T.C.

ALTINBAŞ UNIVERSITY

Electrical and Computer Engineering

CENTRALIZED REINFORCEMENT LEARNING FOR THE INTERNET OF THINGS DEVICES

Ahmed Hefdhi Hussein HUSSEIN

Master Thesis

Supervisor

Dr. Osman N. UÇAN

Istanbul (2019)

CENTRALIZED REINFORCEMENT LEARNING FOR THE INTERNET OF THINGS DEVICES

by

Ahmed Hefdhi Hussein HUSSEIN

Electrical and Computer Engineering

Submitted to the Graduate School of Science and Engineering

in partial fulfillment of the requirements for the degree of

Master of Science

ALTINBAŞ UNIVERSITY

2019

This is to certify that I have read this thesis and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Prof. Dr. Osman N. UÇAN

Supervisor

Examining Committee Members (first name belongs to the chairperson of the jury and the Second name belongs to supervisor)

Prof. Dr. Osman N. UÇAN	Engineering and Natural Science, Altinbas University	
Prof. Dr. Oğuz Bayat	Engineering and Natural Science, Altinbas University	
Asst. Prof. Dr.Adil Deniz Duru	Physical Education and Sports, Marmara University	

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Dr. çağatay AYDIN

Head of Department

Prof. Dr.Oğuz BAYAT

Director

Approval Date of Graduate School of

Science and Engineering: 9 / 5 / 2019

DEDICATION

In the name of Allah the merciful.

I would first like to thank God Almighty who gave me the opportunity and helped me to finish my Master degree.

My dear father..

To my angel in life .. to the meaning of love and to the meaning of compassion and dedication .. to the smile of life and the secret of existence to whom your invitation was the secret of my success and affection

Dear mother..

To the one who has the largest and so I rely .. to a candle burning illuminates the darkness of my life ..to those who have gained strength and love without limits ..to whom did you know the meaning of life

To my dear wife, the source of comfort, happiness, and affectionate heart is for you and for me , to the one who pleases my heart and blesses to the flower garden that sprouts flowers blossoms

To those who are closest to my soul to those who share with me the bosom of pain, and through them I draw my strength and my determination .. My brothers and sisters

Who paved the way in front of me to reach the height of science to .. my professor

"I was in my studies and shared my thoughts reminder and appreciation .. My friends

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Prof. Dr. Osman N. Uçan for all the knowledge and support he provided during my study for the Master Degree and throughout the work to complete this thesis,which have diluted the difficulties I have faced during the study. You have made my dream come true.

ABSTRACT

CENTRALIZED REINFORCEMENT LEARNING FOR THE INTERNET OF THINGS DEVICES

Ahmed Hefdhi Hussein HUSSEIN,

M.Sc., Electrical and Computer Engineering, Altınbaş University

Supervisor: Dr. Osman N. UÇAN

Date: April/2019

Pages: 56

The wide availability of the internet and communications schemes that allow connecting different types of devices to that network has emerged the Internet of Things (IoT). IoT devices are being deployed in different environments to collect data and execute commands, so that, certain tasks can be achieved. However, the limited resources on these devices restrict the range of applications that utilize them. Moreover, the rapid development in machine learning and artificial intelligence increases the demand for these tasks from IoT devices. Reinforcement Learning (RL) is one of these techniques that has been widely investigated in recent years, according to its ability to recognize and interact with different environment and tasks. In this study, a new method that uses a centralized computer to train neural networks that are used for RL in IoT devices and update the operations of these devices. Two approaches are evaluated using the proposed method. The first approach uses a single neural network for all IoT devices. This neural network is trained using all the data incoming from the IoT devices. The second approach uses a separate neural network per each IoT device and train it using only the data incoming from that device. The representation of the states to the neural network is also conducted using two approaches. The state of the environment in the first approach is presented to the neural network as is, without any modification to the image information. In this approach, the neural network is required to learn the extraction of all the important information, such as the direction of the car, its speed and the position of the path. In the second approach, the information about the speed, steering and drifting are extracted from the image that represents the environment and provided as numerical values to the neural network. Moreover, only pixels from the vertical and two diagonal lines of sight are collected from the image and delivered to the neural network. The second representation, which is more efficient, has shown significant improvement to the learning rate of the neural network, as the data is more concentrated and less noisy, in addition to the significant reduction in the size of the data being transmitted from the IoT device to the central computer.

Keywords: Artificial Neural Networks; Reinforcement Learning; Internet of Things; Deep Q Learning.

TABLE OF CONTENTS

LIST OF TABLES

<u>Pages</u>

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

USB	: Universal Serial Bus
WSN	: Wireless Sensor Network
IoT	: Internet of Things
ML	: Machine Learning
RL	: Reinforcement Learning
ANN	: Artificial Neural Network
RFID	: Radio Frequency Identification
MSDS	: Material Safety Data Sheets
REST	: Representational State Transfer
HTTP	: Hyper-Text Transfer Protocol
FF-NN	: Feed-Forward Neural Network
CNN	: Convolutional Neural Network
DQN	: Deep Q-Learning
XML	: eXtensible Markup Language
JSON	: JavaScript Object Notation

1. INTRODUCTION

The peripherals connected to a computer, which enable it from interacting with the environment and external devices, may be divided into two categories. The first category is the sensors, which have the ability to measure one or more variables and convert these values into computer standards. The second category receives commands from the computer and executes these commands on other devices. The use of these peripherals led to the revolution of using computers in automation [1], where computers became capable of interacting with other devices in order to acquire the necessary input data, process them to make a decision, and then execute that decision using the output peripherals. The benefits of using such systems enabled the evolution of these systems from a single-duty device, which is usually capable of interacting with only one kind of inputs or outputs, to the embedded systems that are capable of interacting with multiple inputs and outputs mountainously.

These devices are usually of limited resources. Thus, they are usually connected to a computer in order to make use of the relatively huge capabilities of the computer. To make these devices more useful in different applications, they are designed to be able to use the existing computer ports in order to exchange information with the computer. Earlier devices used the computer's parallel port to send and receive data [2]. Later, the serial port is also used to exchange data between these devices and computers [3]. The use of serial port gives more flexibility to the distance between a device and the computer. Then, with the help of microcontrollers or some specialized converters, these devices are connected to computers using Universal Serial Bus (USB). This enables high-speed data transfer between a device and the computer [4].

The rapid growth of Wireless Sensor Networks (WSN) usage in different fields of application and with the need to communicate to these sensors from different places leads to the urgent need of connecting these sensors to the internet, which created the internet of things (IoT), [5]. These applications may vary from saving lives by monitoring patients' vitals [6] to monitoring the environment for the plants in agricultural applications [7]. These devices are connected to the internet using a different interface such as wireless internet (WIFI module) or using mobile networks (GSM module). Connecting these devices to the internet enables communications among them, to a centralized device or to monitoring and control devices.

Techniques that attempt to allow computers to interact with their environments without a predefined set of rules that define such interaction are investigated in Machine Learning (ML) field of study [8]. ML aims to allow such interaction by learning from examples collected from the environment or by directly interacting with that environment to find the best possible solutions [9, 10]. The use of examples collected from the environment can be divided into two main categories, the supervised and unsupervised techniques [11, 12]. When additional information is added to the examples, to represent the experience from a human expert in that domain, the knowledge extraction is focused on the relations between the characteristics of the inputs and the information added to them [13, 14]. Hence, such techniques belong to the supervised machine learning category. Moreover, when the data are provided without any additional information, the learning is denoted as unsupervised and focuses on the relations among the inputs themselves, depending on their characteristics [15, 16].

Recently, more emphasis is applied to the learning techniques that have the ability to interact directly with the environment, in order to approximate the definition of the environment. Such techniques are known as Reinforcement Learning (RL) techniques, where an agent is set to interact with the environment by executing different actions [17, 18]. However, as the impact of these actions on the environment are unknown to the agent, a reward is sent back to the agent, so that, the agent can recognize the quality of the executed action. Based on these rewards, the agent becomes capable of recognizing the optimal action per each state of the environment, which is described by the inputs collected by the agent [19, 20].

According to their capabilities in approximating computations that define the environment, Artificial Neural Networks (ANNs) are widely used in RL [21, 22]. The aim of the ANN in such application is to predict the reward that the agent can collect from the environment per each possible action it can execute in the current state. Based on these rewards, the agent can simply select the action that maximizes the overall reward, i.e. select the optimal action for the current state [23, 24]. However, when the agent first starts to interact with the environment, the neural network has no approximation to the definition of the environment. Hence, the agent is required to execute random actions per each state to train the neural network to predict the rewards values for the actions the agent can take per an environment state [25, 26].

Neural networks rely on intense matrices computations for both the prediction and training phases. However, the computations required for the training phase is significantly more complex than those required to provide a prediction [27]. Moreover, the existence of more data can significantly improve the performance of the ANN. In RL, providing more accurate reward values can significantly assist the agent for better action selection at a certain state. The reward values predicted by the neural network for the actions available for a state the agent has been through before are more accurate than those predicted for a state that the agent has never been through [28].

1.1 PROBLEM STATEMENT

The employment of RL techniques has been increasing rapidly in recent years, according to its ability to learn directly by interacting with the environment. RL relies mainly on ANNs, which require intense matrices operations, especially during the training phase. Such computations require resources that are beyond the capabilities of IoT devices. Moreover, even if the resources of the IoT device can be employed for such task, training the ANN using these resources reduces the efficiency of the IoT device, as such limited resources can be employed to achieve the actual tasks required from the device.

Moreover, the quality of the predictions provided by the neural network improves by increasing the training data. From RL's point of view, providing the actual rewards collected from the environment, instead of the approximations from the neural network, improves the predictions for each action in the state. Hence, the approximation of the rewards values for actions in other states that the agent has not been through yet. Thus, the performance of the IoT devices can be improved in two main aspects, the efficiency of the resources consumption and the selection of the optimal action per each state by using a centralized computer that collects the data from these IoT devices, train a neural network and update the IoT devices with the new values of the neural network.

