Cooperative AI for Overcooked

Multi-Agent RL with Population-Based Training

Ivan Nestorov

Supervisor(s): Frans Oliehoek, Robert Loftin

EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 25, 2023

Name of the student: Ivan Nestorov
Final project course: CSE3000 Research Project
Thesis committee: Frans Oliehoek, Robert Loftin, Klaus Hildebrandt

Abstract

In ad-hoc cooperative environments, the usage of artificial intelligence to take supportive roles and work in collaboration with humans has proven to be of great benefit. The objective of this research is to evaluate the use of population-based training for reinforcement learning agents in a simplified version of the multiplayer game - Overcooked. The method used to answer that question involves evaluating the performance of the agents when paired with a human proxy and their learning curves on different layouts. Based on the employed method, it was concluded that both PBT and other self-play agents display notable underperformance when compared to human proxies and agents trained using human data. Moreover, while the inclusion of mutated agents enhanced sample efficiency in layouts with minimal collision risks, its effect on the final performance of PBT in those layouts was negligible. However, this approach managed to improve performance in layouts where collisions were the primary limiting factor.

1 Introduction

With the swift progress of technology, innovative approaches are continuously emerging to tackle mundane and repetitive human-operated tasks, among one of these approaches is the adoption of Reinforcement Learning (RL) based Artificial Intelligence (AI). In various industries, such as automotive and healthcare, there is a drive to develop fully autonomous vehicles and utilize RL agents for detecting and localizing abnormalities. Our focus lies in the field of multiplayer games, where we can explore in a risk-free environment the benefits of using human-AI cooperation. By applying reinforcement learning algorithms in video games, we aim to evaluate their ability to adapt and learn from the environment, make real-time decisions, and cooperate with other players, providing us with a clear benchmark for human-AI cooperation.

Overcooked is an example of one such game. Overcooked [1] is a collaborative game, in which a player is able to pair up with three other players. The main goal of the game is in collaboration to prepare and serve different dishes to customers. The environment, which will be used for this research project is a simplified [2] version of the original game, but will still retain the same multi-agent mechanics. This is done to reduce the computational intensity. This research aims to assess the benefits of using RL-based agents in the Overcooked environment, using Population-Based Training (PBT).

PBT is a training method for machine learning models. It uses the evolutionary approach to optimize the neural network. PBT can be helpful in the ad-hoc cooperation environment, which Overcooked provides. PBT allows agents to adapt to changes in the environment and learn from the experience of other agents. The algorithm, which will be used to update the policies in the PBT is Proximal Policy Optimization(PPO)[3].

This research will act as a form of validation of the results from the previous research on the utility of using PBT with PPO for training RL agents in the Overcooked environment[4]. The credibility of the results will be increased by reproducing the experiments[5] of the aforementioned research. Additionally, our objective is to explore various methods for improving the performance of Population-Based Training (PBT) To accomplish this, the research aims to address the following inquiries:

- How does the use of population-based training affect the performance of Multi-agent reinforcement learning (MARL) algorithms?
- What changes can be made to the PBT to improve the agent's performance when paired with a human player?

2 Background

2.1 Reinforcement Learning

Reinforcement learning is a technique employed in machine learning where an agent learns how to behave in an environment by performing certain actions and receiving feedback in the form of rewards. It is designed to optimize decision-making processes by training policies that are most likely to increase the overall reward, based on the status or state of the system.

In our research, we characterize the reward function using two types of rewards: sparse and dense. Sparse rewards are tied to long-term actions where rewards are distributed infrequently. For example, in the context of the Overcooked environment, a sparse reward may be given when the chef completes substantial tasks like fully preparing or serving a dish. This kind of reward doesn't necessarily provide feedback on every single step of the process, but it places value on the achievement of long-term, substantial goals.

Conversely, dense rewards are associated with short-term actions, where rewards are granted for a myriad of tasks. For instance, if the agent avoids collisions or interacts appropriately with the environment, such as picking up an onion or placing it into a pot, it could receive a reward. In the Overcooked environment, collisions occur when objects or players come into contact with each other, thus halting game progress until the collision is resolved. This can include collisions between chefs or between chefs and the environment (such as walls or countertops). The main advantage of dense rewards is that they provide more immediate and granular feedback about the optimality of the agent's trajectory. They help the agent understand which actions are beneficial in the short term, thereby encouraging the replication of such actions.

By defining our reward function in this manner, combining both sparse and dense rewards, our aim is to create a reinforcement learning system that encourages the agent not only to perform tasks that yield immediate rewards but also to learn and execute actions that lead to long-term success. The combination of these two reward structures allows for a more balanced learning experience, fostering an agent that can effectively operate within the bounds of both immediate and long-term goals.

2.2 Overcooked environment

To assess the efficacy of reinforcement learning algorithms in cooperative settings, we will use a simplified version of the popular multiplayer game Overcooked, referred to as Overcooked-AI (see Figure 1).

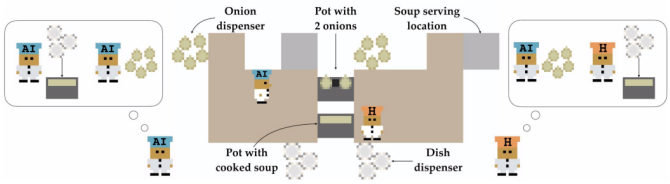


Figure 1: Overview of the simplified Overcooked environment used for this research. Image is taken from[4].

