Research on Tensor-Based Cooperative and Competitive in Multi-Agent Reinforcement Learning

Tsega Weldu Araya, Md Rashed Ibn Nawab, and A. P. Yuan Ling

Abstract — As technology overgrows, the assortment of information and the density of work becomes demanding to manage. To resolve the density of employment and human labor, machine-learning (ML) technology developed. Reinforcement learning (RL) is the recent advancement of ML studies. Multi-agent reinforcement learning (MARL) is useful to train multiple agents in the surrounding environment. The previous research studies focused on two-agent cooperation. Their data representation was held in a two-dimensional array, which is called a matrix. The limitation of this two-dimensional array appears as the training data of agents increases. The growth in the training data of agents creates storage drawbacks and data redundancy. Our first aim in this research is to improve an algorithm that can represent MARL training in tensor. In MARL, multiple agents are work together to achieve joint work. To share the training records and data of numerous agents, we need to collect the previous cumulative experience of agents in tensor.

Secondly, we will discover the agent's cooperation and competition, with local and global goals of agents in MARL. Local goals are the cooperation of agents in a group or team where we use the training model as a student and teacher agent. The global goal is the competition between two contrary teams to acquire the reward. All learning agents have their Q table for storing the individual agent's training data in an environment. The growth in the number of learning agents, their training experience in Q tables, and the requirement for representing multiple data become the most challenging issue. We introduce tensor to store various data to resolve the challenges for data representation in multiple agent associations. Tensor is expressed as the three-dimensional array, although it is an N-way array, which is useful for representing and accessing numerous data.

Finally, we will implement an algorithm for learning three cooperative agents against the opposed team using a tensor-based framework in the Q learning algorithm. We will provide an algorithm that can store the training records and data of multiple agents. Tensor advances to get a small storage size than the matrix for the training records of agents. Although three agent cooperation benefits to having maximum optimal reward.

Index Terms — Local and Global goal; MARL; Reinforcement Learning; Tensor decomposition.

I. INTRODUCTION

Reinforcement learning is a framework in which an agent learns by trial and error with the surrounding environment's interaction. As we see in Fig.1, the agent perceives the entire state of the surrounding location and takes action, and then the agent takes his reward that can be a positive or negative

reward; this leads to change into a new state. The agent obtains a scalar reward warning sign that assesses the quality of this transition. The function that indicates the action to take in an actual state is named policy. The goal of the agent is to discover a policy that can maximize the total accumulated reward. In the Markov decision process (MDP) theory, single-agent interact with the surrounding environment, but the second agent is only part of that environment. Markov game is a stochastic game developed to resolve the limitation of MDP and the theory of game designed for multi-agents. A multi-agent system (MAS) [1] is a gathering of autonomous, cooperating entities sharing a familiar environment. The growth in the complication of numerous tasks of agents makes them challenging to handle through preprogrammed performance. However, the agent's mission is to find out a solution on their own learning skills.

Fig. 1. Representation of Reinforcement Learning.

Previous researchers who work in the Markov game region [2]-[4] focused on studying two-agent teams playing in contradiction of each other. Their research permits a single reward function that one agent tries to maximize, and the opposed agent tries to minimize; this is called a zero-sum game. Nevertheless, as the number of agents rises, the cooperative work or the joint action among agents opens a new door to the research. To resolve the limitation of the previous research area, we propose three agent relationships in one team to get the reward in contradiction of the opposed team in a stationary environment. Our framework is to implement the Q learning function with a representation of tensor in a three-dimensional array. Q learning is the preferred way for agents to train and learn how to perform optimally in Markovian domains. The objectives [5] of Q learning is to train agents in the action value function $Q^\pi(s, a)$ for policy π by minimizing the estimated loss. The policy is the crucial way to know agents to get the optimal reward using Q learning policy; this helps to identify what action is supposed to take and what circumstances. Q learning does not

Published on December 31, 2020.

Tsega Weldu Araya, Northwestern Polytechnical University, China.
(e-mail: tsegawe55@outlook.com)

Md Rashed Ibn Nawab, Northwestern Polytechnical University, China.
(e-mail: rashednawab@outlook.com)

A. P. Yuan Ling, Huazhong University of Science and Technology, China.
(e-mail: cherryyuanling@hust.edu.cn).

necessitate the idea of the surrounding location, and it can solve problems without stochastic transition, reward, and without demanding adaptation.

Tensor is [6] a multi-dimensional array of arithmetical values. However, we can term a tensor to be an N-dimensions. In this investigation, as Fig.2 shows, we can consider a 3-dimensional tensor that displays the same properties as a higher-dimensional. Tensor is a higher-order simplification of matrices, were a multi-way array, which can denote different types of variability in the higher dimension. This placement of agents in multi-dimension termed tensor. There are three models; each model represents one agent. The tensor $T_{i,j,k}$ denotes the model 1 agent indicating action i, where model 2 agent demonstrates action j and model 3 agent showing action k.

Fig. 2. Three-Dimensional Array.