1.2 THE AIM OF THE STUDY

Centralizing the training phase of the neural network used to achieve the same task in different environments can improve the performance of the IoT device. Thus, in this study, a framework that enables IoT devices of making use of the relatively larger resources on a central computer to train the neural network is proposed. The proposed framework defines the structure of the data being sent from the IoT devices to the central computer and vice versa. The parameters of the neural network are updated during the training and the new values are forwarded to the IoT devices, so that, even if the connection to the central server is lost, the IoT devices can still perform their tasks.

Two approaches are proposed in this study to train the neural network. The first approach uses a single neural network that is trained using the data received from all the IoT devices, which is then used to update the parameters of all IoT devices. In this approach, all the IoT devices share the same neural network parameters. In the second approach, a neural network is implemented per each IoT device and trained using only the data received from that IoT devices. In this scenario, the parameters of the neural network in an IoT device is different from those in another device. The first approach allows the neural network to gain the ability of interacting with different environments, as the data is collected from different environments to achieve the same task. The neural networks produced in the second approach are more specialized in the environment that its data are used to train the neural network. The performance of the approach that produces better overall performance is selected for the proposed method, so that, the performance of the IoT devices is maximized using minimum resources consumption.

1.3 LAYOUT OF THE THESIS

The remaining chapters of this thesis are organized as follows:

- Chapter Two reviews the literature related to RL and the methods proposed to implement its techniques in IoT devices.
- Chapter Three describes the proposed framework and the approaches proposed to train the neural networks using the data received from the interaction between each IoT device and its environment.
- Chapter Four presents the experiments conducted to evaluate the performance of the IoT devices using both approaches.
- Chapter Five discusses the results collected from the experiments and compares them to the state-of-the-art method in the literature for the same purpose.

- Chapter Six illustrates the conclusions of this thesis and the future work that is going to be conducted to employ the proposed method in different RL tasks that use IoT devices.

2. LITERATURE REVIEW

The high availability of internet connection has encouraged the use of these connections to communicate information among different devices, other than computers and smart devices usually used by internet users. These devices are used in different applications to provide different kinds of services to the users, where in most cases, the information being exchanged are automatically collected by these devices and require to user's interaction. This phenomenon has created the Internet of Things (IoT) and it has become mandatory to adopt these devices and handle their communications [29]. The use of IoT devices has grown rapidly in recent years, according to the features they provide, such as mobility and accuracy. Thus, the IoT devices have been widely used in different fields of application, such as healthcare, manufacturing, electricity, security and vehicles. The use of IoT devices in different applications is illustrated in Figure 2.1, based on the percentage of devices used in each field [30].

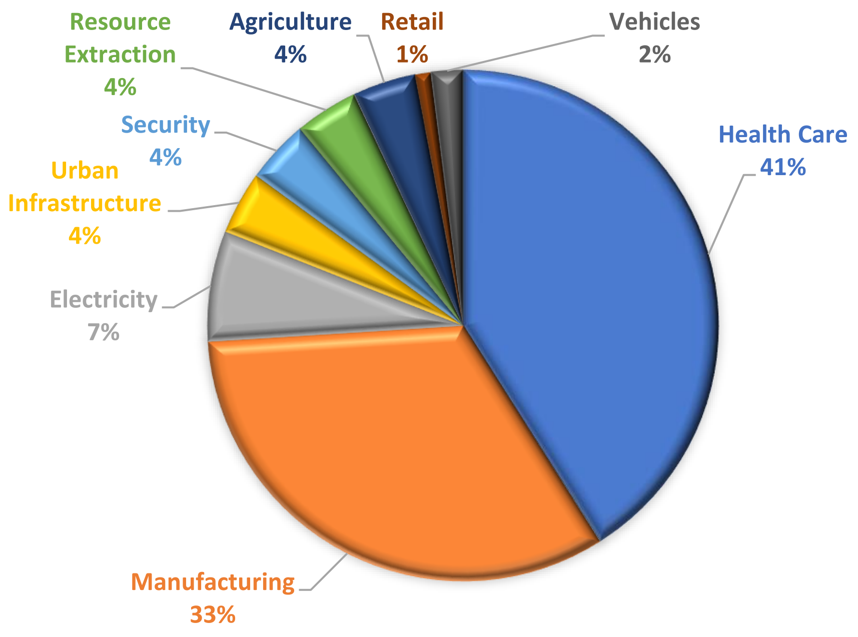


Figure 2.1: The use of IoT devices in different fields of applications [30].

2.1 EMPLOYMENT OF IOT DEVICES

A healthcare system based on IoT is proposed by Arijit et al. [31] that collects several biometric and environment measures to detect any anomaly in these measures in order to alarm the patient to seek for a medical care. The anomaly detection is based on data mining techniques, where measured variables are sent to a remote server to detect these anomalies. The patient’s location, activity, movement and heart activity are monitored using the Global Positioning System (GPS), Accelerometer, Magnetometer, and Photo-Plethysmography (PPG) sensors, respectively. Moreover, an emergency health transmission system is proposed by Govindhan et al. [32] the relies on the IoT to monitor the parameters of a patient’s body in order to assist providing better health care for that patient in case of emergency, where the recent vital measures can be analyzed to predict the required care in need. The existence of such systems imposes the need for rigid long-life monitoring using IoT devices, so that, reliable services are provided.

Although the IoT is not limited to healthcare, this field has the highest share among all other fields as shown in Figure 2.1. However, there are different other services that rely on IoT devices to improve their performance. A system is proposed by Kim et al. [33] manages the security of a chemical laboratory using IoT sensors. This system uses flame, gas and Radio-frequency identification (RFID) sensors to evaluate the environment in the laboratory using Material Safety Data Sheets (MSDS) to detect any hazards, so that, the administrator of the laboratory is immediately informed. The system uses an existing Application Programming Interface (API) to achieve communications among the different parts of the system, using Representational State Transfer (REST) architecture based on Hyper-Text Transfer Protocol (HTTP).

Recently, the employment of IoT devices has enabled significant improvements in the self-driving, i.e. autonomous, vehicles implementation, which is gaining significant attention according to the benefits of such applications in reducing the risks of accidents and the comfort it provides to the drivers [34-36]. These applications rely mainly on RL techniques, according to the enormous number of states and actions the autonomous driver is required to handle [37, 38]. In vehicles with larger power sources, such as cars, more resources can be available for the computing device that is responsible for predicting the optimal action based on the state of the vehicle. However, the quality of the predictions can still be improved by sharing experiences, so that, the driving unit in a certain vehicle can evaluate the actions for a certain state that it has never been through, such as accidents [39].

2.2 REINFORCEMENT LEARNING

Reinforcement learning uses the concepts of agents, environments, states, actions and rewards [40-42]. As shown in Figure 2.2, the environment receives the actions selected by the agent and outputs the new state of the agent and the reward. Agents, on the other hand, collect the new state and the reward in order to select the next action, which is return produces new state and reward from the environment. However, the agent does not have a clue about the way the environment returns the next state and the rewards of a certain action. Thus, in reinforcement learning, the agent attempts to predict the action that maximizes the rewards received from the environment, by approximating the behavior of the environment and how it responds to the actions [18].

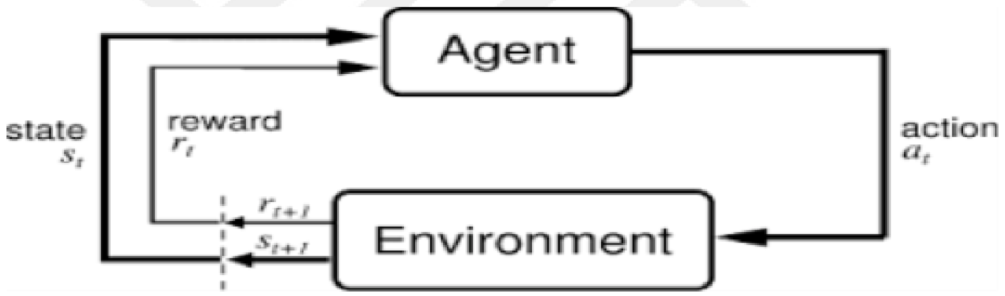


Figure 2.2: Illustration of the interaction between the Agent and the Environment in reinforcement learning.

The main components in RL applications are defined as follows:

- **Agent:** Is the component that is responsible of making the appropriate decision, depending on the state collected from the environment, to achieve the goal of the task assigned to it, such as making a delivery by a drone or navigating a car, safely, to the intended destination.
- **Action (A):** Defines the set of possible actions that an agent can take, so that, the agent can predict the reward it gets upon the execution of each action at a certain state. For an autonomous vehicle, the possible actions at any state are to accelerate, deaccelerate, go left, go right, go straight and do nothing. This set represents the simplest actions for the RL agent, where more actions can

produce better performance but increases the complexity of the decision-making procedure, according to the larger possibilities.

- **Discount Factor:** To allow the agent to focus on maximizing the overall reward rather than emphasizing on the instant one, the maximum reward from the new state the agent becomes into when an action is executed is included in the computation of the current rewards. However, the reward value of the next state is reduced by multiplying it by the discount factor, so that, the effect of the instant reward and the overall reward is balanced. For instance, if an autonomous vehicle is rewarded based on the instant values only, deacceleration at risky situations is not considered by the agent, as it cannot result in the maximum instant reward. Including the final rewards in the computations increases the reward expected from avoiding accidents, which allow the agent to make the appropriate decisions in that manner. Moreover, relying only on the final reward can encourage the agent to take some unwanted actions, such as driving off roads, to maximize the final reward. Thus, the discount factor must be selected to balance all the scenarios and produce the optimal performance from the agent.
- **Environment:** The domain that the agent is interacting with, by executing the actions and collecting the rewards. In autonomous driving, the environment represents the street the car is being driven through and the traffic in those streets.
- **State (S):** The description of the current situation of the agent in the environment, which can be represented to the agent in different formats. For instance, an autonomous driver requires knowledge about the path it is following, its current position on that path, the nearest vehicle and obstacles ahead.
- **Reward (R):** Represents the feedback from the environment for the action selected by the agent. Higher rewards values indicate more appropriate actions for the current state, while lower values indicate that the correspondent actions are less appropriate for the current state. For instances, deaccelerating the vehicle may reduce the reward under certain circumstances, such as clear path

and low speed, but such action can have higher rewards in states that describe an incoming vehicle, which can result in an accident.