It's important to clarify that Overcooked-AI is not identical to the commercially available game, Overcooked. Instead, Overcooked-AI is a version specifically designed for reinforcement learning research. It simplifies the original game environment while maintaining the same multi-agent mechanics and cooperative interactions that are pivotal for the evaluation of AI performance. Similarly to the original game, players control chefs around the kitchen to prepare and serve dishes. Players can choose one of six different actions: up, down, left, right, noop, and interact. The interaction depends on the tile that the player is facing. In the original game, there are different recipes for soups, pizzas, and other dishes. However, in our environment, chefs can prepare only soups of onions. In every layout, there is an unlimited supply of onions, located in one or two places, depending on the layout, plate dispenser, and place where ready dishes are served. The reinforcement learning agents should be able to navigate through the layout and work in collaboration with other agents to maximize the served dishes. One cooking procedure takes the form of a player picking up an onion from the dispenser, putting it into a pot, repeating this process until there are three onions in the pot, then waiting twenty timesteps for the soup to be cooked, then putting it in a plate, taken from the dispenser for plates and serving the dish. This awards all players twenty points.

2.3 Population-Based Training

Population-based training (PBT) is an automated method for hyperparameters optimization. It is inspired by evolutionary algorithms, which combine the concept of training models with the principles of natural selection and genetic algorithms. Please refer to Appendix B for a pseudo-code implementation of PBT.

PBT begins with a population of models with randomly initialized hyperparameters. These hyperparameters define the characteristics of the model, such as learning rate, batch size, network architecture, or any other tunable parameters. Each model is trained for a short period, then evaluated by using dense rewards. The better performing models are selected and their hyperparameters are shared with the under-performing agents, imitating "survival of the fittest" behaviour. Offspring models are created by mutating the hyperparameters of the top models, introducing genetic variations. By mutating the hyperparameters, the population of models expands its exploration of the hyperparameter space, enabling the discovery of potentially better configurations. This process continues until an optimal solution is found or a predetermined number of iterations is met. In this research, the algorithm that will be used to update policies will be Proximal Policy Optimization (PPO).

2.4 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is an on-policies reinforcement learning (RL) algorithm, it aims to address the challenges faced by traditional policy gradient methods, namely, instability and inefficiency during training. The way this is ensured is by preventing high variance in the updates from the original policy. The implementation of this method relies on the actor-critic model, which uses two different neural networks, one that handles actions and the second one the rewards. The actions of each agent are controlled by a single trained policy.

2.5 Human Models

The human models used in the experiments are behavioral cloning models, based on already gathered from previous research - human data. Behavioral cloning is a technique that involves acquiring a policy by observing and imitating an expert's desired behavior. It accomplishes this by employing supervised learning to establish a relationship between states and corresponding actions. The implementation of BC-based agent models in this research consists of three main components: representing the state as input, generating a probability distribution of actions as output, and employing the cross-entropy loss function for training purposes.

Trajectories from the human-human play are collected for each layout, after which they are separated into two single-agent trajectory sets. Through behavioral cloning, human models have been trained for each subset and layout. The BC-based model is the version of training models and the second model is used to evaluate the final performance by simulating a human player.

3 Methodology

3.1 Performance Measuring

In evaluating the effectiveness of Population-Based Training (PBT) in a collaborative context, we prioritize two core metrics: the final performance when teamed with a human proxy, and learning curves. Although both metrics are crucial, our main focus will be on the former metric, given the unique collaborative setting.

Our primary evaluation is based on the final performance of the PBT-trained agent when teamed with a simulated human player. In essence, we are interested in the mean reward per episode when the agent, trained via PBT, is paired with the human proxy. For a robust analysis, this is compared against the mean rewards per episode of various other agents, all paired with the same human proxy. This approach allows for a comparative assessment of the final performance

of our PBT agent when working alongside a human counterpart. Notably, this evaluation provides a crucial indication of PBT's ability to generalize to unfamiliar agents - in this case, a human player. This is especially important as we want to measure PBT's ability to interact effectively when paired with human counterparts.

While our emphasis is on the agent's final performance with a human model, we also evaluate the learning curves to gain insights into the sample efficiency of PBT. Learning curves, represented as average rewards per episode across the environment, illustrate the amount of data and time PBT requires to optimize its policy. These rewards are computed based on the sparse rewards of the chosen agent when paired with all other agents in the population, including itself. By doing so, we can determine how many environmental steps are required for the agent to achieve peak performance. Despite the value of learning curves in assessing sample efficiency, we emphasize that our primary focus lies on the PBT agent's ability to partner effectively with human proxies.

3.2 Experimental Variations

To address our second research question on enhancing PBT performance, we investigate two main modifications. Our main focus is to increase the noise in the population. By increasing the noise, or in other words, by making these mutations more variable, we aim to create a population of agents that are more diverse in their strategies and responses. We hypothesize that this could lead to agents that are better at generalizing toward human players, who are inherently unpredictable and diverse in their behaviors. Inspired by this proposition, we have designed our experimental variations to involve the mutation of selected individuals from the initial population, aiming to promote diversity within the population. In this approach, we will employ a larger number of mutation factors and a higher mutation probability, enhancing the extent and probability of genetic mutations. Importantly, these mutations will be conducted only once at the beginning of the training, ensuring simplicity and cost-effectiveness. Further investigations will be conducted to examine the effects of varying population sizes on the training process.

