T.C.

ALTINBAS UNIVERSITY

Institute of Graduate Studies

Electrical and Computer Engineering

ADAPTIVE BEAMFORMING IN 5G NETWORKS USING DEEP REINFORCEMENT LEARNING

Salam Hazim Salman AL-SAMEERLI

Master Thesis

Supervisor

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Istanbul, 2021

ADAPTIVE BEAMFORMING IN 5G NETWORKS USING DEEP REINFORCEMENT LEARNING

by

Salam Hazim Salman AL-SAMEERLI

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

2021

The thesis titled “ADAPTIVE BEAMFORMING IN 5G NETWORKS USING DEEP REINFORCEMENT LEARNING” prepared and presented by “Salam Hazim Salman AL-SAMEERLI” was accepted as a Master of Science Thesis in Electrical and Computer Engineering.

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM

Supervisor

Thesis Defense Jury Members:

Asst. Prof. Dr. Abdullahi Abdu IBRAHIM	School of Engineering and Natural Sciences, Altinbas University	_____
Asst. Prof. Dr. Mesut ÇEVIK	School of Engineering and Natural Sciences, Altinbas University	_____
Asst. Prof. Dr. Sewale Musadaq Taha TAHA	Faculty of Communication, Beykent University	_____

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

Approval Date of Institute of Graduate Studies: ____/____/____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Salam Hazim Salman AL-SAMEERLI

DEDICATION

First and foremost, I would like to thank Allah Almighty for giving me the knowledge, ability and opportunity to undertake this research study and to persevere and complete it satisfactorily. Heartfelt thanks goes to my father and my mother. Every success is a direct consequence of their influence in my life and their love. At the end I have to mention my brother and sister for their support and love .

ACKNOWLEDGMENTS

I might want to thank my administrators: Asst. Prof. Dr. Abdullahi Abdu IBRAHIM Please let me express my profound feeling of appreciation and gratefulness to both of you for the information: direction and unrestricted help you have given me. I want you to enjoy all that life has to offer and further achievement and accomplishments throughout your life.

ABSTRACT

ADAPTIVE BEAMFORMING IN 5G NETWORKS USING DEEP REINFORCEMENT LEARNING

Salam Hazim Salman AL-SAMEERLI,

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

Supervisor: Dr. Abdullahi Abdu IBRAHIM

Date: 08/2021

Pages: 54

The need for additional bandwidth per each client that is being connected to the mobile cellular network and the rapidly growing number of devices being added to these networks, the need for additional capacity has increased rapidly in recent years. This demand has induced the proposal of a new generation of these networks, which is the fifth generation (5G). In this generation, the spectrum of the frequencies that are used to establish links between the clients and the base stations is widened to include millimeter waves (mmWaves). With the lack of ability of these waves to travel through obstacles and the use of Multiple Input Multiple Output (MIMO) technology to increase the capacity of the system, the enormous number of antennas in the Base Station (BS) is being used to create beams of these waves and direct them towards the best direction that establishes the communications with the designated client. Artificial Intelligence (AI) is being used to handle the high complexity of the decision-making task, which requires processing the input that represents the state of the client in the environment and select an antenna to establish the connection. To handle the temporary and permanent changes that may occur in the environment, the proposed method uses Reinforcement Learning (RL) for the decision making. Then, an antenna is selected based on the predicted bandwidths that each antenna may provide and the actual bandwidth is measured and used to train the neural network of the RL agent, so that, any changes are taken into consideration in future predictions. The results show that the propose method has been able to achieve higher bandwidth, compared to the existing Machine Learning (ML) based method,

which uses a regression training approach. The results also show that the use of the Gated Recurrent Unit has the highest performance, in terms of higher bandwidth, faster adaption and lower execution time.

Keywords: 5G Networks, Reinforcement Learning, Artificial Neural Networks, Deep Reinforcement Learning, Beamforming.

