Fictional Co-Play

In order to answer our research question we have implemented a simplified, yet complete, version of the FCP (Fictional Co-Play, [8]) method as described in the paper. To do this, a list of requirements for the FCP method was written representing the essence of the method:

1 multiple self-play (SP) agents must be trained

2 during their training, those SP agents must save an arbitrary number of checkpoints

3 a final-agent (FA) must be initialized

4 the FA agent must be trained with agents picked randomly among the SP agents and their checkpoints

5 the FA agent must be evaluated by measuring its average reward when playing with a Behavioral Cloning (BC) agent.

Besides the FCP algorithm itself, it is worth mentioning the origins of the method. The multiple self-play agents mean that this is a form of population-based training, and the inclusion of previous checkpoints is also an established technique for reinforcement learning [18].

4 Methodology

4.1 Development Process

In order to develop the FCP agent, a time-efficient approach was taken. Rather than implementing the whole agent ourselves, we decided to use the publicly available code [19] of the Overcooked paper as a boilerplate, and to take their PPO agent (which is the Baselines [20] implementation) and modify it to implement the FCP method.

4.1.1 Limitations

In many ways, our implementation differs from the one presented in [8]. The different ways in which we were limited are the following:

Computing power Because this paper was written by a single person, and all experiments as well, and due to limited access to computing resources, a single Nvidia GTX 1060 graphics card with 6GB of memory was used for training. This means we had to scale down the training parameters compared to [8], instead aiming to achieve as many training steps as [19].

Implementation time As the amount of time available to write this paper amounted to two months, the implementation of the FCP method was limited as well. While all the requirements in section 2.1 have been implemented, some corners have been cut. Namely, SP agent checkpointing is not based on average reward, but on training step number, and the final agent trains against a pre-defined number of training partners once the SP agents are trained. Overall, we believe a good attempt was made to answer the research question.

4.2 Experimental setup

Agent setup As mentioned earlier, the FCP agent is based on the PPO self-play agent implemented in [19]. In order to take advantage of the hyperparameter tuning performed by the authors, and because they showed adequate performance during testing, most of the hyperparameters of their default PPO agent (in the main ppo file) have been kept intact. This includes settings such as reward shaping values, batch size and learning rate. Then, to accommodate for the FCP agent training process, four more hyperparameters have been added. They are explained in detail in the Appendix, section A.2. For the rest of the parameters, we will detail them here. Finally, it is important to mention that the agent's neural networks do not include an LSTM or Transformer network, instead they are made of dense neural networks (DNN) and convolutional neural network (CNN) layers.

parameter choice One important metric is the total number of steps of training for the whole process. As we are trying to compare our results to the ones from [3], and because training algorithms show further improvement when trained for longer [3] [8], we have chosen a number of total steps similar to their implementation of a population-based training agent (found to be 24 million), which was 25 million. Our results therefore test both the adaptivity and training-step efficiency of the FCP method when compared to the other methods. This number was used to design multiple test FCP runs that varied in population size. To keep the same number of total steps, the number of steps per-final agent and per-SP agent were adapted. Another important point is that final agents train for a larger amount of total training steps than population agents, and this was chosen in order to give the final agent more time to train with the large population, as well as to avoid having the SP agents converge too much towards each other. Finally, the evaluation parameters are also the same as the ones found in [19] at path *human_aware_rl/NeurIPS Experiments and Visualizations.ipynb*, which means that the agent is evaluated for 40 rounds per seed with the human proxy.

Data collection There are two kinds of data collected: pre-evaluation data, and evaluation data. Before evaluation, the dense reward and sparse reward are used as a baseline metric for training progress, for both the SP agents and the final agent. This is because average reward shows that the agent is learning, and sparse reward is the actual score of the game, depending only on the number of delivered dishes. What's important in order to answer the question of whether the agent can adapt to human beings then, is its average sparse reward when paired with the human proxy (behavioral cloning agent). Finally, we also collect evaluation data on the FCP agent playing against itself. This allows us to compare that score as a "max performance" so as to see how much reward is lost to inefficiencies in collaboration between the FCP agent and the proxy human.

