Moody Learners - Explaining Competitive Behaviour of Reinforcement Learning Agents

Pablo Barros*, Ana Tanevska*, Francisco Cruz†‡, Alessandra Sciutti*
*Cognitive Architecture for Collaborative Technologies Unit (CONTACT),
Italian Institute of Technology (IIT), Genova, Italy
Email: {pablo.alvesdebarros, ana.tanevska, alessandra.sciutti}@iit.it
† Escuela de Ingeniería, Universidad Central de Chile, Santiago, Chile
‡ School of Information Technology, Deakin University, Geelong, Australia
Email: francisco.cruz@deakin.edu.au

***Abstract*—Designing the decision-making processes of artificial agents that are involved in competitive interactions is a challenging task. In a competitive scenario, the agent does not only have a dynamic environment but also is directly affected by the opponents' actions. Observing the Q-values of the agent is usually a way of explaining its behavior, however, it does not show the temporal-relation between the selected actions. We address this problem by proposing the Moody framework that creates an intrinsic representation for each agent based on the Pleasure/Arousal model. We evaluate our model by performing a series of experiments using the competitive multiplayer Chef's Hat card game and discuss how by observing the intrinsic state generated by our model allows us to obtain a holistic representation of the competitive dynamics within the game.**

***Index Terms*—Explainable Artificial Intelligence, Reinforcement Learning, Intrinsic Confidence.**

I. Introduction

The application of reinforcement learning (RL) models has been on a rise following the onset of deep reinforcement learning [1]. The ability of RL models to solve complex problems, represented mostly by the association of high-dimensional states and a large number of discrete or continuous actions, has recently led to the development of expert systems for guiding autonomous cars [2], [3], predicting the stock exchange impact [4], [5], and coordinating a swarm of robots to protect the environment [6], [7], to name a few examples.

As their popularity grows, the need for providing stable and trustworthy solutions for real-world problems also increases. When deployed in production environments, these RL models, although designed to achieve impressive performance, are not easily understandable, and thus need experts to address the problems that may arise. The robustness of the RL solutions is thus limited by the general inability of explaining easily how these models learn and derive their state-action mapping [8]. This has led to the fast development of eXplainable AI (XAI) [9] as a complementary field for deep reinforcement learning.

The community around XAI has been investigating how to approach the understanding of reinforcement learning when applied to different problems. Problems that involve the processing of human-based data are somehow the easiest ones to address, as demonstrated by explaining recommendation systems [10], applications on the medical domain [11], and robotics applications based on known behavior [12]. However, it is much more difficult to explain the agents' behavior in scenarios where the environment cannot be easily modeled [13], or the solutions are unknown a priori [14]. In this regard, recent applications derive human-level explanation based on the agent's own knowledge of the situation [15]. In particular, transforming the selected Q-values for each action into a confidence metric, using a re-shaping function based on the logarithmic transformation [15], improved the understanding of a robot trying to solve a grid-based navigation task. The confidence metric measures how a specific action contributes to the robot reaching its goal however it does not carry any temporal correlation between the actions selected by the agent during the navigation. This means that the metric cannot be used to explain the behavior of the agent within the entire simulation, but only for each action.

This approach functions quite well when the end state of the task is clearly defined, and it is possible to measure the distance between the current state and the end state. However, it does not deal well with competitive scenarios where a set of agents have to learn decisions that a) maximize their goal, and b) minimize their adversaries' goals. Besides considering the dynamic of the scenarios, they usually have to deal with the interactions between the agents themselves. Some of the most common applications for competitive reinforcement learning involve the design and implementation of learning agents in simulations that simplify several aspects of the real world such as the implementation of grid-based autonomous vehicles [16], life-simulation/resources gathering [17], and multi-player games [18]. In these cases, explaining the agents' behavior based on the simplified environment is somehow possible, but they would not scale for real-world applications.

In this regard, we propose a novel methodology to explain the behavior of artificial agents in real-world modeling a competitive scenario. We deploy our experiments onto the multiplayer Chef's Hat card game [19] as it offers a dynamic interaction between the players and simulates directly the real-world counterpart game. In the scenarios' baseline, four agents play against each other and their performance is measured directly based on how many games they win [20]. It is not possible, however, without an extensive manual observation of their action-selection pattern to explain their winning behavior while the game happens.

To identify the impact of each selected action, we propose the *Moody framework* which uses the agents' own evaluation of the game to provide a temporal reference for the impact of selected actions and translate it into an pleasure/arousal representation [21] of the agent's mood. Our method builds on the introspective transformation from the Q-values [15], and introduces a Growing-When-Required (GWR) network to transform confidence readings into the pleasure/arousal scale, and to create strong inter-action correlation, which directly

maps the action-selection-mood triplet in our competitive card game scenario and carries a temporal momentum that we use to explain the agents' performance while the game happens.

We also investigate the *Moody framework* as a tool for explaining the agents' behavior under a competitive perspective. The intrinsic mood readings are directly formulated by an agent assessing its own actions. In a competitive scenario, the actions of the agent's opponents impact directly the agents' own mood. Thus, being informed on how each agent perceives their opponent's actions can give us a much richer explanation of the performance of the agent while the game happens.

