Explaining the performance of different reward strategies for chef's Hat players based on loss value and Q-value

Gabriel Silva[1] and Pablo Barros[2]

***Abstract*—XAI method for agent behavior analysis using q-value and loss evaluated on DQL agent in chef's hat. the development processes of agent using deep reinforcement learning (DRL) methods with deep Q-learning (DQL) to solve the complexity problems of Q-learning, using the OpenAI Gym environment of Chef's Hat, a card game developed to analyze human behavior with multimodal and competitive interactions in mind, allowing it to be easily followed and modeled by artificial agents. The proposal is to evaluate the performance and explain based on the loss value and the q-value of the agent with different reward strategies: (I) based on moves that focus on playing cards with high values, and (II) based on moves that focus on a higher number of discards.**

I. Introduction

Reinforcement learning (RL) is a machine learning (ML) technique that trains software to make decisions in search of the best results. It mimics the trial-and-error learning process that humans use to achieve their goals [8]. The results of each attempt, regardless of its success, are used to train the agent through a reward/punishment system and determine whether the actions taken are valid or not. The agent learns from its mistakes and successes and, in the end, is expected to perform the task masterfully. However, to get around the complexity of Q-Learning, we can use Reinforcement Learning combined with Artificial Neural Networks (ANN), thus having Deep Reinforcement Learning (DRL) [3 , 10].

In this paper we implement the deep reinforcement learning (DRL) method to develop an agent. We use two different reward strategies, one focusing on the value of the discarded card, and the other focusing on the amount of discarded cards. The competitive game environment of Chef's Hat allowed the performance between the agent and three random agents to be evaluated. The results obtained show how different strategies present performances that, even though different, managed to train the agent in the rules of a competitive game such as Chef's Hat. The objective of this article is to evaluate and explain the agent's performance throughout the training and in the validation testing stage of the trained model, using the loss value and the q value as a basis.

This paper is organized as follows. Section II presents the environment in which the model was trained and evaluated. Section III presents the learning method used to teach the model to play. Section IV presents the methods used in the methodology for developing the agent in reinforcement learning. Section V presents the evaluation of the results acquired during training and testing. Finally, section VI presents the final considerations.

[1]Gabriel Lima G. da Silva is with Postgraduate Program in Computer Engineering (PPGEC), University of Pernambuco, Recife, Brazil glgs@ecomp.poli.br

[2]Pablo Vinicius A. Barros Sony, Belgium pablo.barros@sony.com

II. Chef's Hat Environment

As described in Barros et al [1 , 2] the environment implements specific action and observation spaces, and calculates scores and performances after the game ends. The game has a role-based hierarchy in a kitchen context: each player can be a chef, a sous-chef, a waiter or a dishwasher. Players are given 17 cards, and each player tries to be the first to get rid of his ingredient, trying to make the pizza first before the others, which will promote him to the position of Chef, this happens over several rounds (or turns). The playing flow of the game can be seen in figure 1.

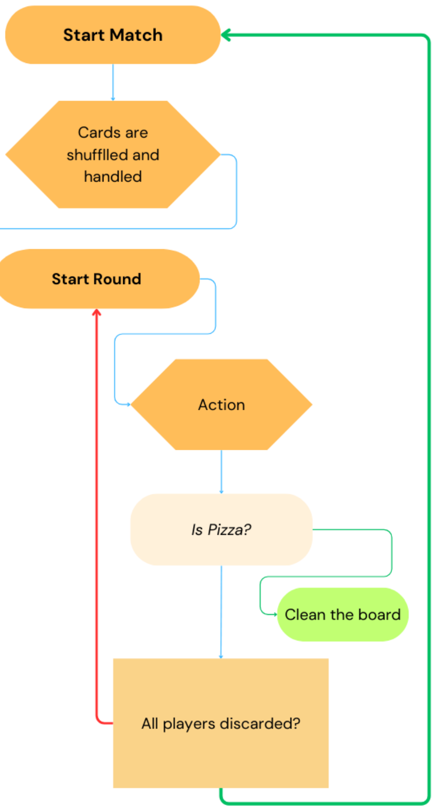

Fig. 1. The flow of the Chef's Hat card game.