5 Results

The first step to understanding the performance of our agent is to see it successfully learn to play. In figure 2 we observe the training curves of the final FCP agent as an average of three seeds, along with their 75% confidence interval. From this we can see that the agent does indeed learn, and that its evolution becomes rather unstable after it has reached a reward value near the top. Given that this happens across multiple seeds, we hypothesise that the stability might be affected by the hyperparameters. For example, the learning rate which works well for the initial high-growth phase might be too high for the kind of fine-tuning that might be happening near the end, causing the agent to overcorrect itself. We also notice that the two curves look very similar, which can be attributed to the fact that in the particular environment layout our agents train in, the actions that yield dense reward most often also lead to an agent delivering a dish. And near the end of the training, the sparse rewards become higher than the dense rewards, possibly indicating that the agent has evolved into a more efficient strategy beyond performing those actions that yield dense rewards.

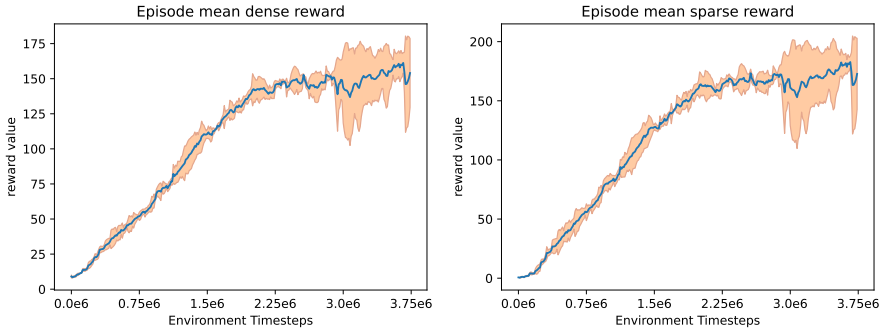

Figure 2: Plots showing the recorded data during training of the final agent in the FCP training process, averaged over three different seeds, along with their 75% confidence interval. On the left is the mean dense reward per epoch. Mean total sparse reward on the right is the reward received by the agent throughout the epoch, during the games. We can see that the agent spends the first half of the time converging, and becomes unstable after that.

The second step is that the agent must demonstrate its ability to generalize. In figure 3 we see the performance of the SP, BC, PBT and PPO_{bc} agents from [3] compared to our own, with standard error. All FCP results are an average of three seeds, and two configurations (start on left and right side of the map). Compared to a minimum 2X improvement in number of deliveries found in [8] (figure 5), which would correspond to double the reward amount, we also see an improvement from our agent, however it is quite smaller than expected. Overall, the FCP agent had a greater average score when playing with H_{proxy} than all other agents except for PPO_{bc}. We believe this to be the case because that agent was able to train with a behavioral cloning agent that is very similar to the human proxy. Also of note is that the FCP agent shows greater error across seeds than the other methods. This reflects what was seen in the previous figure. To show how much an agent can vary, but also the potential of this FCP training method, we also include FCP_{best}, the best score of the FCP agent among all the evaluations.

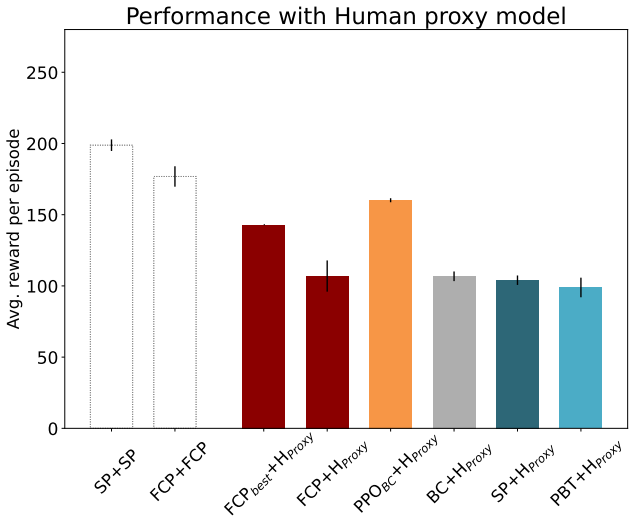

Figure 3: Agents compared by their average scores and standard error. All datas except the ones involving FCP are taken from [3]. Two of the bars (FCP+FCP and SP+SP) are agents playing against themselves (this is interesting because SP+SP usually gets the best score in the game). The other plots are agents paired with the human proxy (behavioral cloning) agent. For further explanation on the agent types, see section 3. FCP agent is averaged over three different seeds and all agents averaged across two different setups (start on the left, or start on the right). FCP_{best} is the only exception, it is the highest score across all evaluation rounds of $FCP + H_{proxy}$.