In this paper, we formalize the entire *Moody framework*, and aim on addressing two main research questions: 1) How the mood readings provide the understanding of the agents' behavior on the Chef's Hat game; 2) How close is the representation of the mood estimated by an agent about its opponents from the real mood of each of these opponents. To answer these questions we perform a series of experiments where different agents play a series of games using the Chef's Hat simulation environment. We explain how the agents' own assessment of the opponents' actions impacts the performance explanation. Also, to better understand the impact of the *Moody framework* on the XAI community, we discuss the effect of our self-evaluation in competitive scenarios and how they can provide a closed-world representation of the entire game.

II. Game Mechanics and Agents Implementation

A. The Chef's Hat Card Game

The game setup which we used as an environment for our artificial agents is the Chef's Hat card game [22]. Chef's Hat as a game provides a controllable action-perception cycle, where each player can only perform a restricted set of actions. This in turn allows each player to behave as organically as possible, and allows us to directly measure the impact of each action within the naturally-controllable real-world scenario.

Chef's Hat is a 4-player round-based card game, where each person has a restaurant-context role (Chef, Sous-Chef, Waiter or Dishwasher), that is updated after each game based on the order of finishing the previous match. At the beginning of the game the players get the full hand of cards (17 cards per player) dealt to them, and taking turns they need to dispose of their cards as quickly as possible. The cards represent ingredients for pizzas, and each round consists of the players making pizzas by discarding cards in a certain manner. The details of the pizza-making rules and the role hierarchy are explained fully in the games' formal description [22]. In order to better explain our model, we detail the flow of one full game in Algorithm 1.

B. The Chef's Hat Simulator Implementation

We implemented our scenario using the OpenAI-based simulation environment of Chef's Hat [19]. The environment simulates all of the game mechanics, and thus provides a 1:1 simulation of the real-world game, and provides an easy implementation of different agents to play the game, as shown in Figure 1.

The current game state for each player is represented as an aggregation of the cards of the player and the current cards in the playing field for a total of 28 values. Each player can choose among 200 different actions to perform in each state, each of them representing a unique combination of cards to be discarded, or a pass action.

Shuffle the deck;
Deal an equal amount of cards per player;
Exchange roles;
Exchange cards;
if *special action is evoked* **then**
  Do special action;
end
FirstPlayer $\leftarrow$ *Hasgolden*11
FirstPlayer discard cards.
while *not end of the game* **do**
  for *each player* **do**
    if *player can, and want, to discard* **then**
      discard cards;
    else
      pass;
    end
    if *All players passed* **then**
      Make the pizza;
      FirstPlayer $\leftarrow$ *Lastplayertodiscard*
    end
    if *All players finished* **then**
      End of game.
    end
  end
end

Algorithm 1: The Game-flow of the Chef's Hat card game.

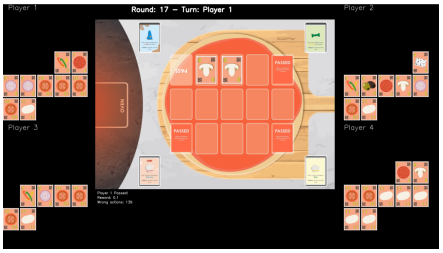

Fig. 1. Chef's Hat rendered simulation environment.

In order for the simulated players to be able to select their actions in an intelligent manner, we needed to train artificial agents to play Chef's Hat. For this we employ two different reinforcement learning algorithms based on Q-learning, which allow our agents to apply a temporal-difference calculation when updating the policy network and thus maximize the state transitions that will lead to the optimal reward.

C. The Learning Agents

The main characteristic of the *Moody Framework* is to provide a self-assessment of the agents' performance based on the agent's own judgments. To better evaluate that, in our experiments, we implement three different types of agents, two learning ones based on Deep Q-Learning - DQL [23] and on on Proximal Policy Optimization - PPO [24], and a dummy one which only select random actions. To guarantee that a taken action by one agent is valid, we use the proposed Chef's Hat action selection greedy policy which limits each agents' final action selection by applying a mask on the final Q-values composed of only the currently allowed actions, given a certain state.

As discussed by Barros et al. [20], the learning agents learn different strategies when trained to play the Chef's Hat game.

While the DQL agent learns a set of restricted actions, which are mostly deployed by the end of the match, the PPO agent usually has higher confidence in winning the game when using a set of actions at the beginning of the match.

In this research, we are interested in establishing an explanation on how each of these agents performs while playing Chef's Hat, so we do not focus or detail the agents' learning processes. To guarantee that each agent has its own strategies when playing the game, we follow the baseline protocols established by the *vs Everyone* self-play routine [20]. Figure 2 illustrates our implemented agents.

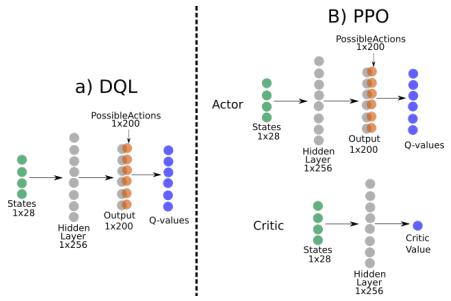

Fig. 2. The detailed implementation of the DQL and the PPO agents.

D. The Principles of Q-Learning

Both of our agents implement a Q-learning routine. The general Q-learning algorithm learns to maximize the probability of choosing an action that leads to maximum reward. For that, it calculates a Q-value (quality value) for each action given a state and updates the policy, in our case represented by a neural network, to maximize the expected reward. Using a temporal-difference calculation, it can take into consideration a sequence of steps that leads to the final state, represented by finishing all cards in your hand. The maximal reward is given once the player is the first one to reach the final state.