3.3 Experimental Layout

Due to time constraints, concerning the computational costs, two layouts will be used, namely 'Asymmetric Advantages' (see Figure 2) and 'Coordination Ring' (see Figure 3). The agents trained for each of these layouts are using four different seeds in order to minimize the noise. The choice of these layouts was based on their ability to display the agents' capability to maximize their strengths 'Asymmetric Advantages' and their ability to coordinate for example to avoid collisions 'Coordination ring'.

3.4 Baseline Agents

To gauge the effectiveness of PBT, compared to different approaches, the final performance of PBT is assessed alongside that of various baseline agents when paired with a human proxy. The comparison involves different agents, which are

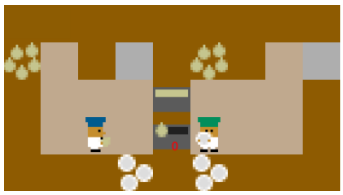


Figure 2: Asymmetric Advantages layout. Image is taken from [4].

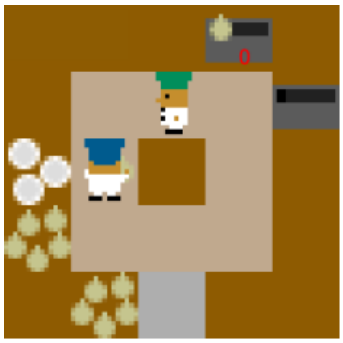


Figure 3: Coordination Ring layout. Image is taken from [4].

listed below. The agents in the evaluation are trained using the same parameters as outlined in the previous research.

- H_{Proxy}: Human model trained through the behavioral cloning approach to simulate the human player.
- BC: An imitation agent, based on a distinct dataset, which was used to train the human proxy.
- SP: PPO agent, which was trained only by the self-play method.
- PPO_{BC}: PPO agent trained with the BC of a human model.
- $PPO_{H_{Proxy}}$ PPO agent trained with human proxy. Gives an overview of the peak performance, which could be achieved by PPO given full access to the human proxy.

4 Experimental Setup and Results

4.1 Experimental Setup

The results shown below are products of the conducted experiments. They are divided into two parts, each one focusing on a single research question. The first results focus on evaluating PBT agents trained based on the experimental setup of previous research[4] in an attempt to reproduce and validate their findings. The second results will focus on analyzing the effects of a larger number of mutation factors and a higher mutation probability in PBT on the same parameters from [4]. Additionally, we have increased the population size from 3 to 4 agents to explore the potential advantages of a larger population.

4.2 Results After Reproduction of Experiments

Learning Curves

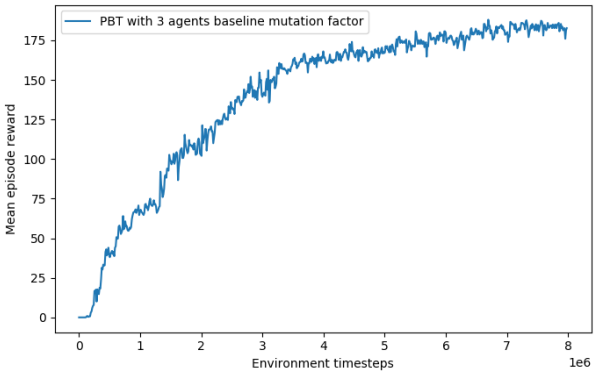
To evaluate the sample efficiency of training the PBT agent, the learning curve is graphed and analyzed for both experi-

ment layouts: 'Asymmetric Advantages' and 'Coordination Ring'. Please refer to Figure 4 for the graphical representation.

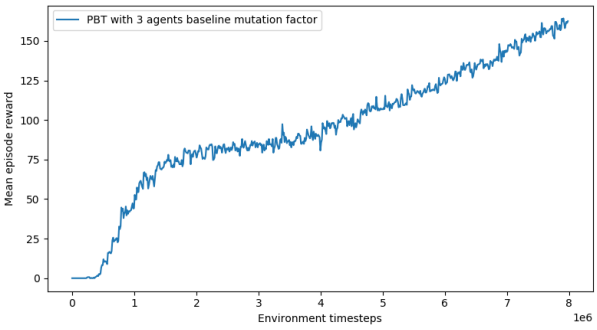
In both layouts, the trained agents have a similar final performance of 180 for 'Asymmetric Advantages' and 170 for 'Coordination Ring'. However, the agents trained on the 'Coordination Ring' layout, require 3x10^6 more timesteps to reach the same maxima in performance. These findings indicate that the agent trained in the 'Coordination Ring' layout exhibits lower sample efficiency, requiring more data for policy optimization.