Our paper emphasizes on implementation of the Q-learning algorithm with the improvement in an extension of a tensor. The Multi Q tensor contains the current Q-Tables of all the agents. The tensor factorization [7] consists of storing each episode; the Q-table for each agent involved in the computation with a 3-way array (Tensor) $MQT \in \mathbb{R}^{S \times A \times \text{No. Agent}}$ where the dimensions are the number of states, number of action, and number of agents on which [8] Tucker decomposition should be accomplished. After the factorization, the MQT tensor is emptied to accept new Q-tables produced during the agents' learning.

II. BACKGROUND

Artificial Intelligence (AI) is a portion of computer science allocating through the simulation of intelligent presentation in a computer. It is a technology [9], making it possible for a machine to acquire experience by adjusting the input and executing human-like tasks. It involves observing the characteristics of human beings' intelligence and then relating them as algorithms in a computer to make a machine related to humans compared to their intelligence. Machine Learning is an arena [10] of AI, a technique that allows learning machine from data, deprived of being explicitly programmed. It is a solution used to find a statistical structure in the data given, and from there, it will come up with the rules for automating the task. RL is [11] a learning technology that is unlike supervised learning and non-supervised learning. RL allows machine and software agents to learn how to behave in an environment by performing actions and getting the related maximum reward. Its learning steps are to see the current state S, then choose an action A and execute it, after that the agent will receive instant reward R; finally, the agent perceives the new state S'. The RL was used in a single-agent

learning environment, but the number of agents opens new learning methods.

MARL is a learning discipline that emphasizes models that consist of two or more agents that learn [12] by dynamically cooperating with their environment. On the other way round, the complication of MARL scenarios rises with the growth in number of agents to train in the environment.

Markov decision process (MDP) studies have been used for single-agent interaction in an environment. MDP [13] is denoted by four-tuple (S, A, R, T) where S indicates a state; A represents as action, T indicates the state transition probability $T: S \times A \rightarrow S$, R is the reward function $R: S \times A \rightarrow r$. The agent's main aim is to discover a policy, whereas to maximize the estimated discount reward. A reward can be divided into a current or immediate reward and a future reward. However, in MDP [14], a single agent trains with one optimal policy π in a stationary environment, but the second agent can only be part of the environment, to fix this problem Markov game added as an extension.

Markov game is a stochastic game generalize as a natural extension of MDP to include multiple agents; it is a suitable mathematic tool to model dynamic interaction situations and permits multiple agents to interact with each other in an environment. Moreover, defined [15] as four-tuple (S, A, R, T) state S, actions $A_1, \dots, A_k$ for every agent in the environment, state transition $T: S \times A_1 \times \dots \times A_n \rightarrow S$, and each agent has its estimated reward $R: S \times A_1 \times \dots \times A_n \rightarrow r$. If the reward and the transition function is not identified, there is a Q learning algorithm used to solve the limitation of MDP. In RL, we need to find a function $Q(s, a)$ that calculates the best action A in state S to maximize a collective reward. This function predicted using Q-learning by repeatedly updates $Q(s, a)$ with the Bellman Equation. Q learning [16] is an algorithm used to implement when the MDP is with unknown reward and transition. Q-Learning algorithm is a model-free off-policy RL algorithm where the learner builds incrementally a Q-function which attempts to guess the discounted future rewards for taking action from given states. Q-values are [17] usually stored in a table. To learn the optimal policy π for an agent Q function wherein state S and action A, $Q: S \times A \rightarrow R$. The main impression of Q-Learning is to explore all opportunities of state-action pairs and estimate the long-term reward that will be received by applying an action in a state as:

$$Q(S, A) \leftarrow (1 - \alpha)Q(S, A) + \alpha(r + \gamma \max_{A'} Q(S', A')) \quad (1)$$

Where γ is a discount factor $0 \leq \gamma < 1$, and α is the learning rate ($\alpha > 0$).

III. LITERATURE REVIEW

Reinforcement learning is a massive area of ML. This research area is vast, and many researchers work on different techniques, but in this manuscript, we will discover some ideas of the researchers who have similar contributions to our research.

The author [18] focuses on learning an agent with

coordinated action to have teammate model and reward allotment. Teammate modeling adopts by observation to predict other teammate agents' actions. Their proposed work is limited to a two-agent teammate model: one prey and two-hunter agent's cooperation. They categorized the reward into two kinds; the first one is the immediate reward, which is executed after each action. The second is the goal reward, which is when the training agent reaches the goal state.

The author [7] proposed implementing RL and MARL algorithms by utilizing results from multi-dimensional data using tensor and tensor factorization. Their target is on presenting how to modify the existing Q learning algorithm using tensor factorization in their Q tables by training multi-agents to generate knowledge about the actions performed in the environment. The action taken by agents is to perform the collaborative effort of multiple agents in an environment to have a joint achievement. They discover many techniques of RL to prove the innovation of their work in tensor decomposition. Although they introduce and implement Tucker decomposition to remove the correlation between the Q tables.