The typical Q-learning algorithm represents a function Q:

$$Q : SxA \rightarrow \mathbb{R} \tag{1}$$

where S is the state, in our case represented by the 28 values composed by the 17 cards at hand and the 11 cards at the board. The actions, A, are expressed using the 200 discrete values for all the possible actions.

To update the Q-values, the algorithm uses the following update function:

$$Q'(s_t, a_t) = Q(s,a) + \alpha \times (TD) \tag{2}$$

where t is the current step, α is a pre-defined learning rate and TD is the temporal-difference function, calculated as:

$$TD = r_t \times \gamma \times maxQ(s_{t+1}, a_t) - maxQ(s_t, a_t) \tag{3}$$

where r_t is the obtained reward for the state (s_t) and action (a_t) association, γ represents the discount factor, a modulator that estimates the importance of the future rewards, and $maxQ(s_{t+1}, a_t)$ is the estimation of Q-value for the next state.

III. A *Moody Framework*

To evaluate the performance of the agents when playing Chef's Hat is not a simple task. At first, state-based evaluations could give us a straight answer to performance measures, but they do need external evaluation and they would be completely biased on the evaluators' knowledge about the game. Counting the number of cards a player has at hand at a specific moment can indicate a precise performance value, but it can vary drastically depending on the players' own strategy. The same goes for measuring number of discarded cards in each turn.

Having a self-evaluation of the agents' own performance, thus, could solve this problem. As the action-selection strategy of the agent is made based on its own knowledge, the agent will be able to explain its own actions based on its own judgment.

One of the simplest ways of explaining the action-selection impact of a player is a Q-value-based observation. The Q-value represents the impact that each of the 200 actions will have, given a certain state, to reach the maximal goal. For a given state, the agent calculates which action would give it the highest probability of reaching the end state. The Q-values reading, however, does not allow us to establish how closely the agent is of winning the game by choosing that action, and thus, does not give us a better understanding of the agents' performance, but only of which action was the most appropriate given that specific state.

A. Calculating Confidence

To address the problem of calculating the impact of the selected action towards reaching the final goal, the introspection-based transformation of the Q-values [15] was proposed. This approach focuses on scaling the selected Q-value towards the final goal using a logarithm transformation which computes the probability of success, that we use as a confidence (C), that this specific action will lead the agent to reach the final state:

$$C = \left(\frac{1}{2} \times log_{10} \frac{Q(s,a)}{R^T} \right) \tag{4}$$

where $Q(s,a)$ is the Q-value of a selected action, and R^T is the maximum reward achieved by the agent at that specific time step. To normalize the probability between the interval $[0,1]$, the probability is saturated and every value smaller than 0 is set to 0, and every value above 1 is set to 1.

The probability is calculated based on how well that Q-value scales towards the maximum reward. In the Chef's Hat game, the agent only reaches the final reward if it wins a game. Also, for each action which is not the final reward, the agent gets a -0.01 reward in order to encourage the agent to prefer shorter paths to the final reward [20]. This impacts directly on the final reward calculation, as for each new turn the agent is playing, the final reward changes. To take this into consideration, we update the final reward per turn following:

$$R^T = 1 - (T \times 0.01) \tag{5}$$

where T represents the current turn on that specific game.

B. A Moody neural network

The observation of the Q-values on the Chef's Hat scenario already allowed for an understanding of how different learning mechanisms impacted on the final performance of each agent during a series of games [19]. Given the Chef's Hat specific greedy policy, however, each action that an agent selects is limited by the possible actions given that specific state. There is a strong limitation on the actions an agent can select, and thus, it impacts directly on the Q-values obtained for that specific state.

In a certain state, only a handful of possible actions are allowed and the agent has to establish which of those actions provide the best way to reach the maximum goal. This translates to the agent having a very small Q-value per action, as the allowed actions do not always comprise on the best ones. We illustrate this behavior on the readings of Figure 3 (a). This is especially impacted by the competitive characteristic of the game. The agent's next action will not be impacted only by its own previous action, but by the combined impact of the actions of its opponents.

Evaluating them alone in the Chef's Hat competitive scenario, thus, give us a very poor reading of the game context, as it only represents how that specific action has the highest probability, among all the allowed actions, to reach the final state. It does not carry any information about how this action was impacted by the previous ones. The same can be said by the confidence values, as they are a direct transformation of the Q-values. Although they can give us a clearer picture of how that specific action is contributing to the agent reaching the maximal goal, it will be extremely sensitive to the fast changes of the competitive dynamics of the game, as illustrated in the example plotted in Figure 3 (b).

To address that and in order to obtain a grounded representation of the agent's own internal state, we propose the implementation of a Growing-When-Required Network (GWR) [25] to generate prototypical representations of the impact of the measured confidences. The GWR was used recently to address personalization on emotion expression perception [26], [27], and in several continuous learning tasks [28], [29]. It is a self-organizing network that creates neurons that represents a series of input data and it can be trained online to approximate to the perceived stimuli.

Before feeding it to the GWR, we transform the calculated confidences into a pleasure/arousal (PA) scale [21], which represents a perceived experience two dimensions: pleasing/unpleasing and excited/calm. The PA model has been used as a standard way of representing intrinsic states in virtual agents [30]–[32], and will allow an easy understanding our *Moody framework* representations, and improve its applicability in different scenarios.