ÖZET

DERIN GÜÇLENDIRMELI ÖĞRENMEYI KULLANAN 5G AĞLARINDA UYARLANABILIR HÜZMELEME

AL-SAMEERLI, Salam Hazim Salman

Yüksek Lisans, Elektrik ve Bilgisayar Mühendisliği, Altınbaş Üniversitesi

Danışman: Dr. Abdullahi Abdu IBRAHIM

Date: 08/2021

Sayfalar: 54

Mobil hücresel ağa bağlanan her istemci için ek bant genişliği ihtiyacı ve bu ağlara hızla artan sayıda cihaz eklenmesi, ek kapasite ihtiyacı son yıllarda hızla artmıştır. Bu talep, beşinci nesil (5G) olan bu ağların yeni bir neslini önerdi. Bu nesilde, müşteriler ile baz istasyonları arasında bağlantı kurmak için kullanılan frekansların spektrumu, milimetre dalgalarını (mmWaves) içerecek şekilde genişletilmiştir. Bu dalgaların engellerden geçme yeteneğinin olmaması ve sistemin kapasitesini artırmak için Çoklu Giriş Çoklu Çıkış (MIMO) teknolojisinin kullanılmasıyla, Baz İstasyonundaki (BS) çok sayıda anten kirişler oluşturmak için kullanılıyor. bu dalgaları ortadan kaldırır ve onları belirlenen müşteri ile iletişimi kuran en iyi yöne yönlendirir. Müşterinin ortamdaki durumunu temsil eden girdinin işlenmesini ve bağlantıyı kurmak için bir anten seçilmesini gerektiren karar verme görevinin yüksek karmaşıklığını idare etmek için Yapay Zeka (AI) kullanılıyor. Ortamda meydana gelebilecek geçici ve kalıcı değişikliklerin üstesinden gelmek için önerilen yöntem, karar verme için Takviye Öğrenmeyi (RL) kullanır. Daha sonra, her bir antenin sağlayabileceği tahmin edilen bant genişliklerine dayalı olarak bir anten seçilir ve gerçek bant genişliği ölçülür ve RL aracısının sinir ağını eğitmek için kullanılır, böylece herhangi bir değişiklik gelecekteki tahminlerde dikkate alınır. Sonuçlar, önerme yönteminin, bir regresyon eğitimi yaklaşımı kullanan mevcut Makine Öğrenimi (ML) tabanlı yönteme kıyasla daha yüksek bant genişliğine ulaşabildiğini göstermektedir. Sonuçlar ayrıca Geçitli Tekrarlayan Ünite

kullanımının daha yüksek bant genişliği, daha hızlı adaptasyon ve daha düşük yürütme süresi açısından en yüksek performansa sahip olduğunu göstermektedir.

Anahtar kelimeler : 5G Ağları, Pekiştirmeli Öğrenme,Yapay Sinir Ağları, Derin Pekiştirneli Öğrenme, Hüzmeleme.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Pages

LIST OF ABBREVIATIONS

MIMO	:	Multiple Input Multiple Output
BS	:	Base Station
AI	:	Artificial Intelligence
SC	:	Small Cell
ML	:	Machine Learning
RL	:	Reinforcement Learning
ANN	:	Artificial Neural Network
TanH	:	Hyperbolic Tangent
ReLU	:	Rectified Linear Unit
GRU	:	Gated Recurrent Unit
MSE	:	Mean Squared Error
DQN	:	Deep Q-Learning
PDR	:	Packet Delivery Rate
RAM	:	Random Access Memory
GPU	:	Graphical Processing Unit

1. INTRODUCTION

With the growing need for more bandwidth per each user of cellular networks, new technologies are being designed and implemented to meet the required specifications. Recently, the fifth generation of cellular networks (5G) has attracted the attention of many researchers to address the challenges imposed by the new approaches that are used to eliminate limitations in earlier generations [1, 2]. One of the important features that 5G networks rely on to provide the users with more bandwidth is reducing the number of clients per each antenna by increasing the number of antennas, i.e. using massive Multiple Input Multiple Output (MIMO) [3]. Compared to only a dozen antennas in 4G, recent 5G networks can employ up to 100 antennas per each base station [4]. Despite the more bandwidth provided for each client, as the bandwidth of the antenna, i.e. port, is shared by a fewer number of clients, assigning an antenna for a client is a challenging task [1].

The use of mmWave in this generation of cellular communications makes it more sensitive to obstacles in the environment. Hence, the assignment of the antenna to the client can have a significant role in establishing reliable links. In certain situations, the Base Station (BS) resorts to the use of surfaces in the environment to reflect the transmitted waves to the client [1, 4]. The task of assigning the antenna that communicates with the client is known as Beamforming and different techniques are being used to handle such a complex task [5]. The complexity of beamforming is inherited from the complexity of the environment that the BS is operating in. Accordingly, Artificial Intelligence (AI) techniques are being widely used recently to address this problem [6, 7].