In theory, the difference in the number of timesteps during training could be attributed to the level of coordination required and the likelihood of collision within a given layout. When comparing 'Asymmetric Advantages' and 'Coordination Ring', the first layout eliminates the possibility of agent collision and allows each agent to complete tasks independently. Consequently, agents in the 'Asymmetric Advantages' may attain higher rewards early in training since they can serve dishes without encountering issues caused by colliding with other agents. However, these rewards achieved may not truly reflect optimized performance, as full coordination with the other agent is not emphasized.

On the other hand, in 'Coordination Ring', agents must coordinate with their teammates from the beginning to avoid collision and maximize rewards. The necessity of coordination to prevent collisions leads to the agent reaching peak performance in more timesteps than layouts that do not mandate coordination, such as 'Asymmetric Advantages'. Nevertheless, further research is necessary to validate this theory.



(a) Asymmetric Advantages



(b) Coordination Ring

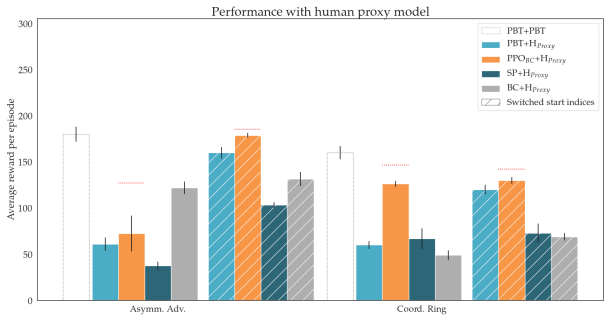
Figure 4: Learning curves of a PBT agent in the 'Asymmetric Advantages' and 'Coordination Ring', displaying the average sparse reward over 3 seeds per episode during training over 400 timesteps.

Final Performance

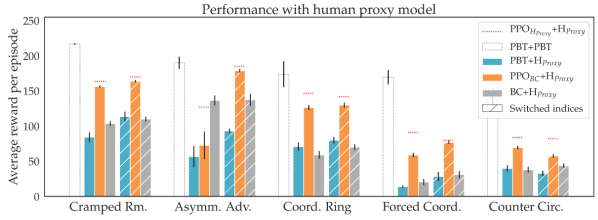
To assess the overall performance of PBT, each type of agent is paired with a human proxy, and the average reward per episode for each pair is graphed as a bar chart in Figure 5a. These experimental results from Figure 5a are subsequently compared and validated against the results presented by [4] in Figure 5b.

The performance of PBT is initially examined in relation to its optimal performance. In Figure 5a, when paired with a human player the mean reward per layout was 65 and 70, respectively, PBT shows a significant decline compared to when it plays against itself (mean reward: 180, 170). A similar observation can be made from Figure 5b. However, since all other models also perform worse than their optimal performance, the dip in PBT's performance could be partially attributed to the suboptimal performance of the human proxy.

Next, PBT is analyzed in comparison to the baseline agents. As supported by previous research[4], PBT outperforms self-play, which has a mean reward of 68 and 57 per layout, respectively, as shown in Figure 5a. In contrast, the PPO_{BC} agent trained with the human BC model achieves higher performance with the human proxy (mean



(a) Performance results produced from self-conducted experiments.[4]



(b) Performance results produced from previous research[4]

Figure 5: Performance results of the PBT agent when paired with the human proxy in comparison to the performance of other agents matched with the same proxy. For each layout, performance is given by the average sparse reward per episode during training over 400 timesteps. Striped bars indicate results when the starting locations of the agents have been swapped.
Note: Horizontal red dotted lines are $PPO_{H_{Proxy}} + H_{Proxy}$

reward: 119, 143) than PBT. Additionally, consistent with the results presented in Figure 5b, the performance of the BC agent (mean reward: 127, 59) falls between that of the PBT and PPO_{BC} agents, as it is a relatively simple model that only mimics a specific subset of the collected human data. The performance gap between the optimal performance of $PPO_{H_{Proxy}}$ and PPO_{BC} highlights the differences between the individual human models, H_{Proxy}, and BC.

In summary, PBT paired with the human proxy shows improvement over the self-play method but underperforms compared to agents trained with human data. This indicates that, to some extent, the population of independently-initialized policies in PBT enhances generalization capabilities towards a human player compared to self-play. However, it is understandable that PBT falls short compared to other baseline agents since it does not learn its policy directly from human data.

4.3 Results After Variation of Experiments

Figure 6 illustrates the learning curves of the selected layouts, which were obtained by incorporating custom mutation factors (refer to Appendix A) into the Population-Based Training process. In addition, the figure displays learning curves for various population sizes, allowing for a comprehensive analysis of the influence of both the mutations and the population size on sample efficiency.

Learning Curves

An analysis of the learning curves displayed in the 'Asymmetric Advantages' layout is performed. As shown in Figures 6a and 6b, the mutation variations have accelerated the rate, at which the agent's performance arrives at its optima. Within $3.8\text{x}10^6$ environment timesteps, the learning curve in Figure 6b has reached its optimal performance (203). Compared to Figure 6a, where we need almost twice as many timesteps to get our peak performance (180). This means that after the variations, PBT in the 'Asymmetric Advantages' layout has a higher sample efficiency.