Each neuron of the GWR has a weight vector w_j that represents a prototype of the perceived information. In our scenario, we feed the GWR with perceived PA values. A newly perceived PA value will be associated with a best-matching unit (BMU) b, which is calculated by minimizing the distances between the perceived PA value and all the neurons on the GWR. Given a set of N neurons, b concerning the input $x \in \mathbb{R}^n$ is computed as:

$$b = \arg\min_{j \in N} \left(\|x - w_j\|^2 \right). \tag{6}$$

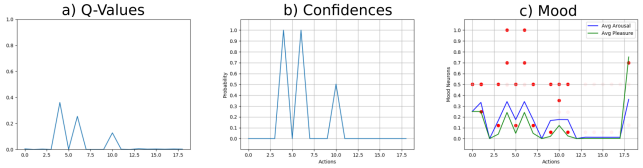

Fig. 3. Examples of the Q-values (a), confidence (b), and mood (c) readings of the same agent on the same game. In the mood readings (c), each of the dots represent a neuron of the network after an action was performed. As less transparent the neuron, as higher its habituation counter, meaning it was used recently to represent a perceived confidence.

When the BMU is found, new connections are created between the BMU and the second-BMU. Every neuron that is connected to the BMU is its topological neighbour. Each neuron has an aging mechanism, represented by an habituation counter $h_i \in [0,1]$, which represents how close this neuron was to the BMU, and thus, how important it is for representing the current input.

The habituation rule is given by:

$$\Delta h_i = \tau_i \cdot \kappa \cdot (1 - h_i) - \tau_i. \tag{7}$$

where κ and τ_i are constants that control the decreasing behavior of the habituation counter [25]. To establish whether a neuron is habituated, its habituation counter h_i must be smaller than a given habituation threshold t_h.

The network is initialized with two neurons and, at each learning iteration, it inserts a new neuron whenever the activity of the network when a confidence c is calculated, of a habituated neuron, is smaller than a given threshold t_a, i.e., a new neuron is created if $a(c) < t_a$ and $h_c < t_h$. The activity of the network is given by:

$$a(c) = \exp(-\left(\|x - w_j\|^2\right) \tag{8}$$

For each perceived PA value, the network updates, in an online manner, the state of the neurons by updating the BMUs and their neighbors, or adding new neurons. Neurons that are old, with a smaller habituation counter than a certain threshold, are removed from the network. That guarantees a dynamic behavior that represents the PA values as soon as they are perceived, and although the model has not a direct temporal processing mechanism, such as recurrent connections, we are able to maintain a temporal relation on the perceived PA values by adding neurons that represent PA values never perceived before and removing neurons which represent PA values that were not perceived anymore.

C. Assessing my own Mood

The goal of the *Moody framework* is to provide a self-assessment, coming from the agent itself, about its own actions. The GWR will generate a temporal momentum on the confidences that will represent the agent's performance based on its own judgment. To achieve it, we calculate the Q-values, and subsequently the confidences, of all the allowed actions over a given state. We then sum all these confidences and represent the agents' confidence that it can reach the final state, given the cards it has at hand and the board at that moment. The last step is to convert the confidence value into a PA scale. We then train the network with the PA values for each action taken by the agent.

Different from the typical clustering applications, the GWR on the *Moody framework* is a stateful model. We obtain the mood reading by averaging all the weights of all the neurons of the network at that specific state. As the neurons change over the game, giving the agents' own estimation, the mood readings will increase or decrease depending on how that agent perceives its own actions. Most importantly, the mood reading will carry the temporal relations between the estimations. This is observed by the example plotted in Figure 3 (c), where we also observe the temporal behavior of the neurons been inserted in the network to map the PA values. We clearly see when a neuron with higher PA value starts to get more active, and thus, has a smaller habituation.

The Chef's Hat card game is made to be played in a series of games. The roles assignments are given based on the finishing position of the last game, and they change the general games' winning probabilities. When playing with completely random agents, each agent has a chance of 25% of winning the game. However, if this agent won the previous game and it receives the Chef role in the current game, the chances of winning the game increase to 35% [22]. To represent the differences between the self-assessed confidence over an action and the environment given feedback when finishing a game, we calculate the PA values as follows:

$$P = \begin{cases} C \times 0.5 & if \quad action \\ 1 & if \quad victory \\ 0 & if \quad notVictory \end{cases} \tag{9}$$

$$A = \begin{cases} 0.5 - ((C-0.5)/0.25) & if \quad action \quad and \quad C < 0.5 \\ 0.5 + ((C-0.5)/0.25) & if \quad action \quad and \quad C \geq 0.5 \\ C1 & if \quad victory \\ C0 & if \quad victory \end{cases} \tag{10}$$

The pleasure (P) scale represents how pleased the agent is given a certain event, and for every action the agent takes, we correlate it directly to the confidence, reducing its impact by 50%. The arousal scale impacts the agent's own perception of how excited it is when performing an action. Again, to modulate the weight of the agents' own confidence in its actions, every action will give the agent value of 0.5 (neutral excitement) and will be modulated by the agent's own assessment. As closer as the confidence is of 1, as higher is the excitement modulation that the agent perceives. The opposite happens when confidence is less than 0.5. Obtaining a victory at the end of the game should give the agent a clear pleasure and arousal signal, and thus, have to be the strongest signal that the agent receives.

The proposed different modulation for pleasure and arousal presented here are to be taken as one perspective on how to interpret the impact of the confidence. Different modulations can be designed and derived, but it will not be the focus of this paper. The use of the proposed modulation will not impact our final considerations.