As mentioned earlier, the main aim of a new generation of cellular networks is to increase the bandwidth available for each user. This aim is met mainly by the 5G network by increasing the frequency of the carrier signal to use the frequencies up to 300GHz, so that, more bandwidth can be achieved using the same channel. Accordingly, the wavelength of the carrier is reduced to millimeter lengths, i.e. mmWaves, as the wavelength of the signals transmitted at 300GHz frequency is 1mm. Despite the additional bandwidth that becomes available when such frequencies are used, these waves are more sensitive toward obstacles in the environment. To solve this problem, Small Cells (SC) are being distributed in the

environment to relay communications with the clients and reduce the effect of the obstacles in the environment on the links.

Another important approach that is used in 5G networks to provide more bandwidth to each client is the use of MIMO, in which each BS can be equipped with up to one-hundred ports, each is connected to a dedicated antenna. Compared to only a dozen antennas in the 4G networks, the use of MIMO can significantly increase the bandwidth available for each client, as the number of clients that share the bandwidth of a single port is reduced. However, such an increment in the number of antennas declines the use of omnidirectional antennas, which is the case in 4G networks, as the interference among these signals becomes a serious issue that limits the ability to use such networks. To overcome this problem, beamforming is used to control the direction and timing of each packet being transmitted, so that, interference among the signals is avoided and the best link with the clients is established.

A packet that is transmitted from a BS in 5G using beamforming may or may not be transmitted using the same antenna that is used for the same client in the previous packet, even if the position of the client is static. Selecting the antenna that the packet is being transmitted at relies on the overall status of the environment, including all clients, obstacles and the BS itself. To handle the making of such complex decisions, AI techniques, especially Machine Learning (ML), are being used in different approaches. The method used in [8] uses ML to provide adaptive beamforming for 5g networks but this method adapts to changes that occur only in certain parameters, such as weather conditions. Accordingly, if a change occurs in the environment that is not of these parameters, the method fails to adapt it and the quality of the services provided to the clients can be affected. Additionally, RL has been used in [9] to control the beamforming and has been able to significantly improve the network, compared to the use of standard link adaption methods. However, as the agent is pretrained using data that represent certain scenarios, this method fails to take into consideration any additional factors that can affect its performance, e.g. changes in the positioning or shapes of the obstacles in the environment.

Reinforcement Learning (RL) is one of the techniques that has been widely used in recent years to achieve different types of tasks. This type of learning enables the method that is employing it to interact with an environment, by executing actions in that environment. Then,

by measuring the response of the environment to the executed action at a certain state, the RL agent gains the ability to predict that response for each action at a certain state before being executed [10]. In summary, the agent requires a function that approximates the response of the environment, so that, the actions that return the best responses are selected. With their outstanding ability to approximate complex function, Artificial Neural Networks (ANNs) are being used by these agents [11, 12]. Reinforcement learning has been used to handle the beamforming problem in several studies [8, 9, 13]. However, these studies train the neural network to handle a certain environment once. Hence, when any significant changes are presented in the environment, the trained neural network becomes obsolete and loses the ability to perform the required beamforming.

1.1. PROBLEM STATEMENT

The need for additional range of frequencies to handle the rapidly increasing number of devices connected to cellular networks. As the frequency spectrum being used in the current generation of cellular communications is close to the mmWave range, the 5^{th} generation has to include these waves in its spectrum, to allow wider spectrum to provide further devices with more bandwidth and satisfy their requirements. With the lack of these waves to pass through obstacles, such as walls and trees, communications are established with the clients by bouncing these waves on the obstacles available in the environment.

Another important feature that 5G employs to provide higher bandwidths to the clients is the MIMO technology. In these networks the number of antennas can reach up to 100, compared to only a dozen in the 4G networks. To avoid interference among the signals transmitted by the different antennas in the base station, beamforming is used to create directional beams that can be directed at a certain direction, unlike the traditional omnidirectional transmission in previous generation of cellular communications.

1.2. AIM OF THE STUDY

This study aims to provide a new beamforming coordination method using deep reinforcement learning. The aim of the proposed method is to allow automatic learning and updating to the state of the environment, so that, the agent can still provide the highest

possible bandwidth. Additionally, the proposed method is also required to update its performance per each decision it makes, so that, the actual bandwidths achieved by establishing communications using the recommended antenna is used to update the predictions of the neural network that agent uses. Hence, any future predictions can consider the actual values, which allows the proposed method to adapt to any changes in the environment.