- **Policy (π):** Is the approach employed by the agent to select the action appropriate for the current state to maximize the reward.
- **Value (V):** Under policy π, the long-term reward expected by the agent for the current state Vπ(s), considering the discount factor defined for the agent. This value allows the agent to avoid being in states that can dramatically reduce the long-term reward, even if it maximizes the instant reward. For instance, increasing the speed above the speed limit can increase the instant reward, as more distance is traveled faster, but considering the possibility of a fine or an accident allows the agent to make more reasonable decisions.
- **Q-Value (Q):** This value defines the overall reward for a certain action at a certain state, i.e. Q^{π}(s, a). The agents rely mainly on this value in making their decisions, so that, the action that returns the maximum overall reward.

Reinforcement is based on the Bellman equation, which is proposed by the American mathematician Richard Bellman. Using this equation, the reward per each action for a certain state can be calculated based on the instant reward and all the rewards collected until the end of the episode, which can be terminated as the agent reaches its goal or by performing a specified number of actions [43, 44]. This reward is calculated as shown in Equation 2.1.

$$Q^{\pi}(s_t, a_t) = E[R_{t+1} + \gamma R_{t+2} + \gamma R_{t+3} + \cdots |s_t, a_t] \qquad (2.1)$$

According to this equation, the highest Q value from a certain state, s_t, can be used to calculate the Q value for any action that ends up with the agent in that state, by simply multiplying it by the maximum Q value, as shown in Equation 2.2.

$$new\ Q(s, a) = Q(s, a) + \alpha[R(s, a) + \gamma max Q'(s', a') - Q(s, a)] \qquad (2.2)$$

where the learning rate α is used to damp the variation in the Q value for the selected action in the current state and γ is the discount factor that controls the balance between the instant and long-term rewards. The new Q value is then used to update the function that is used to represent the environment, so that, the actual reward from executing the action is produced instead of an approximation. This value also assists the computation of the reward values expected in previous states, as this value provides the actual reward received from the environment.

2.3 ARTIFICIAL NEURAL NETWORKS

According to the complexity of the computations required to predict the reward of each action available for the agent at a certain state, deep neural networks are used to make the predictions, based on the inputs collected from the environment at that state. The action selected based on the predictions of the neural network aims to maximize the reward received from the environment by executing the action with the highest. However, these rewards may not be instant and depend on the results of a series of actions. Thus, it is important to wait to the end of the interaction with the environment to evaluate these actions. Moreover, in order to adjust the exploration and exploitation of the selected actions, e a certain percentage of random actions in any time instance are allowed to be executed by the agent, especially before providing significant training for the neural network. This enables finding the optimal output, even after a certain solution is found [45].

Artificial neural networks consist of neurons distributed in layers, where the output of a neuron is weighted and connected to another neuron in a different layer, as input [46], as shown in Figure 2.3. The decisions made by these networks depend on the values of these weights, which are updated using backpropagation [47]. Backpropagation measures the difference between the output of the neural network, and the actual output required from it and update the weights among the neurons, based on the effect of each weight over the output. The effect of each weight is measured by calculating the rate of change of the output values, with respect to that weight. Thus, these computations require intensive processing and neural networks with more layers, known as deep neural networks, have significantly more weights, which increases the complexity of the computations. These computations are very exhausting for the IoT device, according to their limited resources, and require larger computers to achieve them [48]. However, these computations occur during the training phase of the neural network only, and no further updates are required during runtime, in most cases. The computations required to calculate the output of a neural network are relatively easier than those required to train it, and they can be handled by the IoT device itself, as the output of each neuron can be calculated by simply passing the weighted summation, of the outputs collected from the neurons connected to it, through an activation function [49, 50].

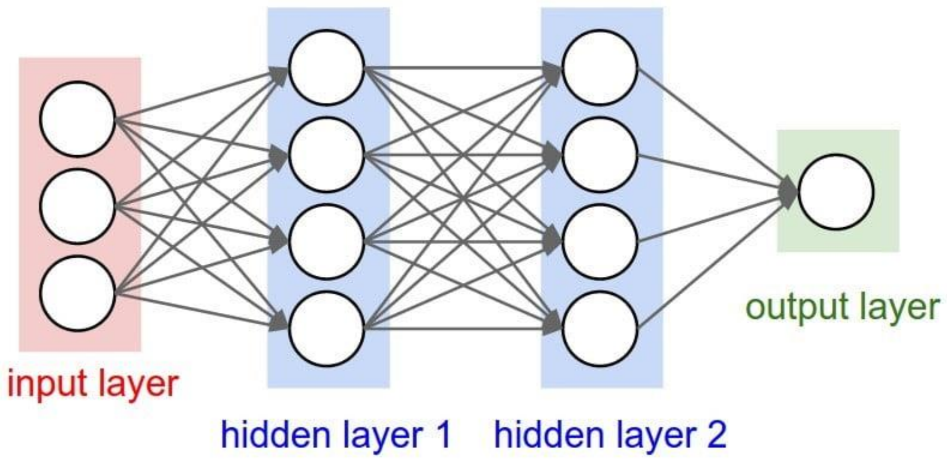


Figure 2.3: Hierarchy of a sample deep neural network.

Depending on the distribution of the inputs collected by each neuron in a layer, different types of artificial neural networks can be produced, for different tasks. The Feed-Forward Neural Network (FF-NN) shown in Figure 2.3 is the basic neural network that is used in different applications. However, when the neuron collects its inputs from two-dimensional windows, i.e. filters, that are convoluted through the two- or three-dimensional input, the neural network is known as Convolutional Neural Network (CNN). CNNs have shown significantly better performance than other types of neural networks when the inputs of the neural networks are images, which are normally represented in two- or three-dimensional arrays. Such better performance is the result of it CNN's ability of detecting and combining local features detected by the filters, regardless of their position in the input [51].

2.4 DEEP Q-LEARNING

The use of artificial neural network to approximate the function that defines the environment and predict the Q values per each action for a certain state, so that, the agent can select the most appropriate action is known as Q-Learning. The aim of this learning approach is to provide the neural network with the actual rewards collected from the environment, so that, it can predict these rewards in future operations [20]. However, as the neural network does not have any knowledge about the environment that the agent is interacting with, the training process relies on executing random actions at the beginning of the training [26]. As the neural network starts to gain more knowledge about the environment, the decisions of the agent can start to be less random and more dependent on the predictions of the neural network. To control such behavior, a value is defined to control the randomness in the decisions made by the agent. This value is denoted as the epsilon and it normally starts with a high value, i.e. more random actions, and reduced as the neural network gains more knowledge about the environment [52].

To select between the execution of a random action or based on the outputs of the neural network, the epsilon value is compared to a randomly generated value. If the random value is less than the epsilon, the action selected by the agent is the action that produces the highest reward, based on the predictions of the neural network. Otherwise, the action is selected randomly and executed against the environment [53]. In both cases, the reward collected from the environment upon the execution of the selected action at the current state is used with the maximum Q value predicted by the neural network for the new state the agent becomes in, to produce a new Q value that is used to train the neural network [54, 55].

When the agent finishes an episode, the neural network is trained using the data collected by the agent during the episode, i.e. the states, actions and rewards, and the epsilon value is reduced by a predefined ration, known as the gamma value. This process is repeated until the defined number of training episodes is reached, in which the neural network is expected to have gained enough knowledge to produce accurate Q value that can assist the agent to select the optimal action per each state it faces [43, 56]. The ability of the neural networks to provide approximations for states that it has never been through, during the training, allows the employment of these networks in the Deep Q-Learning (DQN) approach, so that, the agent still has approximate Q values to make the appropriate decision. Comparing this approach to the use of tables that contains the states and their corresponding Q values shows the benefits of the approximated computations, as Q values for states that are included in the Q table can be recognized by the agent [57, 58]. Thus, DQN has been widely used in approximating the functions of complex environments, such as those faced by autonomous vehicles drivers.

2.5 DQN IN AUTONOMOUS VEHICLES AND IOT

According to the complex nature of autonomous driving, imposed by the enormous number of rules and situations the drivers should consider in their decision about the appropriate actions, DQN has been widely used to assist providing these decisions by evaluating the outcome of each action at a certain state, i.e. the Q values. It is also important for the DQN method to consider all the variables in the environment, i.e. the traffic rules, street signs and vehicles that are sharing the road with the autonomous driver. Despite the increased complexity required to consider these factors, they can assist the agent is executing more efficient actions that can provide safer environment. However, the absence of signs in some intersections can impose a challenge for the agent, as the passing priority and the need to come to a complete stop before entering the intersection is unknown. Thus, a method based on DQN is proposed in [59] to allow autonomous drivers to go through such intersections relying only on the traffic conditions and the positioning of other vehicles in the intersection. This method illustrates the importance of the collaborative training, where certain states that an agent has never been through before are used to train the neural network, so that, better actions can be selected when the agent goes through a similar state.

OpenAI Gym [60] provide a variety of environments that can be used to train and evaluate the performance of RL techniques. The interaction between the RL technique and the environment is defined through a set of actions that the agent can execute at any state. When the action is executed, the environment returns the instant reward based on the selected action and the new state the agent becomes in. These environments are widely used in different RL studies, according to their ease of implementation and integration. One of the popular environments is the 'CarRacing' environment, where a predefined track is set for the car and the agent is required to pass through that path until the all the tiles on the path are collected. Per each tile, the agent is rewarded with three points, while a negative reward, i.e. a punishment, of -0.1 is returned when a frame passes without touching a time, i.e. the car is not moving or driving off roads [61]. A screenshot of this environment is shown in Figure 2.4.