D. Assessing my Opponent's Mood

The same way an agent can assess its mood, it can also assess its opponents' mood. This is important to give us a complete information about the agent's perception of the entire competitive game, taking into consideration how it assesses its actions, and how it assess the others' actions. To achieve the others' assessment, we propose the use of an estimated confidence (E-Confidence), which is calculated using a partial estimation of the opponents' hands.

To calculate its self-confidence, the agent uses the full game state, composed of the cards it has at hand and the board. When observing an opponent, the agent does not have access to the opponent's hand, only to the cards on the board. The agent can, however, compose an estimated hand based on the amount of cards the opponent already discarded and the current cards the opponent played on the field. The estimated hand is composed of 17 cards, and each of them is calculated as follows:

$$card = \begin{cases} 0 & if \quad noCard \\ d & if \quad discarded \\ r & if \quad cardAtHand \end{cases} \tag{11}$$

where d represents the discarded cards by the opponent on that round, if any, and r is a randomly chosen card, between the face value interval $[1, 11]$. To normalize the estimation, we create 100 different combinations of $(e - C)$. Knowing which action the opponent took, the agent will calculate the estimated confidence value for that specific action for the 100 combinations. This will give the agent an approximated assessment of the opponent's action, based on its judgment.

For each of the agent opponents, we create a single GWR to represent an estimated mood. For each opponent's action, we create a specific $e - PA$, following the same procedure demonstrated in Eq. 9 and 10, and train one specific GWR for that opponent. The entire process provides each agent with its own mood readings and the estimated mood readings of each of its opponents. Figure 4 illustrates the entire processing flow of the *Moody framework* , illustrating how the Player 1 obtains its mood reading and the e-mood reading for one opponent.

IV. Experiments

We perform two rounds of experiments in order to address our two main research questions: 1) How the mood readings provide the understanding of the agents' behavior on the Chef's Hat game; 2) How close is the representation of the mood estimated by an agent about its opponents from the real mood of each of these opponents.

A. Mood vs Confidence

Our first experiment aims to demonstrate the capabilities of the *Moody framework* to describe the performance of an agent when making actions on the Chef's Hat game. To do so, we set up one game and put a DQL-based, a PPO-based and 2 dummy agents to play against each other. We collect the confidence and mood values for the DQL and PPO agents and exhibit how they describe each other's behavior during the game. We also collect and exhibit the estimated confidence and estimated mood that each of these two agents calculated about each other. To illustrate further the capabilities of the mood, we also exhibit the same measures for 10 games played in a row.

B. Self vs Estimated

To evaluate how well an agent can estimate the performance of another agent within the same game, we run 100 games where 2 DQL-based agents play against 2 PPO-based agents. We calculate the correlation between the mood and confidences and the estimated mood and confidences obtained by each of the agents about themselves. Using two agents with the same learning algorithm playing against each other will

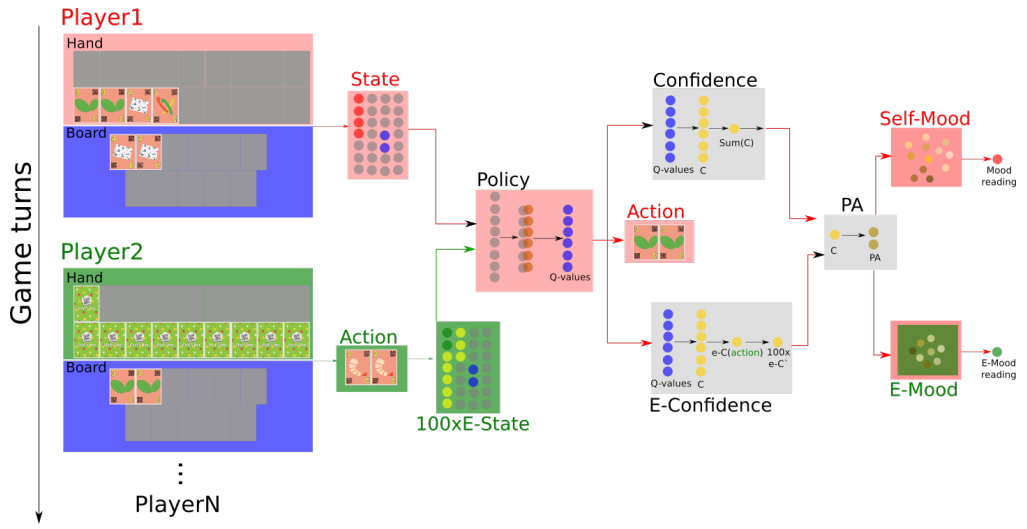

Fig. 4. Illustration of how Player 1 (boxes with the red shade) obtains its own mood reading and the estimated mood reading of Player 2 (boxes with the green shade). Player 1 calculates its own states (represented by the remaining cards at hand (in red), the cards on the board (in blue), and the empty card slots (in gray)), estimates its Q-values and confidences, updates its mood and selects an action, so the board is updated. Player 1 estimates the state of Player 2 by using the cards in the board (blue dots), the discarded cards (green dots), and by estimating 100 combinations for each of the missing cards (light-green dots). It calculates the Q-values for each of the e-states, and subsequently the e-confidences which are used to update the e-mood for Player 2.