This [19] paper gives a general overview of a tensor with its decomposition and its usage in ML. Tensor is a multi-dimensional array, which can be thought of as a data cube. They describe tensor decomposition in three models; a temporal data model is one of the tensor models that can be applied when new data arrives in a never-ending continuous stream, without having to deal with an infinite time dimension. The second model where the tensor arises naturally is the representation of multi-relational data. The third model is that the latent variable model in ML used tensor methods that have been effectively applied to hidden Markov models.

The [20] author developed higher-order tensor decomposition for the large-scale data analysis. In many applications, data are modeled as tensor or multi-dimensional arrays. They avoid sizeable matrix-matrix multiplication and exploit massive data sets' sparsity to minimize intermediate data and flops by sequentially computing the intermediate matrices and generating the intermediate tensor vector wise. Although they describe the CANDECOMP/PARAFAC (CP) and Tucker decomposition.

The authors of those papers [21]–[24] briefly describe the tensor network's advantages in ML and big data. Growth in technology brings difficulty in storage and runtime of the memory. With the increase in the dimensions of tensor, some useful values were unseen. However, a tensor network is used for dimensional reduction, finding hidden structures in large-scale data, handling missing values, and removing noisy data. Tensor network is an efficient way to access larger order tensor where tensor train network and hierarchical tensor network are most recommended methods. They stated the algorithms how to implement from high order tensor to lower-order tensor.

Those papers [25], [26] elaborates that the tensor train network is stable, and its computation is built on a lower rank approximation of auxiliary unfolding matrices. They define the new form of unfolding matrices as a clear and convenient way to implement all basic operations efficiently. They provide the algorithms and graphical representations of tensor

train networks from high order tensor to lower order.

The coordination of two agents is useful to transport an object from one place to another. This transportation of items [27] describes the coordination of MARL where agents are trying to avoid obstacles to reach their given task at the minimum time state. They used [28] two cooperative agent teammates involving joint action to transport an object. The agents have to transport a target object to the home base in minimum time. The first step to transport the item is the agents must have to grasp the item, and then they need to reach the target place, although both agents must have the coordination to pull the object in the same direction.

This paper [3] proposes MARL main idea in Markov games. Markov game allows two agents with different goals to share the environment. This game is stochastic as they are considering two opposed agents playing each other, but their expected reward is single. When one agent wants to maximize, and the opponent agent tries to minimize, then the algorithm called minimax Q learning. Minimax algorithm used to the zero-sum Markov game of agents' competition. The optimal policy is the policy that can maximize the expected sum of the reward, where they use optimal policy to find the matrix game where two agents play against each other. The agent i try to maximize its expected reward, but the opponent agent j tries to minimize it.

The author of those [29] and [30] papers focus on robot soccer on MARL. Their focus is on how to improve shooting abilities for RoboCup League domains. The experiment that they implemented shows the result of their proposed work in the RL algorithm, which was the training of agents to acquire good shooting skills in real-time. Their newly implemented algorithm is called the Temporal Difference algorithm with the value iteration function and the state-value functions. The main purpose of a soccer game is to score goals.

Concurrently [4] to our work, they describe coordination and competition of MARL. Their framework is implemented by zero-sum Markov game to compete with an opponent agent team, and they implement how the teammate Markov game can accomplish the two-agent cooperation. In their proposed work, there are two competitive agent teams, and each agent team has its team commander to be responsible for making a decision. The team commander arranges different roles for all its members. Their simulation is about robot soccer how the agents can cooperate in the environment to enhance the local goal and the overall association to compete with their opposed team. They use two kinds of Markov game technique that is zero-sum Markov game used for competition between agent teams and team Markov game is used for teammate cooperation. Zero-sum Markov game is the model used for the opposing goal were used to get the global goals due to one team agent needs to maximize its reward other team agent needs to minimize its reward. Minimax Q learning algorithm is a unique feature designed for zero-sum games. While in minimax Q learning, team Markov game has single reward all the teammate needs to maximize their teams' goal. Every team member's objective is to get their team's desired reward instead of their individual benefit.

IV. METHODOLOGY

As we know, RL is the arena of machine learning used to solve challenging problems; most successful works have been done on learning a single agent. The increase in the number of agents derives MARL. MARL is the technology of interacting with many agents to provide joint work. The combination or the collection of agents in a coordination system is to have combined output, have united effort, and gain some techniques to defeat the opposed team. While in the comparative behaviour of agents, usually, they are standing against the other team to get their maximum reward. Due to the limitation on the number of cooperative agents and competitive agents in previous works on Markov game and Q learning in the stationary environment as well as in learning multi-dimensional information, we propose to use a tensor-based Markov game framework in MARL.

Fig. 3. Multi-Agent Coordination Framework.

This project's framework is presented in Fig. 3 comparative and cooperative Markov game that adapts the relationship among three agent teammates against the opposed team, which is competitive goals in an environment. The recommended task is to represent three agents' cooperation in a multi-dimensional array by training the agents in an environment. Every agent has its Q table according to the number of state and the number of actions in every step. Team commander has the responsibility to provide different tasks to agent teams according to their learned skills in Multi Q table. To represent the three agent teammates association, we suggest using the Multi Q learning algorithm, which is the idea of tensor factorization. Tensor is the representation of a Multi-dimensional array. Multi Q tensor is the collection of Q tables for all agents.