We can draw similar observations from Figure 6c and Figure 6d, which illustrate the learning curves of agents trained in a population of size 4. In those figures, the optimal performance peaks at 235 and 240, respectively. We will now examine the influence of population size on the learning process, focusing on Figures 6b and 6d. These figures offer valuable insights into how varying population size influences agent performance in PBT. Notably, agents trained in a population of size 3 reach their peak performance at 203, while agents trained in a population of size 4 exhibit a performance level of 240. This suggests that training agents in larger populations may result in better-performing agents.

Now we will analyze the learning curves displayed in the 'Coordination Ring' layout. Similarly to the 'Asymmetric Advantages' layout, custom mutation factors have proved to accelerate the learning rate. In this layout with the custom mutation factor, the agent needs $3\text{x}10^6$ timesteps, to achieve close to the optimal performance, whereas, with the baseline mutation factor, it needs the whole $8\text{x}10^6$ timesteps to achieve similar performance. At the last timestep, the agent with baseline mutation factors finishes with a performance of 170, and the agent, using custom mutation factors 195. This means that after the variations, PBT in the 'Coordination Ring' layout has a higher sample efficiency.

To evaluate the effectiveness of the population in this layout, we will examine Figures 6f and 6h. In contrast to the 'Asymmetric Advantages', the learning curves depicted in those figures have similar shapes and similar peek performances of 195 and 200, respectively.

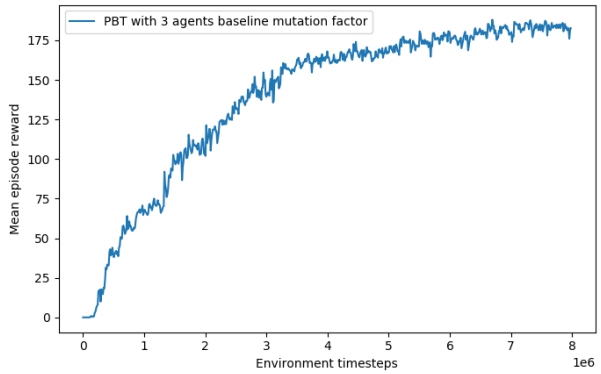
The data analysis reveals that introducing variations has a positive impact on the agent's efficiency in both layouts. An intriguing explanation for the enhanced performance in the 'Asymmetric Advantages' layout is the emergence of coordinated policies within the population. These variations promote agents to collaborate with each other in order to maximize their rewards. Previously, it was assumed that agents in this layout could achieve satisfactory but suboptimal rewards by acting independently. Additionally, this theory also explains why variations have similar effects on the second layout. In this case, agents were already compelled to coordinate from the beginning of training due to the high probability of collisions. Nevertheless, the introduction of mutations in the population might have facilitated earlier coordination among agents to avoid such collisions during training.

Furthermore, we observe that agents trained within larger populations consistently outperform those trained with smaller sample sizes in the 'Asymmetric Advantages' layout. This is due to an improved exploitation-exploration trade-off enabled by the larger population size. PBT balances the refinement of promising solutions (exploitation) and the search for new solutions (exploration). A larger population allows for a better trade-off, with some individuals refining solutions while others explore alternative paths. This dynamic interaction within a larger population enhances overall performance. On the other hand, in the 'Coordination Ring' layout, the performance of trained agents does not substantially change with varying population sizes. This is mainly because the agents have already identified and adopted an optimal solution within the layout. Consequently, achieving further performance improvements becomes challenging due to layout constraints. The cooperative interactions in the 'Coordination Ring' layout have already been maximized, limiting the potential for additional enhancements. By recognizing these distinctions in performance response to population size across different layouts, we gain insights into the effectiveness and limitations of PBT in different scenarios.

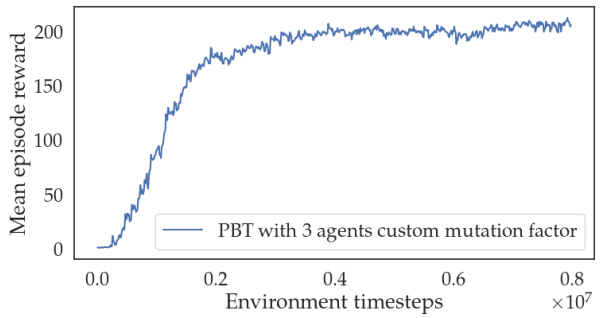
Final Performance

Considering our observation of the most optimal learning curves were produced with 4 agents and the custom mutation factor, we've decided to pair this particular agent for further performance evaluation. To assess the influence of variation and population size on the ultimate performance of Population-Based Training (PBT), we pair every agent type with a human proxy. Figure 7 represents the average reward per episode for each pair. This graphical representation is then compared against the results of the initial experiments as illustrated in Figure 5b.