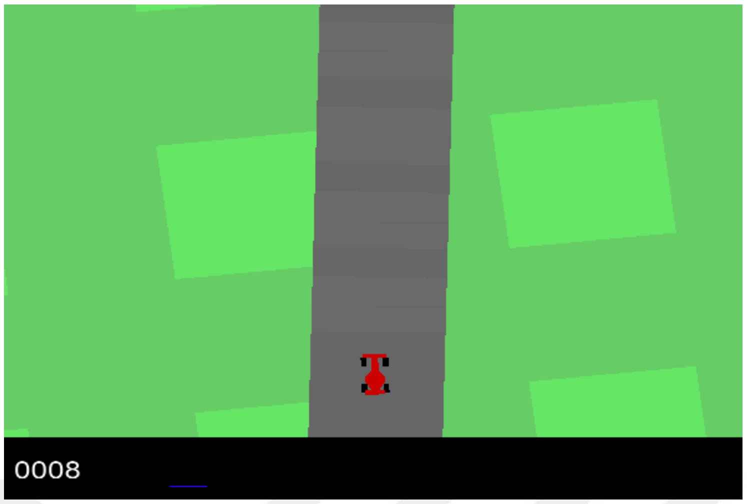
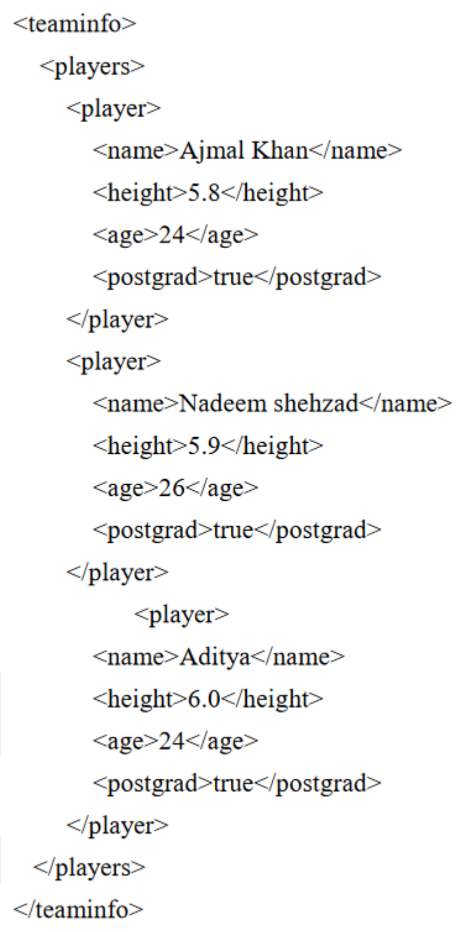


Figure 2.4: OpenAI gym's CarRacing environment.

DQN has also been widely employed to optimize the operation of IoT devices, such as optimizing the power consumption by IoT devices while transmitting data be avoiding jamming communication channels [62] or to enhance the security of these devices [63]. Despite the employment of DQN to improve the performance of the IoT devices, these methods do not use the resources of the IoT devices to execute the RL computations. Instead, information collected from the IoT devices, such as their position and the amount of data they transmit, are used to control their behavior.

2.6 STRUCTURED DATA EXCHANGE OVER THE INTERNET

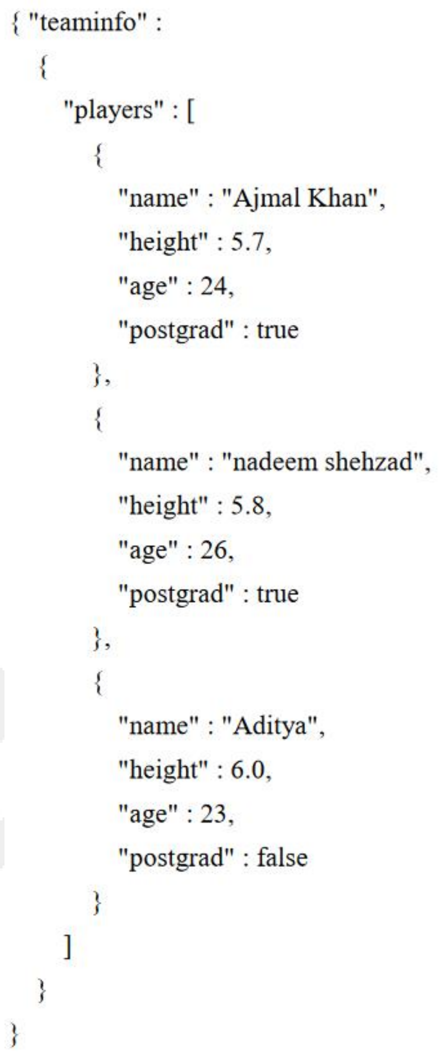
The data collected from a certain environment is structured, i.e. a fixed number of features are collected from the environment to represent the state of the environment to the agent. The communication of structured data over the internet can use different formats. Two of these formats that are widely used for structured data exchange is the eXtensible Markup Language (XML) and JavaScript Object Notation (JSON). The XML format is derived from the HTTP format, which is the most used format over the internet, as the websites are displayed in this format [64]. As shown in Figure 2.5, each object is defined using nested values described by an opening and closing tags [65].



```
<teaminfo>
   <players>
      <player>
         <name>Ajmal Khan</name>
         <height>5.8</height>
         <age>24</age>
         <postgrad>true</postgrad>
      </player>
      <player>
         <name>Nadeem shehzad</name>
         <height>5.9</height>
         <age>26</age>
         <postgrad>true</postgrad>
      </player>
            <player>
         <name>Aditya</name>
         <height>6.0</height>
         <age>24</age>
         <postgrad>true</postgrad>
      </player>
  </players>
</teaminfo>
```

Figure 2.5: A sample set of data in XML format [65].

The JSON format, shown in Figure 2.6, imposes less overload to the actual data being transmitted. According to the limited resources of the IoT devices and the dependency of the power consumption on the amount of data being transmitted, this difference between the XML and JSON is the key to select the JSON format to exchange data with the IoT devices [66].



```
{ "teaminfo" :
  {
    "players" : [
      {
        "name" : "Ajmal Khan",
        "height" : 5.7,
        "age" : 24,
        "postgrad" : true
      },
      {
        "name" : "nadeem shehzad",
        "height" : 5.8,
        "age" : 26,
        "postgrad" : true
      },
      {
        "name" : "Aditya",
        "height" : 6.0,
        "age" : 23,
        "postgrad" : false
      }
    ]
  }
}
```

Figure 2.6: A sample set of data in JSON format [65].

3. METHODOLOGY

The proposed framework defines the format that is used to exchange the data between the IoT devices, from one side, and a central computer that collects these data, train the neural network and return the updated values to the IoT devices, so that, these devices can achieve their tasks even when the communication with the computer is lost. According to its lower overhead, JSON format is selected for data exchange with the central computer. The computer then can use one of two approaches to train the neural networks of the IoT devices, one is to use a single neural network for all the IoT devices and the other is to use one neural network per each device.

3.1 DATA EXCHANGE

The data in the proposed framework flows in two directions. The first data flow is from the IoT device to the central computer. These data contain information about the state, the action executed and the reward collected by the agent from the environment after executing the action, as well as the new state that the agent has become in after the action is executed. The format shown in Figure 3.1 summarizes the JSON structure that is used to send the data from the IoT device to the central computer, where *n* is the number of features in the state.

```
{“IoT-ID”:
  {
     “Current”:[
        {“State”: $D_0$, $D_1$, ………, $D_n$,
         “Action”: A,
         “Reward”: R
        }
         ]
     “Next”:[
       {
        “State”: $D_0$, $D_1$, ………, $D_n$,
        }
         ]
   }
}
```

Figure 3.1: JSON structure for the data transmitted by the IoT devices.

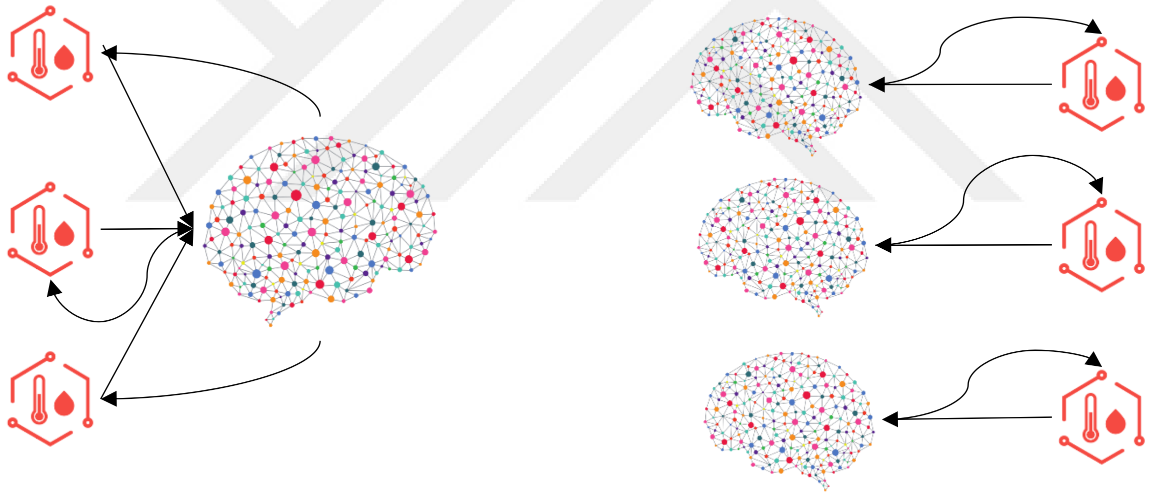
The other direction that the data flows in is from the central computer to the IoT devices. The data traveling in that direction contains the weights and biases values for each layer in the network. These values are used to update the neural network in the IoT devices, so that, the actions taken by the corresponding agent, i.e. the IoT device. This methodology ensures that the IoT device can still perform its task even if the communication with the central computer is interrupted. However, the neural network in that IoT device is not updated, as the computations required to update the neural network are exhaustive to its resources. The structure of the JSON format that is used to transfer the data from the central computer to the IoT devices is shown in Figure 3.1.