1.3. THESIS LAYOUT

The structure of the remainder of this thesis is described below:

i. Chapter Two presents a review of the literature related to the topic of the thesis and illustrates the methodologies of the methods employed in the proposed method.
ii. Chapter Three describes the proposed method in details, including how the status of the client is represented to the neural network and how this network is trained and updated during operation.
iii. Chapter Four illustrates the experiments conducted to evaluate the different types of artificial neural networks in the proposed method.
iv. Chapter Five summarizes the results of the conducted experiments, compares them to each other and to the state-of-the-art methods that exist in the literature. These comparisons illustrate the benefits of the proposed method and how it can be used to improve the performance of the 5G cellular network.
v. Chapter Six summarizes the conclusions of the current work and directions of future work.

2. LITERATURE REVIEW

2.1. REINFORCEMENT LEARNING

Reinforcement learning uses the concepts of agents, environments, states, actions and rewards [14-16]. As shown in Figure 2.1, 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 [17].

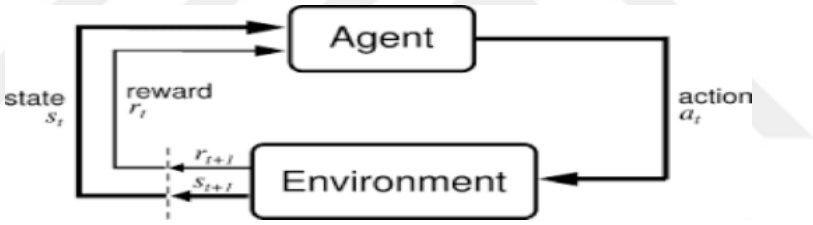


Figure 2.1: Illustration of the interaction between the Agent and the Environment in reinforcement learning.

The main components in RL applications are defined as follows:

i. **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.

ii. **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.

iii. **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.

iv. **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.

v. **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.

vi. **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.

vii. **Policy (π):** Is the approach employed by the agent to select the action appropriate for the current state to maximize the reward.

viii. **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.

ix. **Q-Value (Q):** This value defines the overall reward for a certain action at a certain state, i.e. $Q^{\pi}(s, a)$. 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 [18, 19]. This reward is calculated as shown in Equation 2.1.

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

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.

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

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.

2.2. ARTIFICIAL NEURAL NETWORKS

Inspired from humans' brains, computations in ANNs are implemented in units, known as artificial neurons, distributed over the network in layers. The inputs of a certain neuron can be collected from the external domain or from the outputs of the previous layer's neurons. To calculate the output of a neuron, all collected inputs are weighted, by multiplying each of them with a certain value assigned per each input and summed, before being passed through a nonlinear function, known as activation function, as shown in Figure 2.2. This nonlinearity provides more flexible output that has the ability to detect more complex features. Nevertheless, additional value can be added to the inputs of a neuron to provide bias to the computations, when needed, known as the bias [20, 21].