In order to investigate the potential of our FCP implementation, we decided to investigate population size as the most likely candidate hyperparameter for improvement. Figure 4 shows the average reward for three different runs. Each of those runs required around 25 million steps of training in total, per seed, which also implies that the 4-agent run and 16-agent run have the final agent train against more, and less skilled self-play agents respectively, compared to the 8-agent run. This shows potential for a trade-off between better skilled population agents and a larger variety of agents. Further, we can hypothesise with this graph that there might be a quadratic relationship between the population size and the average reward. If this is true, we could also hypothesise a trade-off happening between FCP+FCP and FCP+H_{proxy} performance. Further testing would be needed in order to test this hypothesis. Finally, we can explain that the best-generalising agent is from the 8-agent population because this configuration has the self-play agents train for 2.5 million steps. This is an interesting number because it's close to the point between where the agent finishes its initial high-growth learning phase, and the slower-growth phase (see figure A.1 in the Appendix). This could indicate a detectable sweet-spot for the population agents that could be used in other configurations.

Lastly, in order to see how the performance of the final FCP agent evolves throughout its training, we have done a special run, where we decided to save checkpoints of this final agent during its training, and to evaluate each checkpoint individually with both the human proxy and itself. Figure 5 shows the results of this, averaged over three seeds. We can clearly see

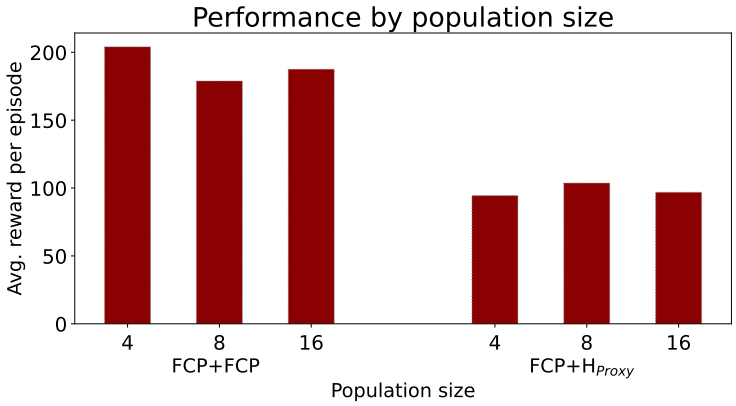

Figure 4: A comparison between different FCP runs, evaluated against the human proxy and against itself. Each run is the average of three seeds, and two configurations (start on left and start on right of the game environment). The varying number is the size of the self-play agent population that the FCP agent trains against. For each run, the parameters are adjusted so the total number of steps for the whole run is around 25 million.

that the FCP+FCP curve on the left follows the training curve from figure 2, and the more interesting one is on the right, as it shows something that did not seem apparent from the training curve alone; it seems like in all three seeds, the agent quickly finds a score of 80 at around a million training steps, and after that the variance increases and stays consistent all the way to the end of the training. This does not correspond with the high variance found near the end of figure 2, and indeed all we see near the end of this plot is that the average score slowly increases. This leads us to believe that there might be more potential in training the final agent for even more steps in order to see how long it would take for the final agent to exhaust all the performance it can gain from training with its population.

5.1 Discussion

The results we have found are surprising because of their stark contrast with the results shown in the FCP paper [8]. Our agent shows a small fraction of the improvement found by them. In figure 4 we hypothesised that there might be a trade-off between FCP+FCP and FCP+H_{proxy} performance, and we think this could be thought of as a trade-off between generalisation ability and specialisation. This generalisation ability would also seem to be quite expensive, given that in figure 3 we see that the FCP agent is only slightly better than the SP agent. The cost of this generalisation is then evident since the SP agent in this figure only took 6 million steps to train, whereas the FCP agent took 25 million.

It is also interesting to think about why FCP shows similar evaluation scores as SP and PBT agents. Indeed, one of the expectations from [3] was that the SP agent's score would suffer drastically when paired with the human due to it being trained to be optimized to play with itself only, and this was confirmed in their results. Perhaps unsurprisingly, the PBT agent suffered the same fate, going from an even better score than the SP agent when playing with itself, to a slightly worse one when paired with the human proxy. So with the FCP agent supposedly benefiting and learning from a diverse set of training partners, its ability to generalise should have given it a distinct advantage when facing the human proxy.