III. Deep Reinforcement Learning

Reinforcement learning uses interactions with the environment in which it is inserted to learn based on rewards as a guide in this learning process for each well-executed action. This process is done entirely without any human intervention during the agent's learning process, as presented in Qiang [4], and Sutton [8]. Reinforcement learning algorithms tend to solve problems where the necessary actions are not explicitly programmed. Using the state spaces and actions that the agent can perform in the environment, thus generating the reward for trial and error, as applied by Barros et al [2] during the training of agents based on reinforcement learning. Deep Reinforcement Learning (DRL) is an algorithm in the field of reinforcement learning, it combines the principles of artificial neural networks (ANN) and reinforcement learning (RL) achieving great success in various types of complex tasks, making agents learn optimal policies in complex environments as described by Li, S [3] and Fan, J [10].

Using the Q-Learning algorithm to compute the optimal action value function (Q-function) that identifies the states for expected future rewards [9]. This is done by replaying experiences acquired during agent training, helping to memorize related sequential experiences and storing them in a replay memory buffer. This memory buffer is randomly sampled during network updates to break temporal dependencies and stabilize learning [5].

IV. Methodology

The reinforcement learning method was used to train an agent in conjunction with artificial neural networks (ANN). Chef's Hat environment in which the agent was inserted has great complexity due to the number of states and actions that the agent will perform during the learning process. The method used was an algorithm that uses neural networks with Q-learning, allowing the agent to learn in complex environments, creating a deep Q-network (DQN). The parameters used to create the neural network responsible for training the agent using deep learning methods are presented in the table I below.

TABLE I
LAYERS OF NEURAL NETWORK

Layers	Neurons	Activation Function
Input layer	28	
Input layer	200	
Dense layer	256	ReLU
Dense layer	256	ReLU
Output layer	200	Softmax

The experiment puts the learning agent to play against three random models; A training routine lasting 1000 games was run, using reward strategies with different focuses:

- Discards with higher value cards.
- Discard the largest number of cards.

The reward focusing on the highest value card, can be seen in the formula (1). In equation, the value of the reward will be proportional to the value of the card played raised to the negative value of the number of cards discarded.

$$reward = cardValue^{-qtdDiscarded} \qquad (1)$$

As for the reward based on the number of cards discarded, it can be seen in the formula (2).In the equation, the reward value is proportional to the amount of discarded cards. Where the value of the card is multiplied by weight with value (-0.11), and the result of the multiplication is added to the amount of discarded cards in that action.

$$reward = [cardValue * (-0.11)] + qtdDiscarded \qquad (2)$$

V. Results

Analyzing the strategies used by the model during training, we have as a sample the graph that shows the agent's loss as the epsilon value decays, as shown in the figure 2.

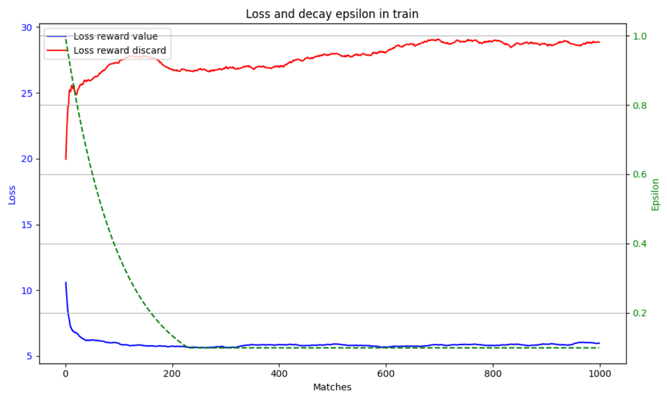

Fig. 2. Decay of loss and epsilon value over training with 1000 games

During the training that was carried out with 1000 games, to know how the agent is learning during training we observe the loss value shown in the figure 2, where:

- Red line: Represents the card discard strategy (number of cards discarded).
- Blue line: Represents the card value strategy (value of cards discarded).