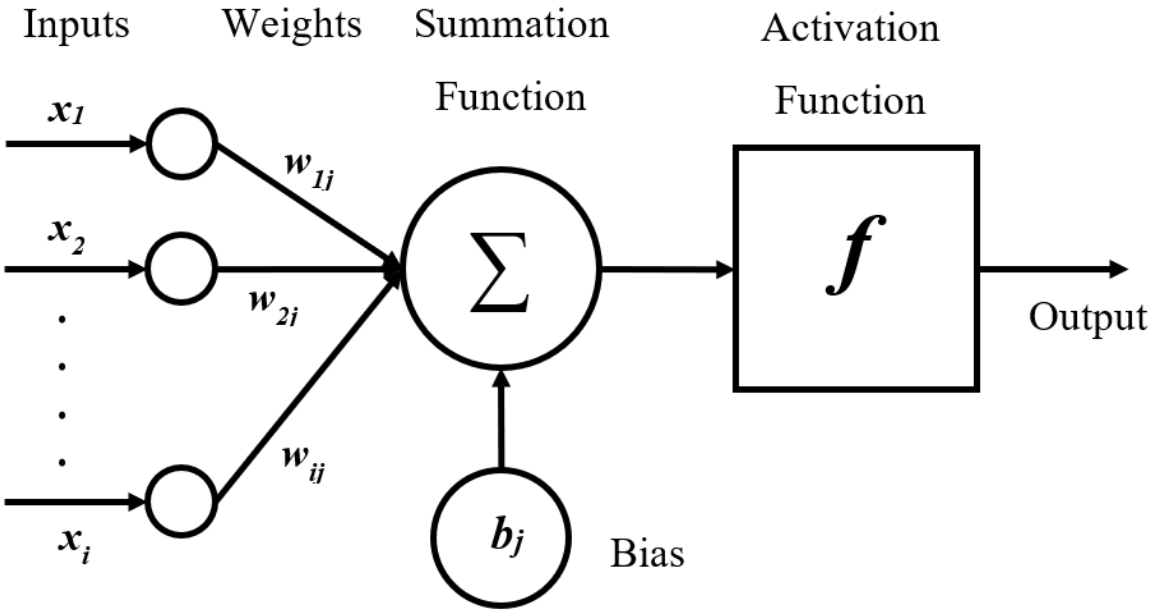


Figure 2.2: Illustration of the computations inside an artificial neuron [20].

Passing the result of the summation into an activation function provides the neuron with the ability of creating nonlinear boundaries for decision making. i.e., if an activation function is not included in the computation of the neuron, the only possible boundary that a neuron can use to split the tuples in the dataset into classes is a linear boundary, which reduces the ability of providing more accurate predictions. Moreover, neurons located deeper in the neural network would have the ability of creating more complex boundaries for each class, which

also improves the accuracy of the predictions provided by the entire neural network. In addition, the use of the bias values within each neuron can assist the creation of these complex boundaries by adjusting the locations of each part of the complex boundary, which is created by combining boundaries of neurons prior to that neuron. Some of the widely used activation functions are the Sigmoid, Hyperbolic Tangent (TanH) and Rectified Linear Unit (ReLU) [22], which are shown in Figure 2.3. However, neural networks with ReLU activation functions have shown significantly better performance than the other activation functions [23, 24]. This non-linearity of the computations allows the output to be calculated from the inputs by detecting the required features. However, as the neural network can follow different routes to reach a certain output, and as an output can be a result of a single feature of a combination of multiple features, these networks are being used as one-way functions to generate hash values that can be used to describe an input, whereas the hash value cannot be used to retrieve the original input [25-27].

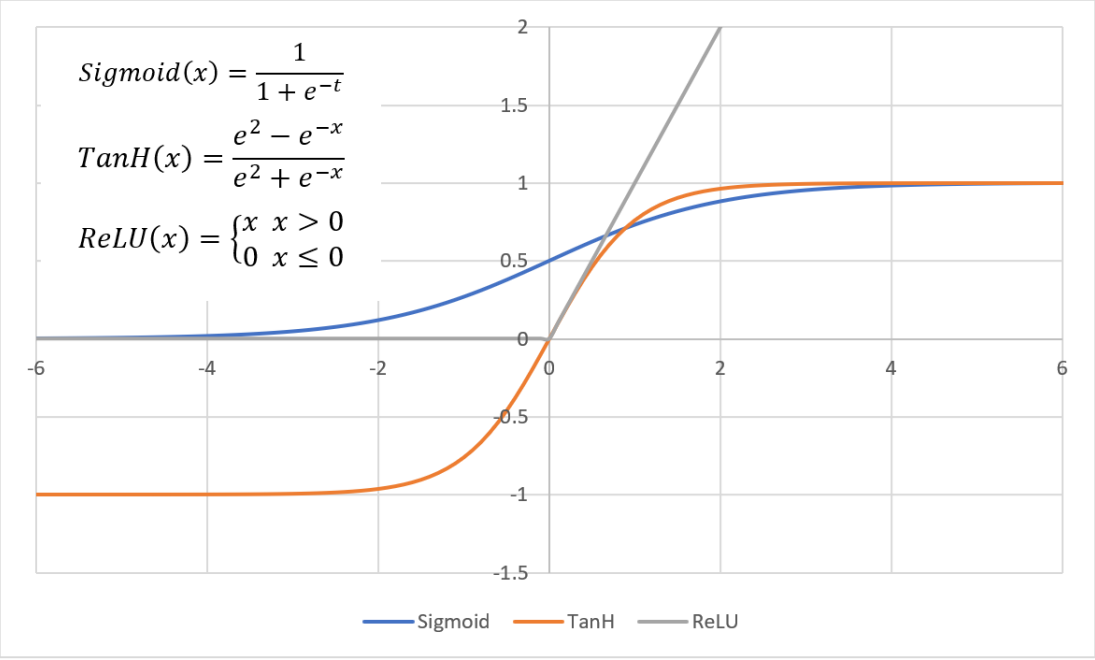


Figure 2.3: Activation Function for Neurons [23, 28].