Fig. 4. Robot Soccer.

Our case study is about robot soccer playing by eleven agents as a teammate against another team, as shown in the

above Fig. 4. To implement this scenario, we use three agent relationships in a team.

Each agent learning process is stored in the Q table as the agent takes every action and state. The Q table has a matrix form with rows and columns. To combine the knowledge of multiple learning agents' data in a Q table, we need to use tensor. Tensor is multi-dimensional generalizations of matrices. This project aims to represent the agent's data in 3rd order tensor and learn how to cooperate with three agents in a team to achieve the goal.

Tensor [31] is a multi-dimensional array of geometric values (datasets) and generalizes matrices to multiple dimensions. The multi-dimensional array is titled as tensor; however, it can be expressed as one dimension like vector, two dimensions like matrices, three dimensions like tensor, and more than three dimensions called the higher-order tensor.

To perform this task, we extend the Q learning algorithm with [32] tensor factorization. Q-Learning is the Q function model, which is useful in value-based reinforcement learning algorithms to discover the optimal action-selection policy. The main purpose is to get the maximum Q value function. Q-table has a matrix illustration where every row denotes a state (s) and every column as action (a). The primary value of the Q table is 0; however, then Q values are updated after training. Q table helps us to discover the best action for each state, and it helps to maximize the expected reward by selecting the best of all possible actions. The values that are kept in the Q-table are entitled as a Q-values, and it is plotting according to a (state, action) arrangement. The goal [33] is to find an approximate optimal action-value function called the Q-value function. The Q-value function is stated as the estimated amount of future reward obtained by taking action in the current state. $Q(\text{state}, \text{action})$ returns the predictable future reward of that action and at that state. Bellman equation is a function that uses to estimated Q-Learning, with repeated updates of $Q(s, a)$. Initially, [29] agents are exploring for all possible action in an environment, then they choose the appropriate action to acquire the reward; finally, they update the Q-table. When the Q-table is prepared, the agent starts to exploit the environment and begin to take better action.

A. Q-Learning Algorithm Process

Fig. 5 describes the hierarchical description of the Q-learning algorithm.

Fig. 5. Q-Learning Algorithm Process.

1) Step 1: Initialize the Q-Table

We will first build a Q-table for each agent training data in a matrix form. There are n columns, where the n= number of actions. There are m rows, where the m= number of states.

Primarily all the values given in the Q-table are zeros.

2) Step 2: Choose an action

This is the second step of the agent's training in the Q learning process. This process goes on until the training time terminates, or the training loop ends as the agent reaches his designated goal. The agent will select an action (a) in the state (s) based on the Q-table.

3) Step 3: Perform an action

To perform an action of agents in an environment, we will use some epsilon-greedy policy techniques. The epsilon-greedy policy is a means of choosing random action with uniform distribution from a set of existing activities. This policy indicates that either agent can select an action with arbitrary action using epsilon probability or choose an action with the 1-epsilon possibility, which can provide a maximum reward in a state. In the beginning, the epsilon rate is higher. However, the agent tries to explore the surrounding environment and randomly choose actions. The logic behind this is that the agent does not know anything about the environment. As the agent discovers the environment, the epsilon rate decreases and the agent starts exploiting the surrounding environment. During the procedure of exploration, the agent gradually becomes more assured in estimating the Q-values.

4) Step 4: Measure the reward

According to the given Q table data, we are able to calculate the change in Q value $\Delta Q(s, a)$ to measure the reward. The reward in Q learning has an immediate reward and future reward. However, we add the initial Q value to the $\Delta Q(s, a)$ multiplied by an earning rate. To maximize the expected reward by selecting the best of all possible actions. Function Q (state, action) returns the estimated future reward of the indicated action at the known state. The agent's game to reiterate the scoring/reward structure is +1, -1, and 0.

5) Step 5: Update Q table

As indicated in fig. 8; this process is the final step of the Q learning algorithm process. To update a Q-value of the trained state, we need to focus on the observed reward and the maximum possible reward for the following state. The updating is done through the formula and parameters described in performing an action. After taking action and observing the outcome and reward, then we need to update the Q function.

B. Comparative Framework

However, this paper explains the use of a multi-agent team, which means each group has its own team commander. The team commander helps to control his team's overall actions and provide tasks according to the individual agent's talent. Due to the multiple agent members, we implement the multi Q learning algorithm. In multi Q learning, every agent has its own Q table and Q value. However, the team commander gets every agent training history or ability from the stored data. To access the stored data or represent the enormous amount of training data, we need a multi-dimensional array called a tensor. Minimax multi Q learning algorithm is used in this scenario, as shown in the diagram (Fig. 6) below, where the team commanders can easily observe agents' training data.

Fig. 6. The Role of Team Commander.