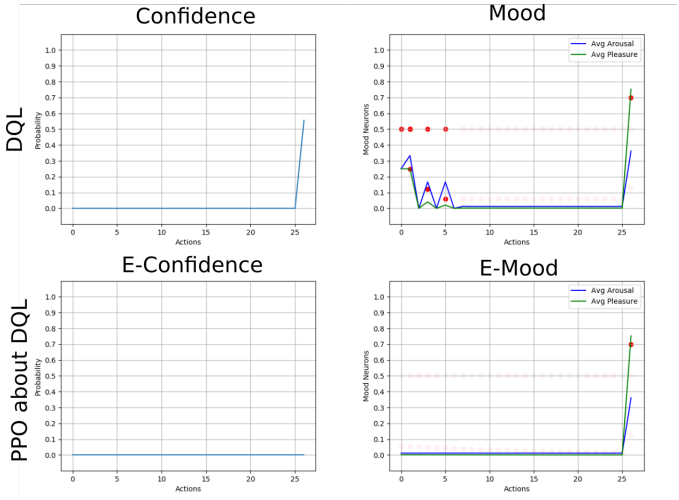

Fig. 5. Confidence and mood of the DQL-based agent, and e-confidence and e-mood estimated by PPO-based agent for 1 game won by the DQL-based agent.

allow us to evaluate also the impact of the learned strategies on how to play the game on the estimated values.

V. Results and Discussions

A. Mood vs Confidence

The winner of the single match was the DQL agent. It won after performing 26 actions. The confidence and mood of the DQL-based agent are exhibited in Figure 5, together with the estimation of the DQL confidences and mood obtained by the PPO-based agent. It is easy to see the inherent behavior enforced by the DQL algorithm on showing high confidence on actions that happen by the end of the game, as explained in the analysis made by Barros et al. [19]. Most importantly, we observe that, in this game, the last action performed by DQL was assessed as having strong confidence. The mood readings map this behavior clearly, presenting a higher pleasure and arousal reading by the end of the match, which is enhanced by this agent winning the match.

The estimation made by PPO-based about the DQL-based agent, at first, seems different from the estimations obtained by DQL-based agent itself. However, once understanding how the PPO agent learns to play the game, as explained in detail by Barros et al. [19], the estimations seem to make more sense. The PPO-based agent develops during training a strategy that enforces it to have higher confidence in its actions at the beginning of the game when compared to the end of the game. That means the PPO-based agent evaluates the actions from the DQL-based agent using its understanding of what good actions are. And thus, presents a very particular version of DQL-based agents' confidence, disregarding the last action as having a higher impact.

Doing the same observation on the plots about the PPO-based agent, illustrated in Figure 6, we find the same behavior. The DQL-based agent estimates the PPO-based agent's confidence based on its interpretation of what a good action is, and thus, creates its version of the PPO-based agent performance. In both cases, however, it is much easier to understand the behavior of each of these agents during the game by observing the mood readings. Even when observing the estimated readings, we can clearly identify how well these agents are doing in the game, and in which point it happened a turnover that allowed the DQL-based agent to win the game, at around action 25.

The advantages of using the mood readings to represent an agents' performance instead of the confidence values become much clearer when we report the results of playing 10 games in a row in Figure 7. By plotting the confidence readings over all these matches, the lack of temporal momentum in-between the actions and games is clearly evident. When observing the mood, however, we can clearly identify when the DQL-based agent is performing well, and win the matches mostly in the

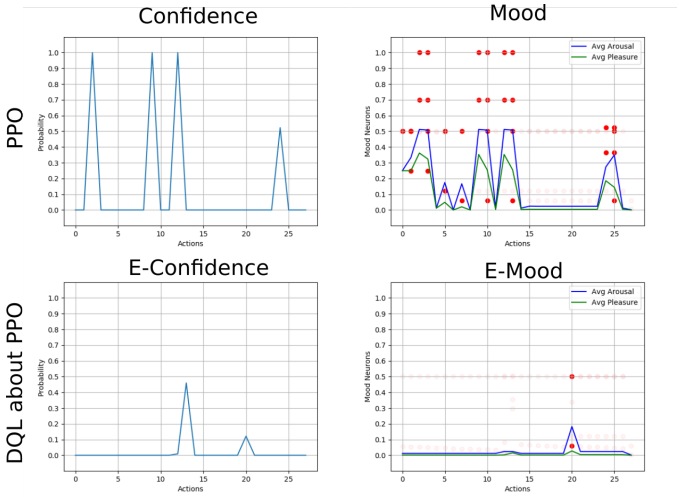

Fig. 6. Confidence and mood of the PPO-based agent, and e-confidence and e-mood estimated by the DQL-based agent for 1 game won by the DQL-based agent.

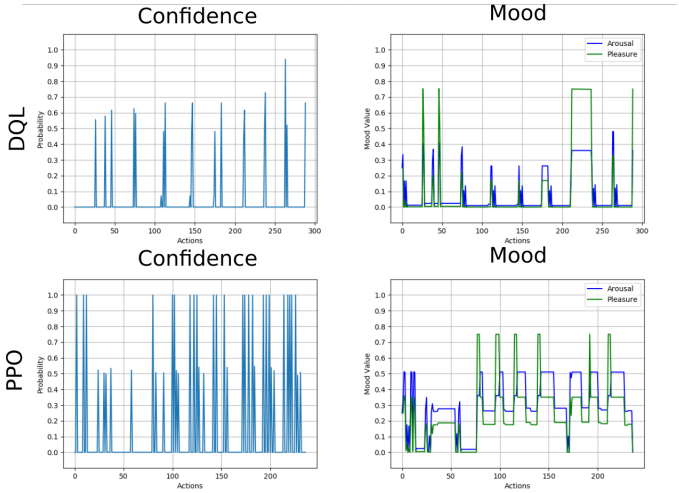

Fig. 7. Confidence and mood of the DQL-based agent and the PPO-based for 10 games, 3 of them won by the DQL-based agent and 7 won by the PPO-based agent.

beginning of the games, while the PPO-based agent has a winning streak by the end of the 10 games.