For the strategy that rewards actions that prioritize plays with card value (blue line), the loss decreases more steadily, suggesting that the model is finding a more robust decision policy. Prioritizing card value leads to more strategic decisions, such as discarding high-value cards to force opponents to pass. In parallel, for the strategy that rewards actions that prioritize plays with the number of cards discarded (red line), the loss shows greater fluctuation, indicating that the model, as it trains finding a consistent policy, moves away from what could be the optimal play. This may be because prioritizing the number of cards discarded can lead to less strategic decisions, such as discarding many low-value cards without pressuring opponents.

With the trained models, testing experiments were performed to validate the model strategies, and evaluate the Q-values for both strategies. These experiments were performed by having the agent play against 3 random agents, where a test was performed for each agent, running an hundred games.

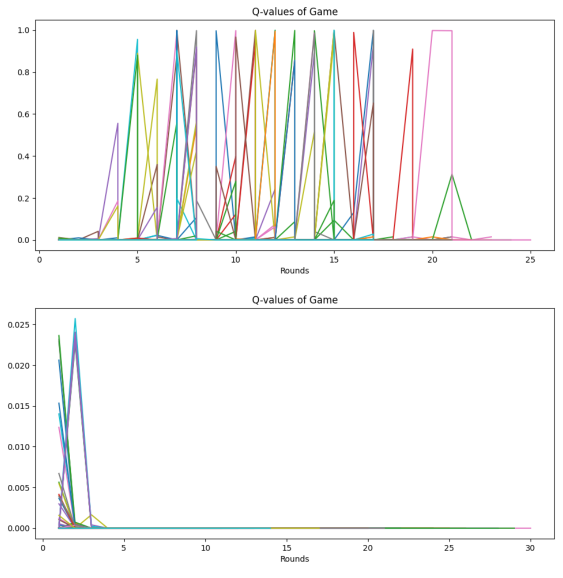

Fig. 3. Q-value values: Plot (I) reward focused on the value of the discarded card, Plot (II) reward focused on the number of discarded cards

Each line in the graph represents the q-value values during a game, and the number of rounds the agent played can be seen on the x-axis. The agent trained with the reward focused on the value of the discarded card starts the games with low q-value values, as shown in graph I in figure 3, and as the game progresses, its values increase. Graph II in image 3, shows the performance of the agent trained with the reward focused on the number of discarded cards, where the agent tends to start the rounds with high values, and as the game progresses, its values tend to decrease. It can also be seen that the number of rounds on the x-axis in graph II is greater than in graph I. It is possible that the agent trained with the strategy based on the number of discarded cards plays more rounds to win the game.

A test evaluation routine where the trained agent plays 10x100 games against random agents, without additional training, measuring the average of the total victories achieved by the model in each game. This experiment aims to provide us with important information about how the model trained using different reward strategies learns to outperform a simple strategy based on random choices. Based on the information obtained through the average number of wins and their standard deviation during tests for both models, it is shown that the strategy of prioritizing plays using cards of higher value and the strategy of discarding the largest number of cards present a learning of their action policies where they manage to win a significant amount against random agents as described in the table II.

To validate the performance of the two strategies, a test was performed with 100 games to compare the number of wins. In this test, an agent trained with the reward focused on discarding the highest value cards, an agent trained with the reward focused on the number of discarded cards, and two random agents were used.

TABLE II
RESULTS OF THE EXPERIMENT : REWARD I - VALUE OF THE DISCARDED CARD, REWARD II - NUMBER OF CARDS DISCARDED.