{“NN-Update”:
{
“Layer-1”:[
{“Weights”: W_{01}, W_{11}, ………, W_{i1},
“Biases”: B_{01}, B_{11}, ………, B_{i1},
}
]
“Layer-2”:[
{“Weights”: W_{02}, W_{12}, ………, W_{j2},
“Biases”: B_{02}, B_{12}, ………, B_{j2},
}
]
.
.
.
“Layer-k”:[
{“Weights”: W_{0k}, W_{1k}, ………, W_{lk},
“Biases”: B_{0k}, B_{1k}, ………, B_{lk},
}
]
}
}

Figure 3.2: JSON structure for the data transmitted by the central computer.

3.2 TRAINING THE NEURAL NETWORKS

According to the complexity of the computations required to train a neural network, which is used for the DQN implementation, and to avoid exhausting the limited resources on IoT device, the proposed method uses a centralized computer to collect the data from the IoT devices and train a neural network to achieve the required RL task. Accordingly, this computer receives data collected by multiple IoT devices, which can be employed in two different approaches to train the neural network. In the first approach, the data collected from multiple IoT devices that are intended for the same task are combined together to train the neural network, as shown in Figure 3.3 (a). In the second approach, a separate neural network is implemented per each IoT device, so that, the neural network of each device becomes more familiar with the corresponding device's environment, as shown in Figure 3.3 (b).



(a) Single Neural Network Approach

(b) Multiple Neural Networks

Figure 3.3: Neural network training approaches in the proposed method.

The IoT device is instructed to send a copy of the collected data to the computer every time such data are collected or be appending them and sending them periodically, depending on the size of the memory available on the IoT device and the network connection. Each data instance sent to the computer must contain the state of the environment before taking the action, the action, the state of the environment after the action is executed and the rewards collected from the environment. However, in applications that do not have instantaneous rewards, it is possible for the IoT device to only send the states and actions until the end of the sequential actions when the reward can be collected and sent to the computer. The computer then uses the collected data to train the neural network, depending on the approach being used, and send updates of the neural network to the IoT device periodically, so that, the IoT devices are still able to interact with the environment even if they are not connected to the computer.

4. EXPERIMENTAL RESULTS

Both approaches are implemented using Python programming language using a Windows computer with 3.1GHz CPU and 16GB memory with additional 11GB of memory in Nvidia GTX1080Ti GPU. According to the ability of the GPUs to parallelize complex matrices operations, they have been widely used to accelerate the training of neural networks, as they contain complex matrices operations. According to the recent use of RL in self-driving cars, the “CarRacing-v0” environment from Gym’s OpenAI environment is selected for the evaluation. The simulated approach utilizes 100 IoT devices each interacting with its own track for 100 episodes. Each episode is terminated by the environment when the agent completes 100 frames or when it becomes out the borders of the defined region, including green regions outside the track. The state of the agent is described to the neural network using two formats. In the first format, the entire image as retrieved from the environment, which requires a transmitting a total of 55299 bytes per each frame. The second format extracts the crucial information from the frame and sends it to the neural network in the central computer, instead of the entire frame, which has reduced the size of the communicated data to only 37 Bytes. The extracted information is:

- The pixels from the vertical line in front of the car in grayscale.
- The pixels from the diagonal line starting at the position of the car to the top right corner of the screen in grayscale.
- The pixels from the diagonal line starting at the position of the car to the top left corner of the screen.
- The overall speed of the car in addition to four speeds, one per each wheel.
- The steering level of the car, from -1 to 1, representing 100% left to 100% right, respectively.
- The drifting level of the car, also varying from -1 to 1 representing 100% left drifting to 100% right drifting.

All the previous information is extracted from the state received from the environment, which is presented in graphical form, as shown in Figure 4.1.

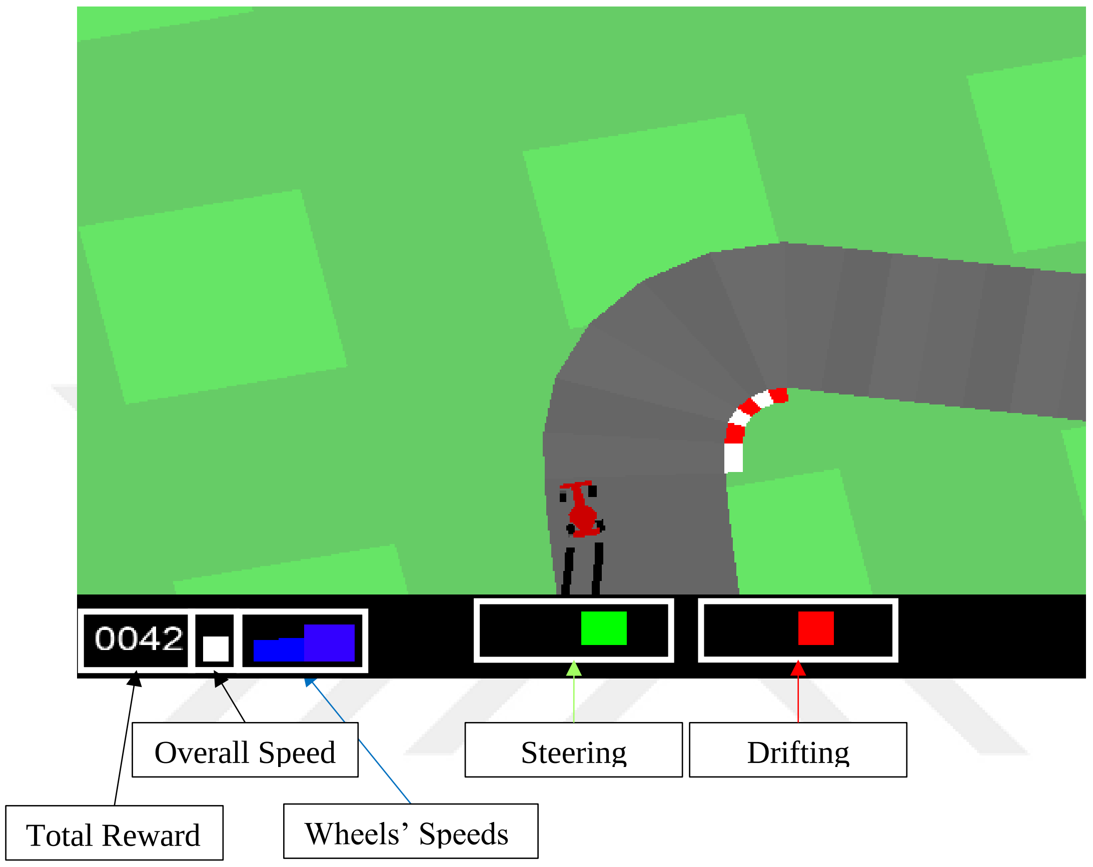


Figure 4.1: Information collected from the RacingCar environment.

However, according to the ability of CNNs in detecting local two-dimensional features, the performance of the proposed method is evaluated using the CNN directly with the image retrieved as the state of the agent from the environment. The action required by the environment consists of three values, the accelerator, brake and steering. However, naturally, driving a vehicle autonomously consists of nine possible actions, shown in Table 4.1. Thus, the output layer of all the neural networks consists of nine neurons, where each neuron predicts the Q value expected for the current state if the corresponding action is selected.

Table 4.1: Defined actions for the agent.

Action	**Description**
1	Accelerate.
2	Brake.
3	Steer Right (maintain speed)
4	Steer Left (maintain speed)
5	Accelerate and Steer Right
6	Accelerate and Steer Left
7	Brake and Steer Right
8	Brake and Steer Left
9	Do Nothing (maintain speed and direction)

The performance of each approach is evaluated by calculating the average score that each simulated IoT device achieves when interacting with its own environment. The score of an IoT device is calculated according to the rewards received from the environment, as illustrated in Section 2.5, using Equation 4.1, where f is the number of frames in the episode and T is the number of tiles that the car in the environment passes through.

$$Score = 3 \times T - 0.1 \times f \tag{4.1}$$

4.1 EXPERIMENT A – A SINGLE CNN FOR ALL IOT DEVICES

The performance of the first approach is evaluated in this experiment, where a single convolutional neural network is created for all the simulated IoT devices. Upon the arrival of each data instance form any of the IoT devices, the network is trained to update the predicted reward value for the executed action, based on the actual reward collected from the environment. The implemented CNN for this experiment consists of the layers described in Table 4.2. By the end of all episodes, the average score is 418 with an average deviation of ±37. Figure 4.2 shows the average reward for all the 100 IoT devices per each episode.

Table 4.2: Structure of the CNN implemented for the DQN.