B. Self vs Estimated

In order to give each agent its own closed-world observation of the entire Chef's Hat game, it is extremely important that the estimated readings are somehow trustworthy. Of course, given that each agent learns to derive playing strategies differently [19], it is clear that their assessment of each other's actions are different. Calculating the correlation of the mood readings and confidence values after running 100 games with two pairs of incarnations of agents that follow the same strategy allow us to validate the agents' estimations. Table I reports all the calculated correlations.

When the same types of agents are estimating themselves, DQL1 and DQL2, and PPO1 and PPO2, it is clear that there is a strong correlation between the estimations. The mood estimation, however, presents a general higher correlation than

TABLE I

CORRELATIONS BETWEEN ESTIMATED CONFIDENCE AND MOOD READINGS (AVERAGED BETWEEN PLEASURE AND AROUSAL) AND SELF-ASSESED CONFIDENCE AND MOOD FOR 100 GAMES PLAYED BY TWO DQL-BASED AGENTS VS TWO PPO-BASED AGENTS.

Estimated Confidence				
	From			
Target	DQL1	DQL2	PPO1	PPO2
DQL1	1	0.86	0.09	0.09
DQL2	0.77	1	0.12	0.11
PPO1	0.12	0.12	1	0.86
PPO2	0.04	0.04	0.86	1
Estimated Mood				
	From			
Target	DQL1	DQL2	PPO1	PPO2
DQL1	1	0.84	0.12	0.16
DQL2	0.88	1	0.24	0.13
PPO1	0.24	0.19	1	0.89
PPO2	0.10	0.08	0.92	1

confidence. This happens because of the temporal correlation that happens on the GWR training. Due to the noise in the estimations, in particular at the beginning of the match - as there are many more cards to be estimated - the confidence values can show many different readings. The prototype neurons of the GWR smooth these differences by approximating the inputs towards a single BMU representation, which guarantees a much closer estimation.

Although the different learning algorithms have a different assessment of each others' actions, which is clearly observed by the lower correlations between DQL-based and PPO-based agents, they show a general correlation trend. Here, the mood also provided a higher correlation, mostly due to the same reason: the GWR approximates the correlation, and independently of the learning algorithm, it is easier to track the general performance of the agent.

C. Why the Estimations matter?

When analyzing the mood and estimated moods of a single agent, we are able to have an insight into how each of the agents is performing during one game, at an action-selection time. This is clearly demonstrated when visualizing the estimations of DQL-based and PPO-based agents about the dummy agents when playing the single game of the *Self vs Estimated* experiment, illustrated in Figure 8. It is possible to understand the random behavior, and how it does not contribute at all for winning the game, from both DQL-based and PPO-based agents' perspectives.

Giving to each agent the capability of measuring not only the impact of its own actions but also the estimation of other agents' actions, allows them to have a complete understanding of the game scenario. This gives the agent an important tool that, although not explored in this work, would allow them to interpret any other agents' actions and take this into consideration when performing their own.

VI. CONCLUSIONS

In this paper, we introduced the novel *Moody framework* which is able to explain the behavior of reinforcement learning agents in a competitive multiplayer card game scenario based on the players' assessment of its own performance.

The model is represented in a pleasure and arousal (PA) scale and builds on the introspective confidence [15] representation of the Q-values selection. It implements a Growing-

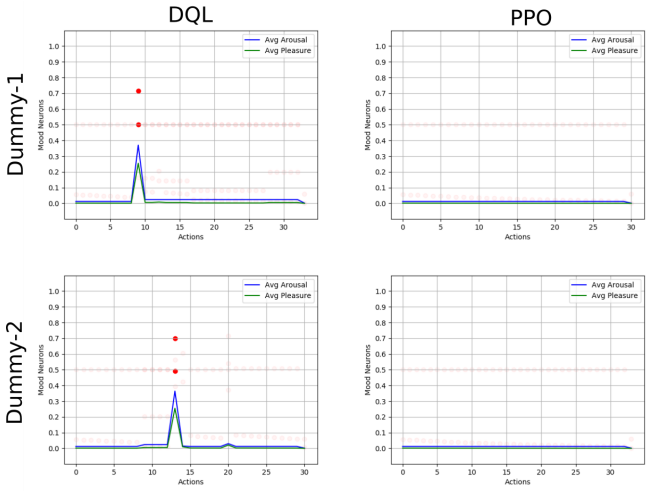

Fig. 8. Mood estimation of the DQL and PPO agents when playing one game against two dummy agents.

When-Required (GWR) network to establish a temporal impact between the assessment of the taken actions. Also, the model allows each agent to measure their opponents' actions based on their own assessing, endowing them with a closed-world representation of the entire game. We demonstrate how the *Moody framework* provides a much more enriching explanation of the agents' performance while playing the Chef's Hat card game than the introspection-based confidences. Also, in our experiments, we quantify how well an agent can assess others' actions based on their own judgment.