Regardless of the type of the ANN, each of these networks has two types of computations, one executed from the input to the output direction, known as the forward pass, while the other is executed in the opposite direction, known as the reverse pass [29]. The forward pass

is used to calculate the output of the network, based on its inputs, by calculating the output of each layer and use in the computations executed in the second one. In the reverse pass, the weights' values are updated through gradient descent. By measuring the deviation between the output of the ANN, from the forward pass, and the intended output values, from the dataset, the derivatives of the output to the weights are calculated. Gradient descent is used to recognize the position weights' value must be updated to reduce that error, which is to the negative of the gradient decent at that position. Such update allows the neural network to produce the intended output from the inputted values, hence, achieve the required task. By repeating this process for several iterations, the loss between the output from the forward pass and the intended output is reduced using backpropagation, which improves the performance of the neural network, until the minimum loss is reached [30, 31].

2.2.1. Convolutional Neural Networks

CNNs contain convolutional layers, which consists of two-dimensional filters that are convoluted throughout the input of each neuron. Mathematically, the filter is actually the weight values of that neuron, which enable the neuron to detect local two-dimensional patterns in the input. The sizes of the filters in a convolutional layer is constant and patterns in the input can be detected within the size of the filter. However, by going deeper into the neural network, i.e. layers farther from the input layer, each filter detects patterns defined by the patterns detected by the previous layer's filters. This enables the CNN to combine the recognized patterns and detect more complex features. Although the output of a neuron in a convolutional layer can have different dimensions from its input, the number of dimensions is similar to that in the input, i.e. a neuron processing a two-dimensional input outputs a two-dimensional array [32, 33].

During convolution, the number of values that the filter moves per each step is defined as the strides, which can have different values for the horizontal and vertical movements. All the values within the filter are multiplied with their corresponding weights and processed in the neuron, which arranges its outputs according to the arrangement received during the convolutions of its filters. Skipping more than one value per each convolution can cause the loss of detecting important patterns, which can negatively affect the performance of the CNN,

despite the reduction in the size of the neuron's output, which can simplify the computations in following layers. To reduce the size of the output from a neuron without losing important information, pooling layers can be placed after a convolutional layer [34].

A pooling layer also consists of filters that are convoluted throughout its input, which is the output of the neuron. However, these filters have a different approach to process the input values, as they are not forwarded to a neuron and has no weights. Despite the existence of different types of pooling layers, Max-Pooling layer is one of the widely used pooling layers that are used to reduce the size of the processed data without losing important information. As shown in Figure 2.4, the filter in a max-pooling layer searches for the maximum value within its dimensions, and outputs that value to represent that region. By selecting the highest value, the most important feature in that region is selected, so that, it is less likely to lose important information as in increasing the strides of the filter in the convolutional layer [34].

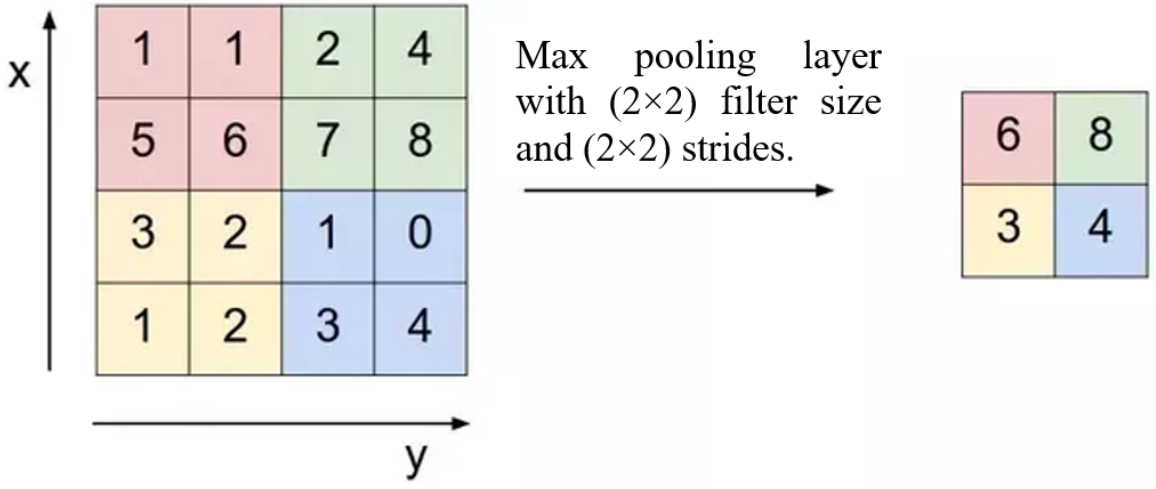


Figure 2.4: Output of Max-Pooling filter.