C. Cooperative Framework

In cooperative MARL, the accurate expected reward is obtained due to the optimal joint actions. In this coordinate framework, agents try to cooperate with their actions to get their desired optimal collective action. We will discuss the three agent coordination in a robot soccer game to get the most accurate reward. This learning system helps agents communicate easily and have common goals, which helps to have common knowledge. This association of three agent training system allows agents to have a united local goal. According to the increasing in the coordination of agents to have joint work, we need to store their training data in multi-way dimension. This multi-way dimensional array we call it tensor. To achieve the multi-agent cooperation's designated goal, we need to use a multi Q table and multi Q value to store and access the agents' different talent history. This elaborates every agent has its own action and state, which means tensor is useful to store the multiple agent association. To demonstrate the agents learning as the three agents' relationship, we label it as a local goal and a global goal of agents.

D. The Proposed System Schema Overview

The design and implementation of MARL in cooperation and competition will be displayed in the diagram below. As we see in Fig. 7, the local goals' flowchart of how the three agents cooperate with each other to perform joint action. This flowchart shows how the agent interacts with the environment to improve his training and how the agents exchange advice in order to have a collective work. As we see in the diagram, a student agent has their own training skill and action value but to cooperate as a teammate with other agents, and they need to communicate with the teacher agent to ask for advice. In this case, the teacher agent will give a response, according to the student agent's request. If the teacher agent doesn't have any advice, then the student agent can use his original action.

Fig. 7. Flowchart of Local Goals.

In Fig. 8, we see the whole system, how it connects each other, and how it works. All agents in RL, in the beginning, they do not have any experience, or they do not have any initial knowledge. However, every agent has his own Q table

at the training stage to store his learning action at every state. As we see in the flowchart below, we use tensor to keep all the Q tables of several agents' training experience. The team commander (MQT) can access the training experience of every agent from the tensor. Each team in robot soccer has its own team commander to control the game and divide the task to agents according to their training experience. Finally, those two teams play against each other to get the optimal reward.

Fig. 8. Flowchart of the Whole System.

E. Task Level Learning

Our framework assumes every agent as a student and a teacher interchangeably; however, as we have shown in the diagram (Fig. 9), we will elaborate the case as request advice agent i as a student and the joint advice response agent j and k as a teacher. Agent, i at certain task-level learning, can use its original action to perform his task, but to get an optimal policy to maximize the local goal agent, i can choose to send requests for advice from his teammates.

Using student policy π_s , the agent, i think whether to ask for advice from agent's j and k or no. If the agent i decide to send a request, then his teammate agents j and k will check the teacher policy π_t and task level policy to decide what can they respond to the advice according to agent i request. Then the agent i executes the advised policy that comes from joint action a^j and a^k . Finally, the agent i updates his task level policy; the advantage of advising policy is that the agent can study how to use the local joint knowledge to improve the teammate goals.

Concurrently to previous works, our model is also on when/what to advise actions using three-agent association. Task level policy is used for learning agents how to coordinate in the environment to maximize their performance in a given task. Advice level policy is used to advise their teammates to be well skilled and get the maximum reward where the agents are not experts, but this policy shows how to advise effectively in the process of task-level policy to accelerate learning.

Fig. 9. Student and Teacher Policy.

V. EVALUATION METHOD

The developed algorithms are verified that cooperation among three agents increases the possibility to get the maximum reward. The training data of these three agents'

collaboration can be represented in tensor due to the storage and accuracy issue. To perform a joint work of multiple agent teammates, they store their training data in MQT. To model the cooperation among agents, we use the student and teacher policy.

A. Algorithms on Local Goal of Agents

Algorithm 1. Coordination Learning Algorithm

```

for Phase II episode  $\tilde{e} = 1$  to  $\tilde{E}$  do
    Initialize task-level policy parameters  $\theta$ 
    for Phase I episode  $e = 1$  to  $E$  do
         $o \leftarrow$  initial task-level observation
        for task-level timestep  $t = 1$  to  $t_{end}$  do
             $\tilde{o} \leftarrow$  GETJOINTADVISEOBS( $o, \theta$ )
            for agents  $\alpha \in \{i, j, k\}$  do
                Exchange advice  $\tilde{a}^\alpha$  via advising policies
                if No advising occurred then
                    select action  $a^\alpha$  via local policy  $\pi^\alpha$ 
                end
            end
             $\tilde{a} \leftarrow \langle \tilde{a}^i, \tilde{a}^j, \tilde{a}^k \rangle, \tilde{a} \leftarrow \langle \tilde{a}^i, \tilde{a}^j, \tilde{a}^k \rangle$ 
             $r, o' \leftarrow$  Execute action  $\tilde{a}$  in task
             $\theta^i \leftarrow L^i(\theta^i, \langle o', a^i, r, o^i \rangle)$ 
             $\theta^j \leftarrow L^j(\theta^j, \langle o^j, a^j, r, o^j \rangle)$ 
             $\theta^k \leftarrow L^k(\theta^k, \langle o^k, a^k, r, o^k \rangle)$ 
             $\tilde{o}' \leftarrow$  UPDATEJOINTADVISEOBS( $o', \theta'$ )
             $\tilde{r}_t^i, \tilde{r}_t^j, \tilde{r}_t^k \leftarrow$  Compute advising-level rewards
            Store  $\langle \tilde{o}, \tilde{a}, \tilde{r} = \tilde{r}_t^i + \tilde{r}_t^j + \tilde{r}_t^k, \tilde{o}' \rangle$  in buffer  $\tilde{M}$ 
        end
    end
end