Layer	**Type**	**Size**
Input Layer	-	(96, 96, 3)
Conv1	Convolutional	128 × (3, 3)
Conv2	Convolutional	64 × (3, 3)
Dense1	Fully-Connected	128
Dense2	Fully-Connected	9

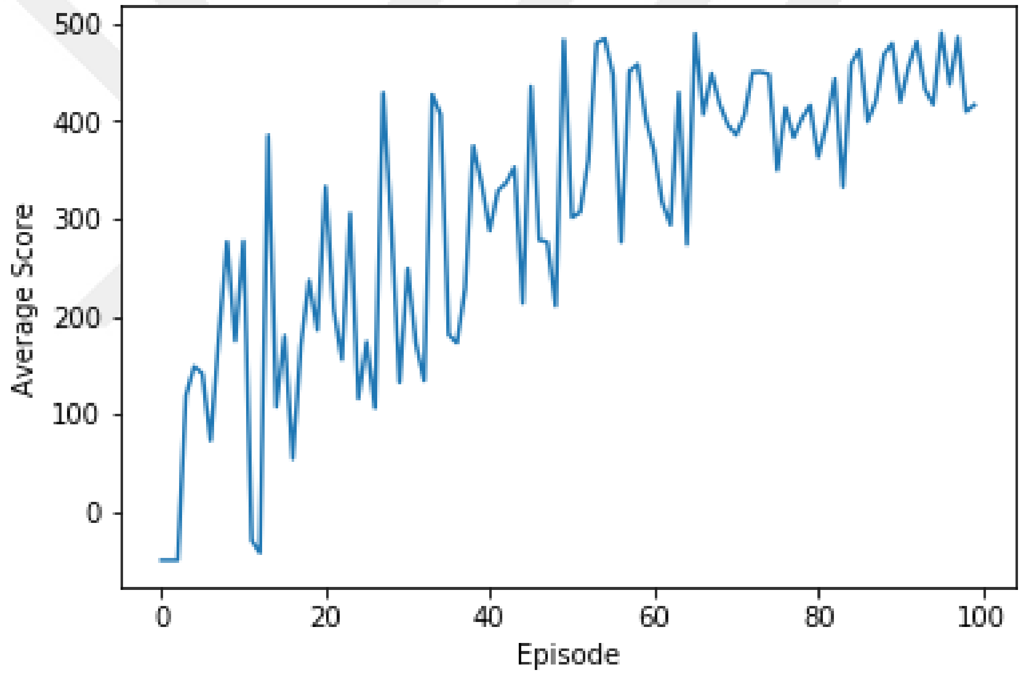


Figure 4.2: Average score versus episodes for the single CNN approach.

The results show that the CNN has not been able to gain enough knowledge to assist the decision-making of the agent. According to the complexity of the features in the convolutional layer, more training is required to allow more accurate predictions, so that, better decisions are made by the agent. Moreover, the amount of data transferred from the IoT device to the central computer, in this experiment, is equal to 54KB for the previous and current states, in addition to three bytes that represents the reward collected from the previous state and the action executed to get that reward.

4.2 EXPERIMENT B – A SINGLE FF-NN FOR ALL IOT DEVICES

The performance of the first approach is evaluated in this experiment, where a single feed-forward neural network is created for all the simulated IoT devices. Upon the arrival of each data instance form any of the IoT devices, the network is trained to update the predicted reward value for the executed action, based on the actual reward collected from the environment. Table 4.3 summarizes the FF-NN implemented for this experiment. By the end of all episodes, the average score is 937 with an average deviation of ±19. Figure 4.3 shows the average reward for all the 100 IoT devices per each episode.

Table 4.3: Structure of the FF-NN implemented for the DQN.

Layer	**Type**	**Size**
Input Layer	-	(37)
Dense1	Fully-Connected	128
Dense2	Fully-Connected	64
Dense3	Fully-Connected	9

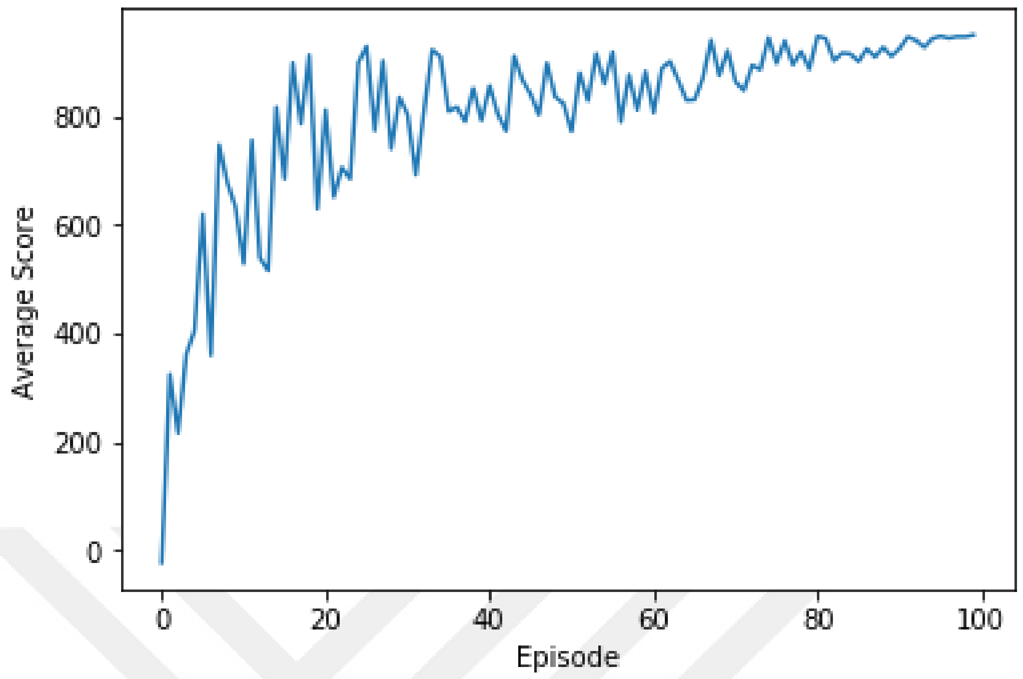


Figure 4.3: Average score versus episodes for the single neural network approach.

The results show that the implemented FF-NN has better performance for the DQN task, according to the simpler structure, hence, a lower number of parameters, i.e. weights and biases, to update. Moreover, as the state of the environment is provided to the neural network in a more efficient way, by providing the speed, steering and drift in numerical values, it has been able to show faster learning, compared to the CNN. Additionally, the size of the communicated data is significantly lower than that in CNN, with only 77 bytes to represent the previous and current state, as well as the reward and the selected action.

4.3 EXPERIMENT C - A SINGLE CNN FOR EACH IOT DEVICE

The second approach where a neural network is implemented per each IoT device is evaluated in this experiment, using a sperate convolutional neural network per each IoT device. Accordingly, a total of 100 convolutional neural networks, shown in Table 4.2, are created and each network is trained using the data received from the corresponding IoT device. By the end of all episodes, the simulated IoT devices have achieved an average score of 351 with an average deviation of ±29. The average score for all the 100 IoT device per each episode for this experiment is shown in Figure 4.2.

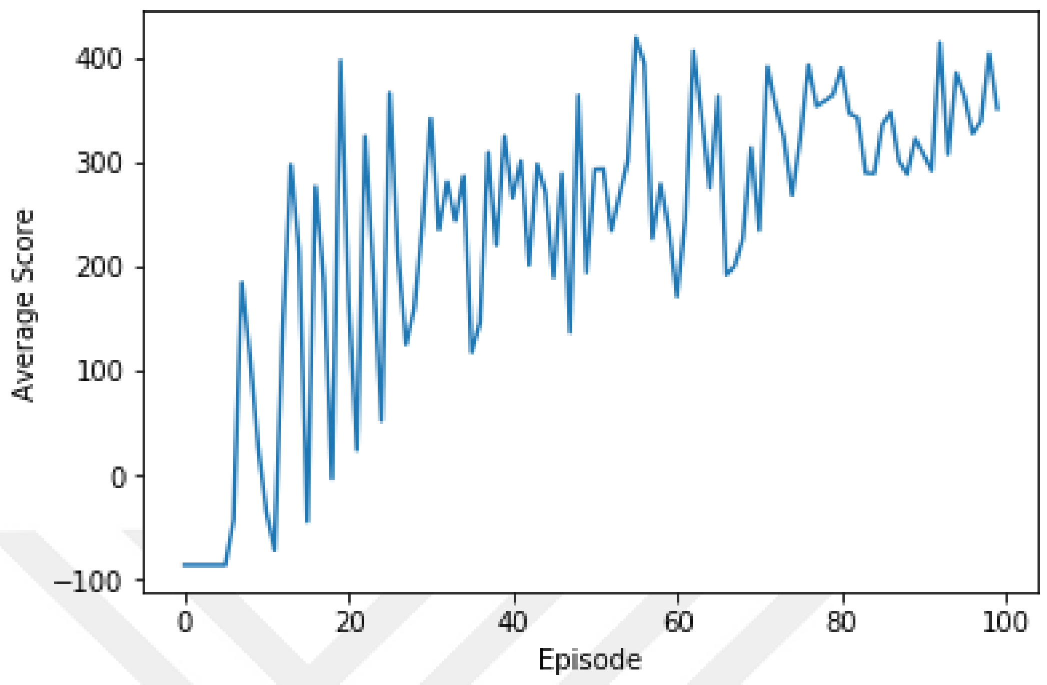


Figure 4.4: Average score versus episodes for the multiple neural network approach.

The figure shows huge oscillations, which are the results of new states that an IoT device may pass through that has no prior knowledge of how to handle them. In the first approach, when a single IoT device goes through such a state, the reward from the action executed by that IoT device is propagated through all the other devices, as the same neural network is used in these devices and this neural network has gained the ability to predict the reward from that action in that state. The size of the communicated data is similar to that is Experiment A.

4.4 EXPERIMENT D - A SINGLE FF-NN FOR EACH IOT DEVICE

The second approach where a neural network is implemented per each IoT device is evaluated in this experiment, using a sperate feed-forward neural network per each IoT device. Accordingly, a total of 100 neural networks are created and each network is trained using the data received from the corresponding IoT device. By the end of all episodes, the simulated IoT devices have achieved an average score of 890 with an average deviation of ±24. The average score for all the 100 IoT device per each episode for this experiment is shown in Figure 4.2.