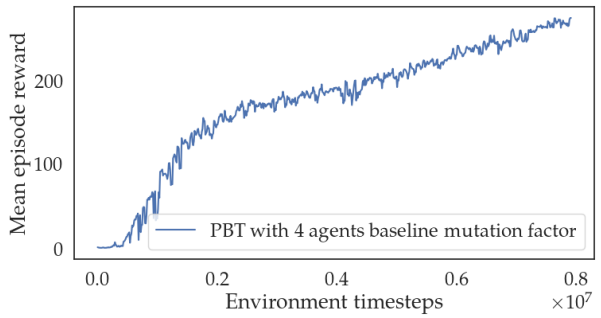
The analysis of PBT's performance, as depicted in Figure



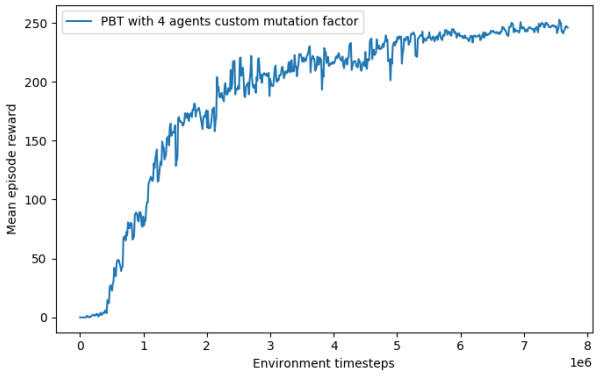
(a) Asymmetric Advantages with 3 agents and baseline factors of mutation.



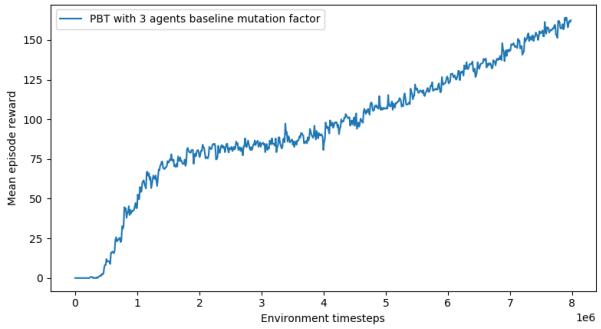
(b) Asymmetric Advantages with 3 agents and custom mutation factors.



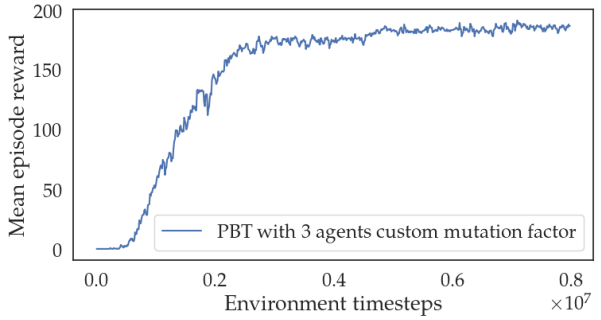
(c) Asymmetric Advantages with 4 agents and baseline factors of mutation.



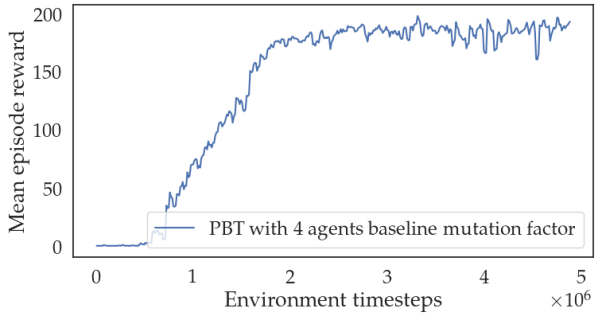
(d) Asymmetric Advantages with 4 agents and custom mutation factors.



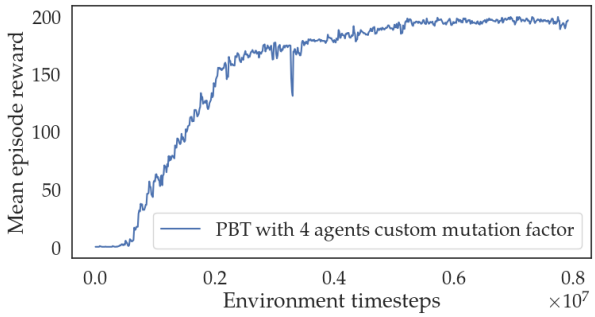
(e) Coordination Ring with 3 agents and baseline mutation factors.



(f) Coordination Ring with 3 agents and custom mutation factors.



(g) Coordination Ring with 4 agents and baseline factors of mutation.



(h) Coordination Ring with 4 agents and custom mutation factors.

Figure 6: Learning curves of a PBT agent in the 'Asymmetric Advantages' and 'Coordination Ring', displaying the average sparse reward per episode during training over 400 timesteps.

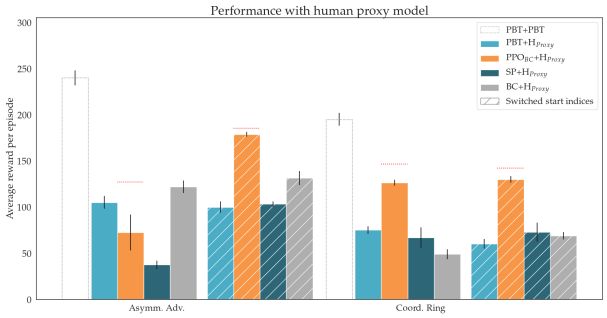


Figure 7: Performance results of the PBT agent when paired with the human proxy in comparison to the performance of other agents matched with the same proxy. For each layout, performance is given by the average sparse reward per episode during training over 400 timesteps. Striped bars indicate results when the starting locations of the agents have been swapped.
Note: Horizontal red dotted lines are $PPO_{H_{Proxy}} + H_{Proxy}$