```

The developed algorithm enlightens about agents' local goals and how they behave to coordinate each other, where agent i, j , and k learn joint work to perform their given tasks. This learning algorithm is suitable to develop a student agent and teacher agent's coordination learning algorithm. In every learning episode, those agents interrelate with the environment, and they try to enhance their skills by collecting data from the given algorithm. For every agent trains to execute their task by using the task level policy π , the student policy π_s is used to request action advice, and the teacher policy π_t is used to respond to the asked action advice. In this implemented algorithm, every agent can use the character of the teacher and student position. However, first, we assign for the agent, one who has a ball at the current time is always requesting advice from his teammate. The second scenario is the agents without a ball at the current game are the advice-givers by arranging their positions. The student agent i execute the joint action advice a_j and a_k that he got from his team's response instead of his original action. This learning method we call it local goals, which helps the agents transform the local knowledge into task level policies to get actionable advice.

B. Algorithms on Global Goal of the Agents

This developed algorithm for the global goal of agents has verified that the combined knowledge of those agents in a team improves the overall performance of a system. Besides, we clearly state the training data representation and the general knowledge for every agent in a tensor. Tensor is useful to combine or to store the Q table of each agent training action.

Furthermore, the main objective of agents training is to look for a policy that can maximize their related reward. To have continuing action with a successful goal, agents need optimization for long run reward by focusing on the current

or short-term reward. As we see in the above algorithm implemented to show the Q learning process, which is useful to show the long-term reward of agents and the minimax Q learning algorithm which is used to the competition of two-agent teams against the opposed team. The multi-agent Q learning algorithm represents the multiple Q table of each agent training experience in tensor. MQT is used tucker decomposition to get access to the training data without any duplication.

Algorithm 2. Multi Agent Q learning

```

Init Multi Q(s,a,o),  $s \in S, a \in A, o \in O$ 
Init  $\theta(a) = 1, a \in A$ 
Set number of Q Tables to factorize, nrqtuples
Init MQT(s,a,o)
for  $i \leftarrow 1, \text{max episodes}$  do
  for  $j \leftarrow 1, \text{Agents}$  do
     $s_t \leftarrow \text{current state}$ 
     $a_t \leftarrow \text{greedy}(s)$ 
    execute  $a_t$ , observe  $s'$  and the reward  $r$ 
     $Q_t \leftarrow \text{MultiQ}(:, :, j)$ 
     $Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) - \alpha_t w_t(a_t) [r(s_t, a_t) + \gamma \max_a [Q_t(s_{t+1}, a)] - Q_t(s_t, a_t)]$ 
    Multi Q(:, :, j)  $\leftarrow Q_{t+1}$ 
    MQT(:, :, mod(i, nrqtuples), j)  $\leftarrow Q_{t+1}$ 
  end
  if mod(i, nrqtuples) == 0 then
    MQT  $\approx \text{Core} \times_1 S_p \times_2 A_p \times_3 O_p$       Tucker Decomposition
     $w_t \leftarrow |A_p| \text{row} / |A_p|$ 
    indexmax  $\leftarrow \max(|O_p| \text{row})$ 
     $Q_{\max} \leftarrow \text{MultiQ}(:, :, \text{indexmax})$ 
     $o \in O, \text{MultiQ}(:, :, o) \leftarrow Q_{\max}$ 
    reinitialize MQT
  end
end
end

```

Tucker decomposition is one of the decomposition methods used in tensor factorization or tensor decomposition. It is used for analyzing higher-order tensor into the matrix to remove the unused correlation or duplication of elements in the representation of tensor. This decomposition works by decomposing tensor into core tensor, were used to multiply by matrix alongside with mode. The main advantage of Tucker decomposition in MARL is for the use of Q table at every episode and Q tables of different team agents that correlate with each other, which is used for their whole team to share the trained data. Due to the Q table's representation as a slice of a tensor, this placement helps to remove the duplication of values in the Q table. As we are looking for implementing a tensor with MARL, the commonly used slices of the tensor are states of agents, agents' action, and the Q table of agents in one episode. The total training values of every agent in a team are stored on the multi Q tensor. The optimal action of agents can get after many training stages, and this helps to have long-term rewards; however, we use tensor to store the previous experience of multiple agents, and tensor helps agents analyze the ability of their action. Finally, tucker decomposition is also useful to eliminate noisy training data in the Q table.

VI. EXPERIMENTAL RESULTS

This experiment describes how the proposed system works and how it integrates different software with a computer system to perform the task. This experiment is done in robot soccer how the agents interact with each other to enhance the training ability and how to store the agents' training data. In this experiment, we will perform four activities, the first experiment is the general training of agents in RL using Q learning, the second is the cooperation among agents to get the optimal reward, the third is the competition of agents against the opponent team, and finally tucker decomposition

for the storage of a large amount of multi Q data in tensor without duplication of data. However, to show the cooperation of the three-agent relationship and to get the optimal reward, we use the student agent and the teacher agent, where teacher agents provide joint advice to the student agent.