In general, we see this work as the basis for the implementation of intrinsic-aware agents towards competitive reinforcement learning scenarios. We envision the integration of the mood readings towards an action-selection process in order to modulate the agents' performance based on its own understanding of how well its opponents are performing. Also, we will investigate in the future the integration of intrinsic personality traits in the agents' mood, in order to model unique social relations.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[3] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2034–2039.

[4] E. Ponomarev, I. Oseledets, and A. Cichocki, "Using reinforcement learning in the algorithmic trading problem," *Journal of Communications Technology and Electronics*, vol. 64, no. 12, pp. 1450–1457, 2019.

[5] T. L. Meng and M. Khushi, "Reinforcement learning in financial markets," *Data*, vol. 4, no. 3, p. 110, 2019.

[6] R. N. Haksar and M. Schwager, "Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1067–1074.

[7] T. K. Yu, Y. M. Chieh, and H. Samani, "Reinforcement learning and convolutional neural network system for firefighting rescue robot," in *MATEC Web of Conferences*, vol. 161. EDP Sciences, 2018, p. 03028.

[8] F. Cruz, R. Dazeley, and P. Vamplew, "Memory-based explainable reinforcement learning," in *The 32nd Australasian Joint Conference on Artificial Intelligence (AI2019)*, 2019, pp. 66–77.

[9] D. Gunning, "Explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, 2017.

[10] X. Wang, Y. Chen, J. Yang, L. Wu, Z. Wu, and X. Xie, "A reinforcement learning framework for explainable recommendation," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 587–596.

[11] A. Holzinger, C. Biemann, C. S. Pattichis, and D. B. Kell, "What do we need to build explainable ai systems for the medical domain?" *arXiv preprint arXiv:1712.09923*, 2017.

[12] S. Anjomshoae, A. Najjar, D. Calvaresi, and K. Främling, "Explainable agents and robots: Results from a systematic literature review," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1078–1088.

[13] R. Goebel, A. Chander, K. Holzinger, F. Lecue, Z. Akata, S. Stumpf, P. Kieseberg, and A. Holzinger, "Explainable ai: the new 42?" in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. Springer, 2018, pp. 295–303.

[14] R. K.-M. Sheh, "" why did you do that?" explainable intelligent robots," in *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[15] F. Cruz, R. Dazeley, and P. Vamplew, "Explainable robotic systems: Understanding goal-driven actions in a reinforcement learning scenario," *arXiv preprint arXiv:2006.13615*, 2020.

[16] L. Fridman, J. Terwilliger, and B. Jenik, "Deeptraffic: Crowdsourced hyperparameter tuning of deep reinforcement learning systems for multi-agent dense traffic navigation," *arXiv preprint arXiv:1801.02805*, 2018.

[17] H. Xu, K. Paster, Q. Chen, H. Tang, P. Abbeel, T. Darrell, and S. Levine, "Hierarchical deep reinforcement learning agent with counter self-play on competitive games," 2018.

[18] M. McKenzie, P. Loxley, W. Billingsley, and S. Wong, "Competitive reinforcement learning in atari games," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2017, pp. 14–26.

[19] P. Barros, A. C. Bloem, I. M. Hootsmans, L. M. Opheij, R. H. Toebosch, E. Barakova, and A. Sciutti, "The chef's hat simulation environment for reinforcement-learning-based agents," *arXiv preprint arXiv:2003.05861*, 2020.

[20] P. Barros, A. Tanevska, and A. Sciutti, "Learning from learners: Adapting reinforcement learning agents to be competitive in a card game," *arXiv preprint arXiv:2004.04000*, 2020.

[21] P. T. Costa Jr and R. R. McCrae, "Mood and personality in adulthood," in *Handbook of emotion, adult development, and aging*. Elsevier, 1996, pp. 369–383.

[22] P. Barros, A. Sciutti, I. M. Hootsmans, L. M. Opheij, R. H. A. Toebosch, and E. Barakova, "It's food fight! introducing the chef's hat card game for affective-aware hri," 2020.

[23] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

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

[25] S. Marsland, J. Shapiro, and U. Nehmzow, "A self-organising network that grows when required," *Neural Networks*, vol. 15, no. 8–9, pp. 1041–1058, 2002.

[26] P. Barros and S. Wermter, "A self-organizing model for affective memory," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 31–38.

[27] P. Barros, G. Parisi, and S. Wermter, "A personalized affective memory model for improving emotion recognition," in *International Conference on Machine Learning*, 2019, pp. 485–494.

[28] G. I. Parisi, J. Tani, C. Weber, and S. Wermter, "Lifelong learning of humans actions with deep neural network self-organization," *Neural Networks*, vol. 96, pp. 137–149, 2017.

[29] G. Parisi, J. Tani, C. Weber, and S. Wermter, "Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization," in *arXiv:1805.10966*, 2018.

[30] L. Peña, J.-M. Peña, and S. Ossowski, "Representing emotion and mood states for virtual agents," in *German Conference on Multiagent System Technologies*. Springer, 2011, pp. 181–188.

[31] J. Zhang, J. Zheng, and N. Magnenat-Thalmann, "Modeling personality, mood, and emotions," in *Context aware human-robot and human-agent interaction*. Springer, 2016, pp. 211–236.

[32] M. Shvo, J. Buhmann, and M. Kapadia, "Towards modeling the interplay of personality, motivation, emotion, and mood in social agents," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 2195–2197.