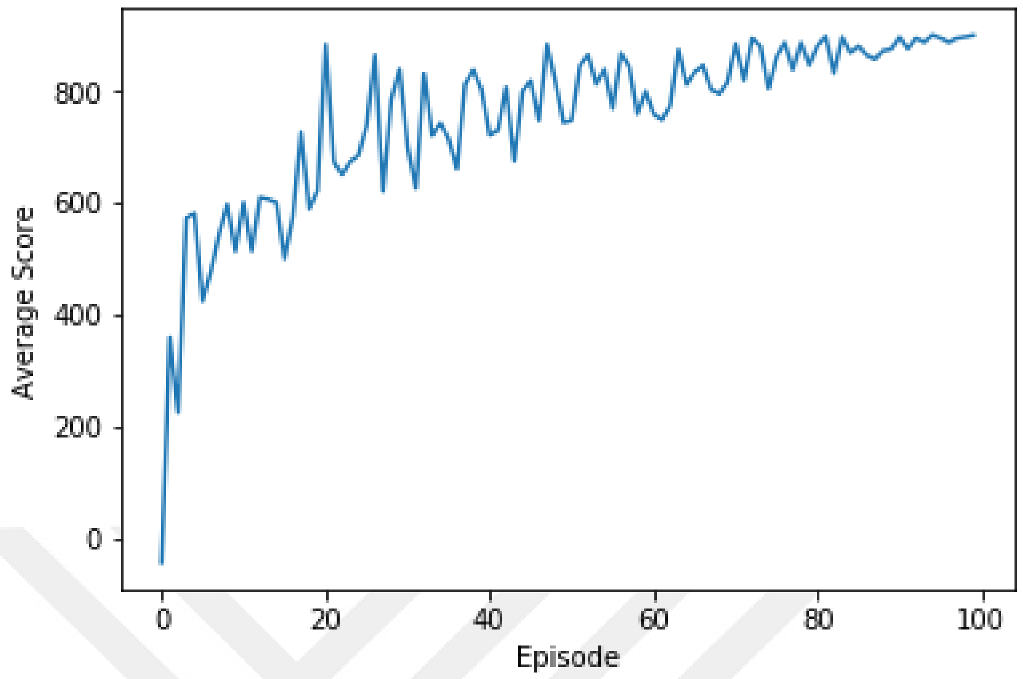


Figure 4.5: Average score versus episodes for the multiple neural network approach.

The figure shows that the more efficient representation of the states has produced less oscillation in the collected data, as the neural network has gained better approximation capability, in comparison with the results of Experiment C. The size of the communicated data in this experiment is similar to that in Experiment B.

5. DISCUSSION

In addition to the higher score achieved by the simulated IoT device in the first approach, where a single neural network is implemented for all IoT device, the lower deviation in the scores of these devices, compared to the second approach, show that the first approach has produced better performance. Moreover, the summary of the results, shown in Table 5.1, shows that the more efficient presentation of the state has been able to accelerate the learning of the neural network, for more accurate predictions of the Q value per each action in a certain state. Additionally, improving the efficiency of the states' representations has been able to significantly reduce the amount of data transmitted from each IoT device to the central computer, in addition to the improvement in the learning rate of the neural networks, as shown in Figure 5.1. This summary shows that the use of a single feed-forward neural network has achieved the best performance by scoring the highest average score. Moreover, the lower size of the data improves the performance of the IoT devices, as transmitting more data requires more of the limited resources of the IoT device.

Table 5.1: Performances summary from the conducted experiments.

Method	**Average Score**	**Exchanged Data Size (Bytes)**
Single CNN	418 ±37	55299
Single FF-NN	937 ±19	37
Multiple CNNs	351 ±29	55299
Multiple FF-NNs	890 ±24	37

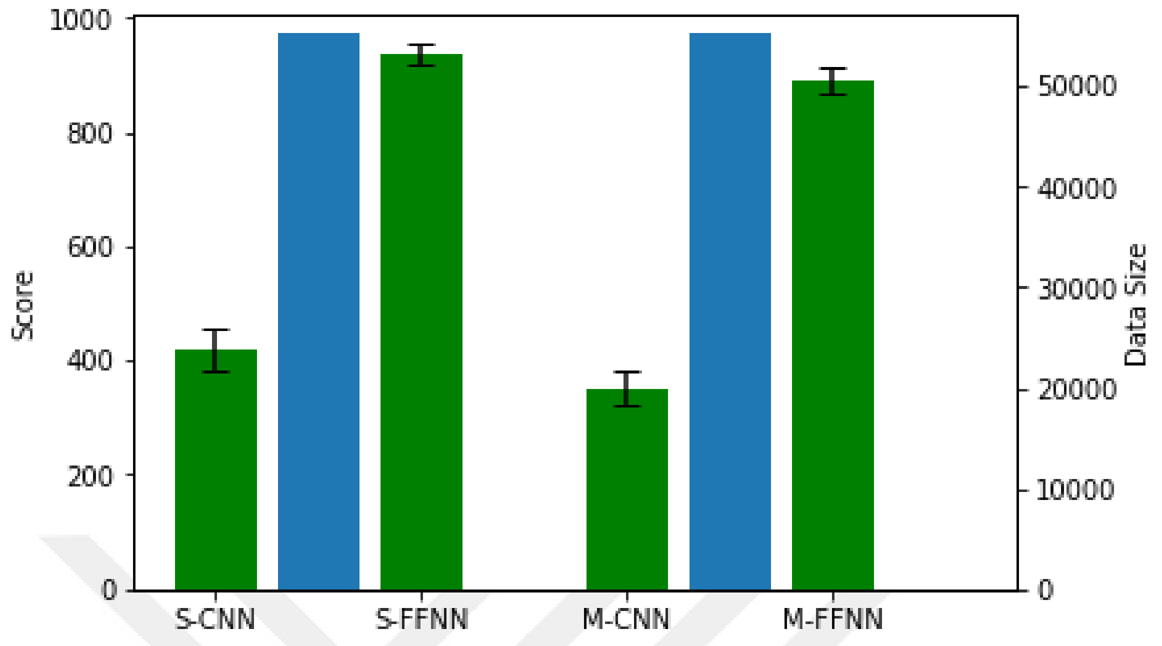


Figure 5.1: Average scores and data sizes of the conducted experiments.

Moreover, the comparison with the methods evaluated in [24], which is summarized in Table 4.1 and Figure 4.3, shows that the cooperation among the IoT device has been able to improve the performance of all these devices.

Table 5.2: Performance comparison with earlier methods.

Study	**Method**	**Average Score**
Ha and Schmidhuber [24]	DQN	343 ± 19
	A3C (continuous)	591 ± 45
	A3C (discrete)	652 ± 10
	DQN (memory replay)	838 ± 11
	Dual NN	906 ± 21
This Study	Single CNN	418 ±37
	Single FF-NN	937 ±19
	Multiple-CNNs	351 ±29
	Multiple FF-NNs	890 ±24

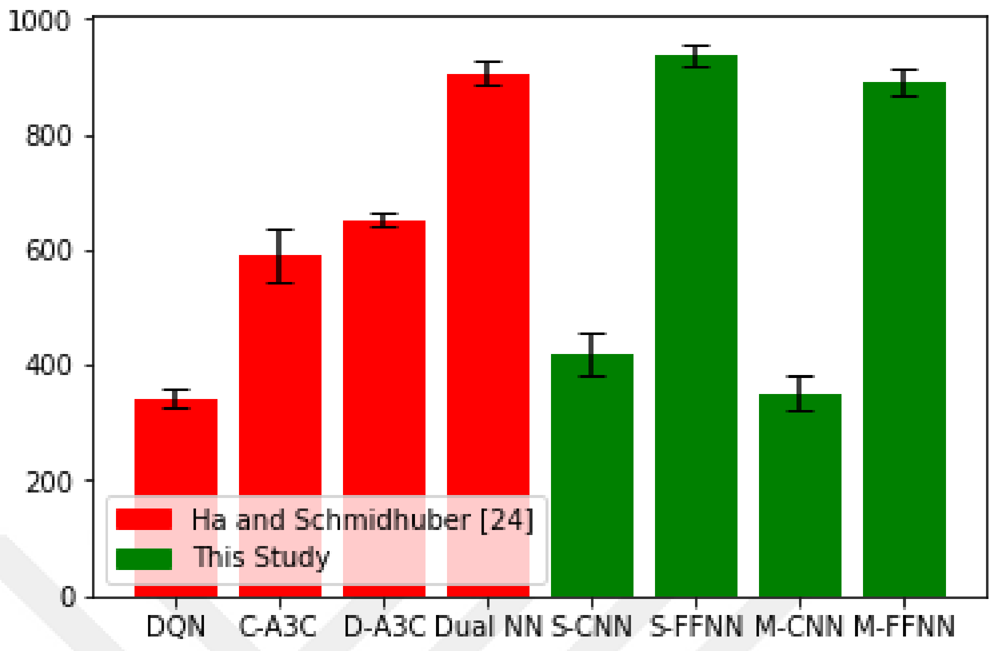


Figure 5.2: Illustration of performance comparison with earlier methods.

6. CONCLUSION

IoT devices are widely employed in different applications, according to their high mobility and low-cost. Accordingly, lower resources are available on these devices, which can limit the applications they are employed in. To allow a wider range of applications to utilize these devices, computers can be used to support their operations. Reinforcement learning is one of the fields that is attracting more attention in recent years, according to its ability of interacting with different environments. RL mainly utilizes artificial neural networks, which require intensive computations during training. Such training is difficult to be executed using the limited resources of IoT devices.

In this study, a new method is proposed to allow computers to train neural networks and update the operation of the IoT devices. Data collected by these devices are delivered to the central computer to train the neural network and update the weights of the similar network used to interact with the IoT's environment. The proposed method uses two different approaches for IoT devices that are intended to achieve the task. The first approach implements a single neural network that is trained using all the data collected from the IoT devices, be appending all the received data from these devices. The second approach implements a neural network per each IoT device, so that, this neural network becomes more familiar with the environment of the corresponding IoT device. The results of the experiments conducted in this study show that the use of a single neural network has improved the performance of all IoT device, as these devices are considered to be cooperating to update the neural network. However, the use of a sperate neural network per each IoT device has been slightly lower in performance than the first approach.