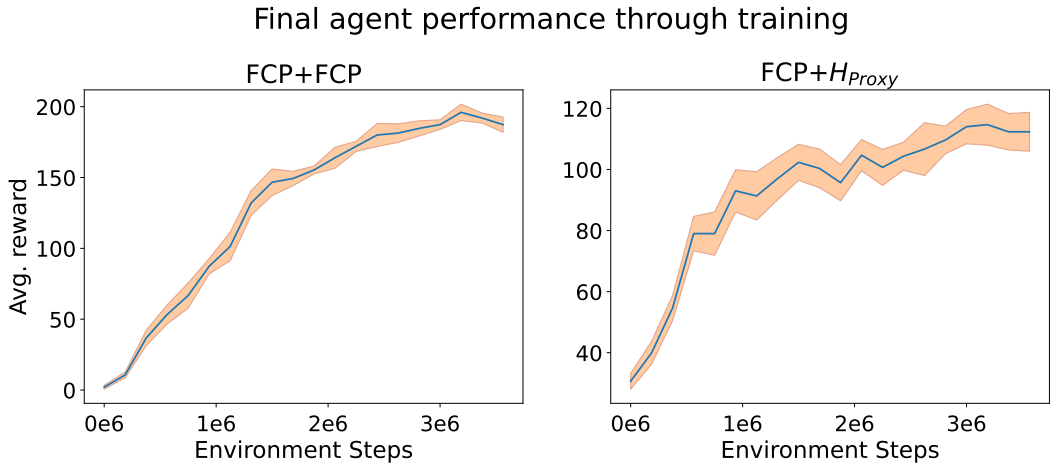

Figure 5: Plots showing evaluation results of the FCP final agent at twenty checkpoints during its training. In other words, twenty checkpoints were made during the training of the final agent, and this plot shows the evaluation results of each, averaged over three seeds, along with their 75% confidence intervals, and evaluated over ten rounds of training both with itself (right) and with the human proxy (left).

This could be because of a lack of diversity in the SP agents. To test this hypothesis would be very interesting, especially to see how much of the cross-seed variation can be explained by the diversity of its population. And of course, it is likely that there are oversights in our implementation, however it seems unlikely that this would cause such a drastically different score than expected, given that [5] (see figure 4) has achieved similar results in their own experimentations using the FCP method.

In the end, it is also possible that the results from the original FCP paper [8] would only show up when the agents are trained at orders of magnitude more training steps. One possible indicator of this is figure 5. The FCP+H_{proxy} curve in particular could suggest that the final agent still has room to grow its generalisation ability for many more steps. This is also interesting because technically, increasing the number of steps in the final agent does not cost a lot of total training steps when compared to increasing the number of steps per population agent, so if further training yields higher generalisation, this would mean an increase in performance at the same cost as a self-play agent (which is comparatively the step-wise cheapest approach). This is yet another place where more investigation can be done.

6 Responsible Research

6.1 Ethics

In following the principles of ethical research, we have made sure to include proper references in the IEEE standard, to show links to the open source code we have used, and since we have not run any tests on living beings, we can guarantee that no living being was at risk during our tests. Finally, because this paper was written using relatively low-cost hardware, the training did not spend much energy and therefore did not contribute to a sizeable impact on the environment like other, larger models would.

6.2 Reproducibility

To ensure reproducibility of our results, we make our code available as a Docker image, and we strongly encourage any interested reader to use it: https://hub.docker.com/r/nataxcan/fcp (latest tag). Additionally, we have made sure that the following points are true of this paper:

- All code will be available on a publicly-hosted Github repository, and a public Docker container already exists that allows one to reproduce all our results.
- The methodology section was written so that a developer is able to emulate the development methodology of this paper almost completely.
- The publicly-available Github repository will contain a tutorial explaning how to run the code to reproduce the results. Which files to run, in which order, and which dependencies to install and how will be included.
- The hardware used to train the FCP agent was mentioned so that anyone could reproduce the hardware configuration used for this paper.
- The parameters were chosen in order to be able to reproduce the results of this paper within a maximum of 24 hours, given similar training hardware.
- Given current (as of 30th of May 2022) prices for online-available training hardware, we estimate the training of this agent can be realized on third-party hardware for about three euros. This means that no great expenses are required in order to replicate the results of this paper.