According to the ability of CNNs to consider the position of an input, in addition to its value, these networks are being widely employed in NLP. For example, such network can recognize that the phrase *"does not exist"* is equivalent to the word *"absent"* in a sentence, so that, the effect of these two neurons can be similar with respect to the output of the neural network. Moreover, when the output required from the neural network is not two-dimensional, which is the case in most applications, the output of the last convolutional layer can be flattened and fully connected to another one-dimensional layer. Depending on the complexity of the

features in the input, more layers can be added to the neural network before the output layer [35, 36].

2.2.2. Recurrent Neural Network

Similar to CNNs, recurrent neural networks can handle two-dimensional inputs and output a single value per each set of inputs. However, the approach RNNs use to process these inputs is different, where the output from a previous input tuple is weighted and appended to the inputs collected from the previous layer, or the external domain. As shown in Figure 2.5, suppose a weight value *f* is used to adjust the value of the output from the tuple previous to the current tuple positioned at *t*. During the computations of the output of the neuron at *t*, the output *h* from *t-1* is included after being weighted using *f*. The output at this *t* tuple is also weighted using *f* and included with the inputs *x* of the next tuple at *t+1*. This process is repeated until all the tuples in the input set are processed [37, 38].

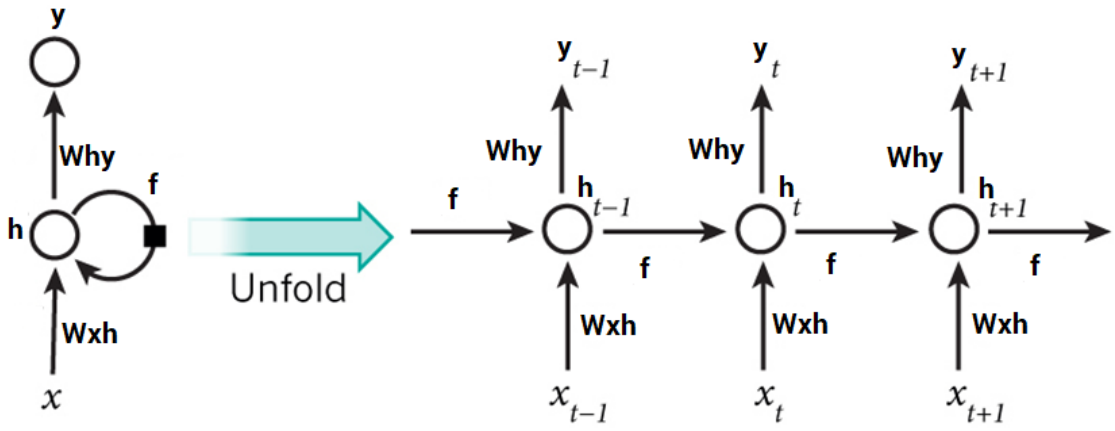


Figure 2.5: Computations in an RNN neuron.

According to the ability of RNN's to include outputs calculated from previous tuples in the computations of the current one, this type of neural networks is widely used in timeseries and NLP applications. A phrase can be analyzed according to the effect of each word in that phrase and its position. For instance, the output of processing a negative word, such as *not*, can be combined with the inputs of the next word, so that, the meaning of that word can be inverted. Moreover, errors can be detected by recognizing wrong combinations, when a word

following another is in wrong formation, depending on the definition of the suitable form in the grammar [39, 40].

2.2.2.1. Long- Short-Term Memory

As illustrated in the previous section, the effect of a certain output from the neuron is relative to the position of the tuple being inputted to the network, with respect to the one being processed in this instance. At instance *t*, the output from *t-1* has more influence on the current output than that from *t-2*. However, in many applications including NLP, such behavior can be of significant importance in certain conditions, and of negative influence in other. Thus, a more complicated type of RNNs is being used in these applications, where the influence of a certain output is adjusted according to its importance in the current computations, rather than its position in the series [41].