Moreover, the state of the environment that is delivered to the agent is also represented to the neural network in two formats. In the first format, the data received by the agent are forwarded to the neural network as they are, without any modifications or extractions. In the second approach, a more efficient way is used to represent the data to the neural network, where only three lines of sight are used to collect the data from the entire image and the speed, steering and drifting values are extracted from the image and delivered in numerical format to the neural network. The neural network is then trained to predict the Q value per each action for a certain state, in both formats. The results show that the use of the more efficient representation does not only significantly reduce the size of the transmitted data from the IoT device, it has also been able to improve the learning rate of the neural network. Such improvement in the learning rate is a result of the lower number of parameters, i.e. weights and biases, to be updated during training. Thus, more accurate results can be collected from the predictions of the neural network. Hence, the best performance of the evaluated approaches and data formats combines the use of a single feed-forward neural networks that predicts the reward values per each state per each state using the information extracted from the environment, instead of sending the entire image that represents the environment.

In future work, the benefits of using the centralized learning approach are going to be evaluated using other classification technique, such as random forest and Support Vector Machine. Despite the ability of training these classifiers locally in the IoT devices, the centralized approach can have a significant improvement in the learning speed, as more data is collected from different IoT devices and use them to accelerate the training and produce a better performance as more knowledge is available in the training.

References

[1] D. Popovic and V. P. Bhatkar, *Distributed Computer Control Systems in Industrial Automation* vol. 66: CRC Press, 1990.

[2] S. Rosminah and A. Z. M. Ali, "Development of hardware-interfacing learning kit for novice learning programming," *International Journal of Information and Education Technology,* vol. 6, p. 647, 2016.

[3] V. K. Patel and M. N. Patel, "Development of Smart Sensing Unit for Vibration Measurement by Embedding Accelerometer with the Arduino Microcontroller," *International Journal of Instrumentation Science,* vol. 6, pp. 1-7, 2017.

[4] E. Popa and V. Popa, "Graphic interface for numerical commands on the USB port of PC compatible computers," in *MATEC Web of Conferences*, 2017, p. 01008.

[5] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future generation computer systems,* vol. 29, pp. 1645-1660, 2013.

[6] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, *et al.*, "Health monitoring of civil infrastructures using wireless sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, 2007, pp. 254-263.

[7] F. TongKe, "Smart agriculture based on cloud computing and IOT," *Journal of Convergence Information Technology,* vol. 8, 2013.

[8] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science,* vol. 349, pp. 255-260, 2015.

[9] R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi, and C. Kim, "Machine learning in materials informatics: recent applications and prospects," *npj Computational Materials,* vol. 3, p. 54, 2017.

[10] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, 2000, pp. 1-15.

[11] P. Chaovalit and L. Zhou, "Movie review mining: A comparison between supervised and unsupervised classification approaches," in *Proceedings of the 38th annual Hawaii international conference on system sciences*, 2005, pp. 112c-112c.

[12] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials,* vol. 18, pp. 1153-1176, 2016.

[13] E. W. Ngai, L. Xiu, and D. C. Chau, "Application of data mining techniques in customer relationship management: A literature review and classification," *Expert systems with applications,* vol. 36, pp. 2592-2602, 2009.

[14] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering,* vol. 160, pp. 3-24, 2007.

[15] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping multidimensional data*, ed: Springer, 2006, pp. 25-71.

[16] Y. Kim, W. N. Street, and F. Menczer, "Feature selection in unsupervised learning via evolutionary search," in *KDD*, 2000, pp. 365-369.

[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*: MIT press, 2018.

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

[19] P. Dayan and B. W. Balleine, "Reward, motivation, and reinforcement learning," *Neuron,* vol. 36, pp. 285-298, 2002.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, p. 529, 2015.

[21] G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Transactions on Neural Networks*, vol. 16, pp. 57-67, 2005.

[22] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057-1063.

[23] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *nature*, vol. 529, p. 484, 2016.

[24] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Advances in Neural Information Processing Systems*, 2018, pp. 2455-2467.

[25] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, *et al.*, "Progressive neural networks," *arXiv preprint arXiv:1606.04671,* 2016.

[26] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Advances in neural information processing systems*, 2015, pp. 2863-2871.

[27] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Advances in neural information processing systems*, 2015, pp. 442-450.

[28] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 901-909.

[29] Z. Yang, Y. Yue, Y. Yang, Y. Peng, X. Wang, and W. Liu, "Study and application on the architecture and key technologies for IOT," in *Multimedia Technology (ICMT), 2011 International Conference on*, 2011, pp. 747-751.

[30] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials,* vol. 17, pp. 2347-2376, 2015.

[31] A. Ukil, S. Bandyoapdhyay, C. Puri, and A. Pal, "IoT healthcare analytics: The importance of anomaly detection," in *Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on*, 2016, pp. 994-997.

[32] P. Govindhan, G. V. Pratap, S. Balaji, M. Gurumoorthy, and H. Khudhrathulla, "Emergency Health Transmission System via Internet," *International Journal of Engineering Science,* vol. 16508, 2018.

[33] H. Kim, E. Lee, D. Kwon, and H. Ju, "Chemical laboratory safety management service using IoT sensors and open APIs," in *Information and Communications (ICIC), 2017 International Conference on*, 2017, pp. 262-263.

[34] T. Tettamanti, I. Varga, and Z. Szalay, "Impacts of autonomous cars from a traffic engineering perspective," *Periodica Polytechnica Transportation Engineering,* vol. 44, pp. 244-250, 2016.

[35] A. Lari, F. Douma, and I. Onyiah, "Self-driving vehicles and policy implications: current status of autonomous vehicle development and minnesota policy implications," *Minn. JL Sci. & Tech.,* vol. 16, p. 735, 2015.

[36] A. H. H. HUSSEIN, O. N. UÇAN, and O. BAYAT, "Centralized Reinforcement Learning for the Internet of Things Devices," *AURUM Journal of Engineering Systems and Architecture*, vol. Submitted, 2019.

[37] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[38] G. Hartman, Z. Shiller, and A. Azaria, "Deep Reinforcement Learning for Time Optimal Velocity Control using Prior Knowledge," *arXiv preprint arXiv:1811.11615*, 2018.

[39] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[40] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings 1994*, ed: Elsevier, 1994, pp. 157-163.

[41] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330-337.

[42] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279-292, 1992.

[43] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[44] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, *et al.*, "Deep q-learning from demonstrations," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[45] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, *et al.*, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[46] G. Ertaş, H. Ö. Gülçür, O. Osman, O. N. Uçan, M. Tunacı, and M. Dursun, "Breast MR segmentation and lesion detection with cellular neural networks and 3D template matching," *Computers in biology and medicine*, vol. 38, pp. 116-126, 2008.

[47] P. Gorgel, N. Kilic, B. Ucan, A. Kala, and O. N. Ucan, "A Backpropagation Neural Network Approach For Ottoman Character Recognition," *Intelligent Automation & Soft Computing*, vol. 15, pp. 451-462, 2009.

[48] O. Osman, A. M. Albora, and O. N. Ucan, "Forward modeling with forced neural networks for gravity anomaly profile," *Mathematical Geology*, vol. 39, p. 593, 2007.

[49] I. N. Da Silva, D. H. Spatti, R. A. Flauzino, L. H. B. Liboni, and S. F. dos Reis Alves, "Artificial neural networks," *Cham: Springer International Publishing*, 2017.

[50] O. Osman, A. M. Albora, and O. N. Ucan, "A new approach for residual gravity anomaly profile interpretations: Forced Neural Network (FNN)," *Annals of Geophysics*, vol. 49, 2006.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[52] Y. Chen and E. Kulla, "A Deep Q-Network with Experience Optimization (DQN-EO) for Atari’s Space Invaders," in *Workshops of the International Conference on Advanced Information Networking and Applications*, 2019, pp. 351-361.

[53] S. Yoon and K.-J. Kim, "Deep Q networks for visual fighting game AI," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 2017, pp. 306-308.

[54] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153-160.

[55] A. HUSSEIN, O. UÇAN, and O. BAYAT, "Centralized Reinforcement Learning for the Internet of Things Devices," *AURUM Journal of Engineering Systems and Architecture,* vol. Submitted, 2019.

[56] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, 2016, pp. 2137-2145.

[57] A. Pritzel, B. Uria, S. Srinivasan, A. P. Badia, O. Vinyals, D. Hassabis*, et al.*, "Neural episodic control," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 2827-2836.

[58] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances in neural information processing systems*, 2016, pp. 4026-4034.

[59] 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)*, 2018, pp. 2034-2039.

[60] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang*, et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540,* 2016.

[61] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Advances in Neural Information Processing Systems*, 2018, pp. 2450-2462.

[62] Y. Chen, Y. Li, D. Xu, and L. Xiao, "Dqn-based power control for iot transmission against jamming," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, 2018, pp. 1-5.

[63] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?," *IEEE Signal Processing Magazine*, vol. 35, pp. 41-49, 2018.

[64] J. Vihervaara and T. Alapaholuoma, "The impact of HTTP/2 on the service efficiency of e-commerce websites," in *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2018, pp. 1317-1321.

[65] Z. U. Haq, G. F. Khan, and T. Hussain, "A Comprehensive analysis of XML and JSON web technologies," *New Developments in Circuits, Systems, Signal Processing, Communications and Computers*, pp. 102-109, 2015.

[66] J. L. Rebelo Moreira, L. Ferreira Pires, and M. Van Sinderen, "Semantic Interoperability for the IoT: Analysis of JSON for Linked Data," *Enterprise Interoperability: Smart Services and Business Impact of Enterprise Interoperability*, pp. 163-169, 2018.