All of this is motivated by our goal to conduct ethical, responsible research for the scientific community, and we believe that it ensures this work is reproducible by a very large number of people, with the only large limiting factor being programming literacy.

7 Conclusions and future work

Research questions This paper was started by the question: "Can FCP create agents that adapt well to human beings?". In order to answer this question, we have implemented the FCP method as best as we could, and run as much testing as was possible. However, this was limited by the amount of time available to write this paper, and by the compute available to run training of this agent. Nevertheless, our results showed that an FCP agent on average shows a small improvement over the agents created in [3] with high cross-seed variance, which is consistent with the results found by [5]. This showed that the FCP method

trains a PPO agent to be at least as adaptable as the rest, more so than the Self-Play and the Population-based training methods, and that it has the potential of scoring very high when playing with the human proxy. In the end, we think that this method shows promise in terms of performance, and that we estimate it to be better able to adapt to playing with human beings.

Future work For each shortcoming in this work, there is an opportunity for improvement. With more computing power, it would be interesting to see how far this method extends to more complicated domains. With more time, other aspects of FCP agents could be tested further, such as whether the teamplay ability of the agent in Overcooked can be transfer-learned to another game, thereby showing faster training speed. To extend this method however, we propose the combination of the FCP method and the diversity-maximisation approach in [5]. As a reminder, in their technique, each agent in a population is partly rewarded for how *diverse* they are compared to the rest of the population, according to their metric. And since the working principle of FCP is that the final agent learns to adapt to a diverse set of training partners, we hypothesise that implementing the "diversity reward" in the self-play population will only push that effect further, and thereby increase the performance of the final agent, seemingly at no extra cost in training time.

We also propose an adaptive learning rate that adjusts based on how far the final agent is in training, in order to deal with the instability that we see in the agent's training curves. Finally, we think the biggest potential improvement would be to add a time-sensitive component to the neural networks of the agents, such as an LSTM, so that the final agent could learn to adapt to another agent (as a skill), rather than trying to optimize a general strategy on a per-state basis.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," Tech. Rep. arXiv:1312.5602, arXiv, Dec. 2013. arXiv:1312.5602 [cs] type: article.

[2] N. Schurr, J. Marecki, M. Tambe, and P. Scerri, "Towards flexible coordination of human-agent teams," *Multiagent and Grid Systems*, vol. 1, pp. 3–16, July 2005.

[3] M. Carroll, R. Shah, M. K. Ho, T. L. Griffiths, S. A. Seshia, P. Abbeel, and A. Dragan, "On the Utility of Learning about Humans for Human-AI Coordination," Jan. 2020. Number: arXiv:1910.05789 arXiv:1910.05789 [cs, stat].

[4] G. T. Games, "Overcooked," 2016.

[5] R. Zhao, J. Song, Y. Yuan, H. Haifeng, Y. Gao, Y. Wu, Z. Sun, and Y. Wei, "Maximum Entropy Population-Based Training for Zero-Shot Human-AI Coordination," May 2022. Number: arXiv:2112.11701 arXiv:2112.11701 [cs].

[6] K. R. McKee, J. Z. Leibo, C. Beattie, and R. Everett, "Quantifying the effects of environment and population diversity in multi-agent reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 36, p. 21, Apr. 2022.

[7] J. G. Ribeiro, C. Martinho, A. Sardinha, and F. S. Melo, "Assisting Unknown Teammates in Unknown Tasks: Ad Hoc Teamwork under Partial Observability," Jan. 2022. Number: arXiv:2201.03538 arXiv:2201.03538 [cs].

[8] D. J. Strouse, K. R. McKee, M. Botvinick, E. Hughes, and R. Everett, "Collaborating with Humans without Human Data," Jan. 2022. Number: arXiv:2110.08176 arXiv:2110.08176 [cs].

[9] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine, "Diversity is All You Need: Learning Skills without a Reward Function," Oct. 2018. Number: arXiv:1802.06070 arXiv:1802.06070 [cs].

[10] A. Lupu, B. Cui, H. Hu, and J. Foerster, "Trajectory Diversity for Zero-Shot Coordination," in *Proceedings of the 38th International Conference on Machine Learning*, pp. 7204–7213, PMLR, July 2021. ISSN: 2640-3498.