Model	Victories	Random1	Random2	Random3
reward I	**55.5**±4	14.7±3.56	14.2±2.8	15.6±2
reward II	**45.4**±3.2	17.4±3.28	16.9±3.48	20.3±3.62

TABLE III
RESULTS OF THE EXPERIMENT : NUMBER OF WINS OVER 100 GAMES

Model	Victories
Player - reward value card	55
Player - reward discarded quantity	30
Random I	9
Random II	6

Based on the information obtained through the number of victories during the games, it is shown that the strategy of prioritizing plays using higher value cards proved to be more effective than the strategy of discarding the largest amount, indicating that the results are more consistent as shown in the image III.

VI. CONCLUSION

In this paper, an agent was developed using the deep reinforcement learning method with two different strategies for its reward, namely: reward based on the value of the discarded card and reward based on the number of discarded cards, both for the same scenario of the competitive game Chef's Hat, and using the loss and Q-values to explain the agent's performance.

Evaluating the loss value during the training stage, it is shown that the loss value for reward based on the value of the card has a more stable progression, decreasing throughout the training, while the loss value for reward based on the number of discards has a more unstable progression, demonstrating that the agent was learning more slowly. In the Q value during the validation testing stage, it was shown that both strategies performed well against random agents. However, the agent trained with the discarded card value strategy had an average win rate of over 50%, and the agent trained with the discarded card number strategy had an average win rate of over 40%.

While a game run to compare the two strategies showed that rewarding based on the value of the discarded card was more effective and stable than rewarding based on the number of discarded cards, as the agent trained using rewarding based on the value of the card won 55 games out of 100, while the agent trained using rewarding based on the number of discarded cards won 30 games out of 100.

As future work, use explainable AI methods, such as an introspection-based approach, to have a better explanation of the agent's Q-value data during its plays, such as "why was this action selected instead of another?".

References

[1] Barros, P., Tanevska, A., Cruz, F. & Sciutti, A. Moody learners-explaining competitive behaviour of reinforcement learning agents. *2020 Joint IEEE 10th International Conference On Development And Learning And Epigenetic Robotics (ICDL-EpiRob)*. pp. 1-8 (2020)

[2] Barros, P., Tanevska, A. & Sciutti, A. Learning from learners: Adapting reinforcement learning agents to be competitive in a card game. *2020 25th International Conference On Pattern Recognition (ICPR)*. pp. 2716-2723 (2021)

[3] Li, S. Deep reinforcement learning. *Reinforcement Learning For Sequential Decision And Optimal Control*. pp. 365-402 (2023)

[4] Qiang, W. & Zhongli, Z. Reinforcement learning model, algorithms and its application. *2011 International Conference On Mechatronic Science, Electric Engineering And Computer (MEC)*. pp. 1143-1146 (2011)

[5] Arulkumaran, K., Deisenroth, M., Brundage, M. & Bharath, A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*. **34**, 26-38 (2017)

[6] Barros, P., Sciutti, A., Bloem, A., Hootsmans, I., Opheij, L., Toebosch, R. & Barakova, E. It's food fight! Designing the chef's hat card game for affective-aware HRI. *Companion Of The 2021 ACM/IEEE International Conference On Human-Robot Interaction*. pp. 524-528 (2021)

[7] Barros, P., Bloem, A., Hootsmans, I., Opheij, L., Toebosch, R., Barakova, E. & Sciutti, A. The Chef's Hat Simulation Environment for Reinforcement-Learning-Based Agents. *ArXiv Preprint ArXiv:2003.05861*. (2020)

[8] Sutton, R. Reinforcement learning: An introduction. *A Bradford Book*. (2018)

[9] Watkins, C. & Dayan, P. Q-learning. *Machine Learning*. **8** pp. 279-292 (1992)

[10] Fan, J., Wang, Z., Xie, Y. & Yang, Z. A theoretical analysis of deep Q-learning. *Learning For Dynamics And Control*. pp. 486-489 (2020)