A. Result Analysis

From the above experiment, we can see the results of this research project. In this section, we will compare the previously used method result and our proposed model result. To improve the traditional model, we plan to have a suitable storage mechanism, the accuracy of the local system training, and the time of the giving training data. In fig.10, we see the general training of any agent in reinforcement learning. As we see primarily, the agent does not know anything, so the negative reward is higher than the positive reward. After many trials and errors, the agent tries to improve his action, and his training value is stored in the Q table. The agent is in training until he gets the best action in the surrounding environment to perform a positive reward.

Fig. 10. General Training of Agents in RL.

In Fig. 11, we can see the difference between the traditional model of two agent association and the proposed model of three agent coordination. This result is compared according to the score or the way how to get the optimal reward. As we have shown below, the three agents' coordination receives a better score than the two agents' coordination. The advice that comes from two teacher agents is better because they are looking for the best action from a different angle view, so their joint advice is more accurate than one agent advice. In our model and the previous model, we use the student and teacher relationship. However, in our research, we have two teacher agents and one student agent; however, they use one student agent and one teacher agent in the traditional model. In general, thought the joint action or two mind thinking is better than one agent action.

Fig. 11. Cooperation of Agents.

A tensor can store multiple data dimensions, but a matrix can only store two-dimensional data. As we see in Fig. 12, the training data of numerous agents are stored in a tensor. Every agent has its own Q value and Q table with respective action and state. The learning experience of each agent Q table is stored in this tensor. However, to remove the correlation or

duplication of data in tensor, we use tucker decomposition. Tucker decomposition is useful to express the entries of a tensor by mode of matrices to reduce the vast amount of storage size.

Fig. 12. Tensor Representation of Multi Q Table.

The cumulative training data of multiple cooperative agents on storing their experience is the main problem of many current types of research. The previously studied researches were used as a matrix model to store their training data because the cooperation is among two agent associations. However, in our study, we use three agent associations, so we need to store this agent association using tensor because tensor can store multiple training data. As we see in Fig. 13, the comparison between agent training data in tensor and matrix. In two dimensional array, which means matrix, we cannot store the three agent associations, but we can store single-agent training data with action and state.

Moreover, we can store two agent associations in a matrix. So it means, as we see in the diagram, to store more than two agent association training data requires many matrix storages, but in tensor, we can keep the three agent association without any difficulties because tensor is multi-dimension. Also, as we compare the storage and accuracy, a tensor is better than a matrix because, in a tensor, we can store the three agent training data at the same time so that it requires a small storage size; however, in a matrix, we need many sizes in MB because we cannot store the three agent training data as the same time. To have the best action and get the optimal reward requires the joint action of many agent suggestions.

Fig. 13. Storage Requirement for Matrix and Tensor Representation.

VII. FUTURE WORK

MARL is the training of agents on how to interact with the environment and how to reach their target area. This paper proposes the Q learning algorithm using a tensor framework where three agents cooperate in playing robot soccer. The traditional model of a robot soccer Markov game is played by collaborating two agent teammates against the opponent team where their experiment was conducted in matrix form as it is a two-agent cooperation process. Our proposed model was the relationship between the three agent teammates to compete with the other team to win. To represent the three agent associations, we use a tensor to store our training data.

As the tensor dimension rises, we call it a higher-order tensor, where the tensor network is viral for dimensional reduction. So, in our future work, we will try to model tensor networks that are useful for conventional neural networks and auto-encoder to alleviate the curse of dimensionality. This tensor network helps identify the hidden values on a large scale of data and handle the missing values and noisy data.

VIII. CONCLUSION

RL is one of the learning techniques in machine learning. RL is unique from all current learning because most of them try to begin their learning from the previously stored data. However, in RL, every agent is trained at the moment without any previous experience and without any prior knowledge. This implies RL is the online training without any limitation to those agents, even though it is free role learning, but at the training stage, we have to tell the agents the action which cannot be done to get the optimal reward. As we are discussed previously, the agents can achieve their tasks by trial and error many more times. After taking some training, the agents can be able to differentiate which action can maximize their reward and which action can drag them to the negative reward. The training knowledge of the agents is stored in a Q table. When the agent first visits the environment at state s , then the agent looks at what action is supposed to take to get his reward. At this stage, the agent's achievement starts to store in the Q table to look for future rewards. Also, in multiple training agents' Q table, it is useful to share agents' training data to have joint action. MARL is the training of various agents on how to interact with the environment.

This paper covers the Q learning algorithm using a tensor framework where three agents cooperate playing in a team against the opposed team. Our focus was on playing agents in a cooperative and competitive scenario where each agent's training knowledge is stored in the Q table. To represent the three-array data for each agent, we need to use tensor. Tensor is useful to keep the training data of each agent's experience in Q table, to have an association between the learner agents where tensor is supportive of sharing the previously-stored experience of agents and of having the long term reward agents must have to follow their current reward. To conclude, this project proposed task is attained to represent the agent's learning experience in 3rd order tensor and learn how to cooperate the three agents against the opposed team to get the desired reward.