[11] J. Lin, "Divergence measures based on the Shannon entropy," *IEEE Transactions on Information Theory*, vol. 37, pp. 145–151, Jan. 1991. Conference Name: IEEE Transactions on Information Theory.

[12] H. Hu, A. Lerer, A. Peysakhovich, and J. Foerster, ""Other-Play" for Zero-Shot Coordination," May 2021. Number: arXiv:2003.02979 arXiv:2003.02979 [cs].

[13] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems, Artificial Intelligence, Multiagent Systems, and Cognitive Robotics, Cham: Springer International Publishing : Imprint: Springer, 1st ed. 2016 ed., 2016.

[14] D. A. Pomerleau, "Efficient Training of Artificial Neural Networks for Autonomous Navigation," *Neural Computation*, vol. 3, pp. 88–97, Feb. 1991.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017. Number: arXiv:1707.06347 arXiv:1707.06347 [cs].

[16] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population Based Training of Neural Networks," Nov. 2017. Number: arXiv:1711.09846 arXiv:1711.09846 [cs].

[17] M. D. Buhmann, P. Melville, V. Sindhwani, N. Quadrianto, W. L. Buntine, L. Torgo, X. Zhang, P. Stone, J. Struyf, H. Blockeel, K. Driessens, R. Miikkulainen, E. Wiewiora, J. Peters, R. Tedrake, N. Roy, J. Morimoto, P. A. Flach, and J. Fürnkranz, "Reward Shaping," in *Encyclopedia of Machine Learning* (C. Sammut and G. I. Webb, eds.), pp. 863–865, Boston, MA: Springer US, 2011.

[18] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent Complexity via Multi-Agent Competition," Oct. 2017.

[19] M. Carroll, R. Shah, M. K. Ho, T. L. Griffiths, S. A. Seshia, P. Abbeel, and A. Dragan, "Human-Aware Reinforcement Learning," Nov. 2020.

[20] OpenAI, "Baselines," June 2022. original-date: 2017-05-24T01:58:13Z.

A Appendix

A.1 SP training curve

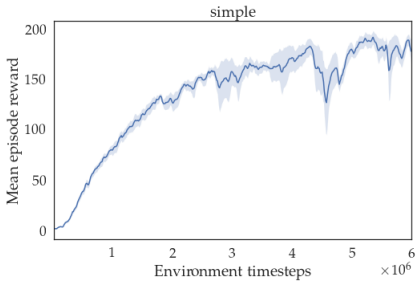

Figure 6: Training curve for a PPO agent trained using self-play. This was the justification for choosing the number of training steps for the FCP agent's self-play agent population. Found in [3] at path *human_ aware_ rl/NeurIPS Experiments and Visualizations.ipynb*.

A.2 FCP-specific parameters

An FCP run consists of:

type	value
SP agent population size	8
n checkpoints per SP agent	2
n training steps, SP agent	2.5 million
n training steps, final agent	3.75 million
n training sessions, final agent	60

Table 1: Agent-specific parameters for an FCP run and the values chosen for testing in this paper

Where the *SP agent population size* is the number of self-play agents the final agent will train with. The *n checkpoints per SP agent* is the number of checkpoints that will be saved during training of a self-play agent. For example, if an agent trains for 900 steps in total with two checkpoints, they will be saved at 300 and 600 steps. The two *n training steps* parameters are for each SP agent, and for the final agent respectively. Finally, the number of training sessions is the number of times the final agent will pick an agent at random from the population (of SP agents and their past checkpoints) during the course of its training. The SP population size was chosen to be among the numbers tested by [8] (found on table 2 of the appendix), by running test runs for 4, 8 and 16 agents while keeping the same number of total training steps. The number of checkpoints was chosen based on the FCP paper [8], which used one checkpoint near the beginning of the run, and one once it reaches the middle of its maximum dense reward during training. The training steps for the SP agents were chosen so as to have a total number of steps equal to 25 million. As for the final

agent's steps, it was set to 1.5 times that of the SP agents so that it has more time to spend adapting to its population agents. The number of sessions is set to 60 because of a series of test runs over a single seed that indicated this might be the optimal number. If this paper had a longer timeframe, more runs would have been made.