7, is provided. In the 'Asymmetric Advantages' layout, an agent trained in a population of 4 with customized mutation factors achieves an impressive peak average reward of around 100 when paired with H_{Proxy}. When paired with other instances of itself, it maintains a mean reward of 240. This performance surpasses that of the baseline-mutated agent, which was trained in a population of three and achieves a peak score of 65 and 180 when paired with the same agents. Furthermore, in the 'Coordination Ring' setup, the baseline agent demonstrates an average performance of 70 and 170. However, the agent trained in a larger population, using an increased mutation probability, achieves even better results. It attains a peak average reward of 80 and 195 when paired with H_{Proxy} and itself, respectively. The modifications applied in the experiments have a positive effect on PBT's functional efficacy. The performance in comparison to other baseline agents remains relatively consistent, with PBT outpacing the PPO_{BC} agent only in the 'Asymmetric Advantages' layout. In the experiments without alterations, PBT's average reward continuously fell below that of PPO_{BC}. Generally, the PBT agent's final performance in the 'Coordination Ring' layout demonstrates a slight advancement, while in the 'Asymmetric Advantages' setting, a substantial boost of performance of almost 40 is observed. Based on our reward system, this enhancement suggests that, on average, agents trained in a population size of 4 with customized mutation factors were able to serve an additional two dishes.

To summarize the analysis of Figure 7, using custom mutation factors within a larger population appears to provide a small performance boost in the 'Coordination Ring' layout while proving highly beneficial for agents trained in the 'Asymmetric Advantages' layout. Consequently, although adding high-variance agents to the initial population improves sample efficiency for some layouts, it remains unclear whether this method can consistently enhance PBT's final performance across all layouts.

5 Responsible Research

In the course of the investigation, there are several ways in which the finding could have been misrepresented. First of all, the evaluation of the PBT agents relies heavily on the human models derived from previously collected human data. However, this approach may have its limitations, as information could have been omitted or incorrectly recorded during the process, potentially impacting the validity of our results. While it is true that missing data in the original experiments may not completely invalidate the results, training against a human model derived from limited data does not provide a complete picture of the performance of these methods with real humans. However, it is worth mentioning that the original paper, which makes use of the same human data, conducted experiments involving actual people who corroborated their findings. This adds credibility to the results and reduces the likelihood of significant misrepresentation.

Secondly, misrepresentation could occur if the data relating to the agents' learning curves or final performance were fabricated or falsified. Environmental parameters such as the seeds or the number of episodes executed could be misrepresented or reported inaccurately, which could compromise the validity of the results. However, such malpractice would undermine the very purpose of our research, which is to accurately gauge the effectiveness of PBT in a cooperative environment.

In terms of reproducibility, the experiments described in the paper can be replicated, given sufficient computational resources. The specifications of the experiments can be found in this public repository [1]. In the repository, a Jupiter Notebook, used to generate the confirming graphs can be found as well. The primary limitation of reproducing the experiments is the time required to train agents, which could span several days. Nonetheless, given enough time, the experiments can be fully replicated, allowing the findings to be validated.

6 Related Work

6.1 Population-Based Training

Population-based training (PBT) is a powerful and flexible reinforcement learning approach that dynamically adjusts hyperparameters based on agent performance within the population. This characteristic makes it an ideal tool for complex and ever-changing multi-agent environments.

The power of PBT has been demonstrated in several previous studies. For example, PBT was used to optimize the hyperparameters of deep neural networks in various Atari games[6]. They demonstrated PBT's superior performance and computational efficiency compared to traditional methods like grid search and random search.

OpenAI[7] used PBT in combination with Proximal Policy Optimization (PPO) to train agents in the game Dota 2. The results highlighted the effectiveness of PBT in discovering powerful learning policies and hyperparameters across different scenarios.

Additionally, Chaplot et al.[8] utilized PBT to train agents for 3D navigation tasks. Their study demonstrated PBT's

[1] https://github.com/INestorov/overcooked_pbt

ability to devise strategies that are robust against environmental changes.

The aforementioned studies support the use of PBT for training agents in multi-agent reinforcement learning tasks, establishing it as a promising direction for future exploration. As discussed by Carroll[4], the combination of PBT with PPO in the Overcooked environment, has further underlined the effectiveness of PBT, but also highlighted opportunities for improvement when agents are paired with human players. Building on this foundation, our research will not only seek to validate these findings but also explore modifications to PBT to potentially enhance its performance.

6.2 Ad-Hoc Teamwork

Population-based training (PBT) holds the potential to significantly enhance robust ad-hoc cooperation in multi-agent environments, a capability that sets it apart from other approaches like zero-shot coordination[9]. While both methods aim to ensure compatibility with diverse agent types, it is important to examine the concepts of ad-hoc cooperation and zero-shot coordination to provide a comprehensive understanding of their differences.

Ad-hoc cooperation refers to the ability of agents to collaborate spontaneously without prior coordination or explicit communication. In the context of the Overcook environment, it involves agents dynamically adapting their behaviors to work together towards a common goal, such as efficiently preparing meals in a chaotic kitchen. This type of cooperation is essential for agents to effectively tackle unpredictable situations, where the composition and objectives of the agent team may vary over time.