REFERENCES

- [1] R. Babu and B. De Schutter, "Multi-agent reinforcement learning : An Multi-Agent Reinforcement Learning : An," vol. 19, 2010.
- [2] J. Hu and M. P. Wellman, "Nash Q-Learning for General-Sum Stochastic Games," vol. 4, pp. 1039–1069, 2003.
- [3] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," 1910.
- [4] B. Fan, Q. Pan, and H. C. Zhang, "A Multi-Agent Coordination Framework Based on Markov Games," pp. 230–233, 2003.
- [5] S. Kapoor, "Multi-Agent Reinforcement Learning: A Report on Challenges and Approaches," pp. 1–24, 2018.
- [6] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," no. November 2007.
- [7] S. Bromuri, "A Tensor Factorization Approach to Generalization in Multi-Agent Reinforcement Learning".

- [8] P. Rai, C. Hu, M. Harding, and L. Carin, "Scalable Probabilistic Tensor Factorization for Binary and Count Data," 2014.
- [9] N. Kumar, "Ethical Aspects and Future of Artificial Intelligence," no. Iccics, pp. 111–114, 2016.
- [10] N. Jonsson and N. Jonsson, "Ways to use Machine Learning approaches for software development," 2018.
- [11] W. Qiang, "Reinforcement Learning Model, Algorithms and Its Application," no. 1, pp. 1143–1146, 2011.
- [12] M. Tan, "Game theory and Multi-Agent RL," *Mach. Learn. Proc.* 1993, pp. 330–337, 1993.
- [13] Y. Nagayuki, S. Ishii, K. Doya, and T. Corporation, "Multi-Agent Reinforcement Learning: An Approach Based on the Other Agent's Internal Model".
- [14] E. Even-dar, "Learning Rates for Q-learning," vol. 5, pp. 1–25, 2003.
- [15] C. H. I. Zhang and X. I. A. Z. H. A. N. G. Jiao-long, "A GAME-THEORETIC LEARNING MODEL IN MULTI-AGENT SYSTEMS o s," no. November, pp. 4–5, 2002.
- [16] D. Pandey, "Approximate Q-Learning : An Introduction," pp. 317–320, 2010.
- [17] S. Levine, "Lecture 9: Deep RL with Q-Functions".
- [18] Z. Pucheng, "Multi-agent Cooperation by Reinforcement Learning with Teammate Modeling and Reward Allotment," no. 4, pp. 1316–1319, 2011.
- [19] S. Rabanser and S. Günnemann, "Introduction to Tensor Decompositions and their Applications in Machine Learning," pp. 1–13.
- [20] L. Li and D. Boulware, "High-Order Tensor Decomposition for Large-Scale Data Analysis," pp. 11–14, 2015.
- [21] A. Cichocki, "Era of Big Data Processing : A New Approach via Tensor Networks and Tensor Decompositions," pp. 1–28, 2013.
- [22] A. Cichocki, "Tensor Networks for Big Data Analytics and Large-Scale Optimization Problems," pp. 1–36, 2013.
- [23] A. Cichocki et al., "arXiv: 1609. 00893v3 [cs. NA] 11 Sep 2017 Low-Rank Tensor Networks for Dimensionality Reduction and Large-Scale Optimization Problems: Perspectives and Challenges PART 1 1".
- [24] N. Lee, I. V Oseledets, A. Phan, Q. Zhao, and D. Mandic, "Low-Rank Tensor Networks for Dimensionality Reduction and Large-Scale Optimization Problems: Perspectives and Challenges PART 1".
- [25] W. Hsiao, "Journal Club: Brief Introduction to Tensor Network."
- [26] I. V Oseledets, "Copyright © by SIAM. Unauthorized reproduction of this article is prohibited. Copyright © by SIAM. Unauthorized reproduction of this article is prohibited", vol. 33, no. 5, pp., 2295–2317, 2011.
- [27] L. Bu, R. Babu, and B. De Schutter, "Multi-agent reinforcement learning : An overview * Multi-Agent Reinforcement Learning : An," vol. 19, 2010.
- [28] H. Ren, "Reinforcement Learning in Cooperative Multi-Agent Systems," pp. 1–8.
- [29] M. Yoon and J. Bekker, "New reinforcement learning algorithm for robot soccer," no. February 2016, 2017.
- [30] F. Farahnakian, "Reinforcement Learning for Soccer Multi-agents System," 2009.
- [31] A. Cichocki and N. Lee, "Tensor Networks for Dimensionality Reduction and Large-Scale Optimization Part 1 Low-Rank Tensor Decompositions," vol. 9, no. 4, pp. 249–429, 2017.
- [32] L. G. May, "Tensor Factorization via Matrix Factorization," vol. 38, 2015.
- [33] M. Pouyan, A. Mousavi, S. Golzari, and A. Hatam, "Improving the Performance of Q-learning Using Simultaneous Q-values Updating," no. Ictck, pp. 26–27, 2014.