On the other hand, zero-shot coordination primarily focuses on enabling agents to coordinate their actions without prior knowledge about each other's behavior or capabilities. It relies on predefined coordination mechanisms or communication protocols to establish a common understanding among agents. While this approach can be effective for short-term interactions, it may lack the adaptability required in evolving and dynamic cooperative environments, where new agents with unfamiliar behavior might join or existing agents may change their strategies.

In this context, PBT stands out as a powerful technique for training agents capable of robust ad-hoc cooperation. PBT not only emphasizes compatibility with diverse agent types but also offers long-term adaptability. This adaptability is particularly valuable when unfamiliar agents enter the scene, as PBT allows the agents to continuously adapt and improve their cooperation strategies over time. Consequently, agents trained through PBT have the potential to cooperate more reliably with humans in real-world scenarios, although further experimentation is needed to verify this hypothesis.

7 Conclusions and Future Work

7.1 Conclusion

Although agents utilizing self-play methods during training excel when paired with similar agents, their performance significantly declines when collaborating with human players. A study conducted using the cooperative game Overcooked demonstrated that population-based training (PBT) falls short compared to agents trained with human models. Nevertheless, PBT shows improvement over purely self-play techniques. This suggests that incorporating a population of independent policies has partially succeeded in enhancing performance when simulating human players. PBT also exhibits higher sample efficiency, requiring less data to optimize policies in scenarios with frequent collisions and demanding coordination.

In an effort to enhance generalization in gameplay with human players, the initial PBT population has been supplemented with custom mutation factors. Additionally, changes have been made to increase the population size, aiming to improve the exploitation-exploration balance that PBT relies on. However, it was observed that the inclusion of noise and larger population sizes had minimal impact on the final performance of PBT itself in layouts with high collision risks, such as the 'Coordination Ring' layout. Conversely, these changes had a significant positive effect on layouts where collisions are highly improbable, such as the 'Asymmetric Advantages' layout.

To determine whether incorporating high-variance population members and exploring different population sizes can enhance PBT's overall performance, further research is required.

7.2 Limitations

The primary restrictions encountered during the research stemmed from limited computational resources and a restricted timeline for the study. Whenever PBT's performance had to be evaluated, we needed to train a collection of agents numerous times across different seeds, each instance consuming many hours. Given the short period for the research, this process significantly increased the burden on the parameters (such as seeds, population size, and mutation factors) and reduced the number of experiments that could be conducted.

7.3 Future Work

Future research should continue investigating the potential impact of incorporating high-variance agents into the PBT population to determine whether it consistently improves overall performance. Exploring more sophisticated methods for introducing variability beyond simple population mutation could also yield advantages.

Another area of interest for future research is examining the integration of a human-modeled agent, such as the BC agent, into the PBT population. PBT's suboptimal performance against agents trained on human models may be attributed to the absence of human data during training. By introducing a human model to PBT, we can ascertain whether this is the underlying cause and whether PBT could outperform agents like PPO_{BC} if trained on a population of policies. Validating this would reinforce the notion that diversity within the population enhances performance against human players.

Additionally, it is important for future research to investigate the implications of varying population sizes on agent training and performance. Due to the constraints of our study,

we were unable to fully explore this aspect. Conducting experiments on different layouts would also be valuable in order to evaluate the benefits of using higher population sizes and custom mutation factors in a broader context.

Overall, further research in these areas would contribute to a deeper understanding of PBT's capabilities and potential improvements in training and performance.

A Mutation Factors

Mutation	Mutation Probability	Mutation Factors
Baseline Mutation Factors	0.33	[0.75, 1.25]
Custom Mutation Factors	0.45	[0.5, 0.75, 1.25, 1.5]

B PBT Pseudo code

```
Initialize a population of N agents with random policies

Repeat until convergence:
    Evaluate the current population of agents by running them in the Overcooked environment

    Sort the population based on their performance

    Select the top-performing agents for exploitation
    Select the bottom-performing agents for exploration

    For each agent in the exploitation group:
        Mutate the agent's policy using small perturbations

    For each agent in the exploration group:
        Copy the policy of a randomly selected agent from the exploitation group

    Train each agent using Proximal Policy Optimization (PPO):
        For a fixed number of epochs:
            Collect trajectories by running the agent in the Overcooked environment
            Compute advantages and value estimates using the collected trajectories
            Update the agent's policy using PPO's update rule

    Replace the current population with the updated agents

Return the best-performing agent from the final population
```

References

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

[2] HumanCompatibleAI, "Humancompatibleai/overcooked_ai: A benchmark environment for fully cooperative human-ai performance.."

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[4] 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," 2020.

[5] HumanCompatibleAI, "Humancompatibleai/human_aware_rl: Code for "on the utility of learning about humans for human-ai coordination"."

[6] 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," *CoRR*, vol. abs/1711.09846, 2017.

[7] OpenAI, :, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," 2019.

[8] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," 2020.

[9] H. Hu, A. Lerer, A. Peysakhovich, and J. N. Foerster, ""other-play" for zero-shot coordination," *CoRR*, vol. abs/2003.02979, 2020.