To achieve such a task, LSTM networks use gates to control the flow of the values between the input and the output. Each gate is controlled using a separate network that accepts inputs from certain position. As shown in Figure 2.6, net_c is the input network that receives the values from the external domain and calculates the outputs depending on its weights. Another network net_{in} receives a copy of these inputs in order to control the gate that defines the flow of the output from net_c, through the input gate value y^{in}. The effect of the previous output is adjusted using the forget gate values y^{ϕ}, which is controlled using net_{ϕ}. This output S_c is squashed using an activation function before being adjusted using the values y^{out} acquired from the output gate, which is controlled using net_{out} that calculates the values of the gate using the outputs collected from the previous time instance. As each gate is controlled using a different neural network, the weights of each neural network are updated during the training of the networks, so that, the appropriate decision is made based on the input values of the current time instance and the outputs collected from the previous ones [42].

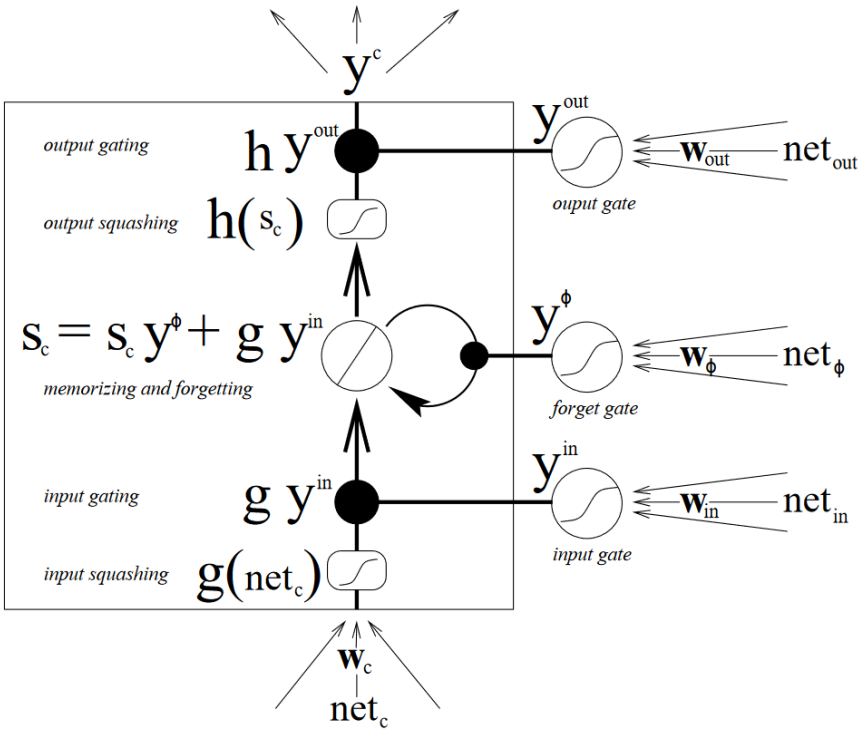
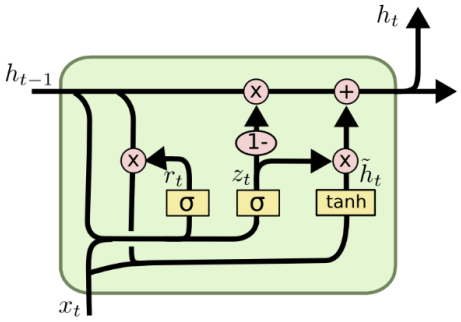


Figure 2.6: Illustration of the data flow in an LSTM neural network [42].

2.2.2.2. Gated Recurrent Units

To reduce the complexity of the LSTM, Gated Recurrent Unit (GRU) has been proposed to avoid the exploding and vanishing gradient problem using lower computations. A GRU contains two gates to control the flow of the values through the neuron, which are the reset and update gates, as shown in Figure 2.7 [43]. The reset gate controls the effect of the values outputted from the previous timestep, depending on the importance of those values in the computation for the current input. The update gate controls the effect of the current input on the output of the unit, so that, the output can consider both the current and previous values depending on the decision made at these gates. Such topology achieves the same methodology of the LSTM using fewer computations, as it uses fewer gates. However, the qualities of the predictions for both methods are very similar and both methods must be evaluated in order to select the appropriate method for the required application [44-46].

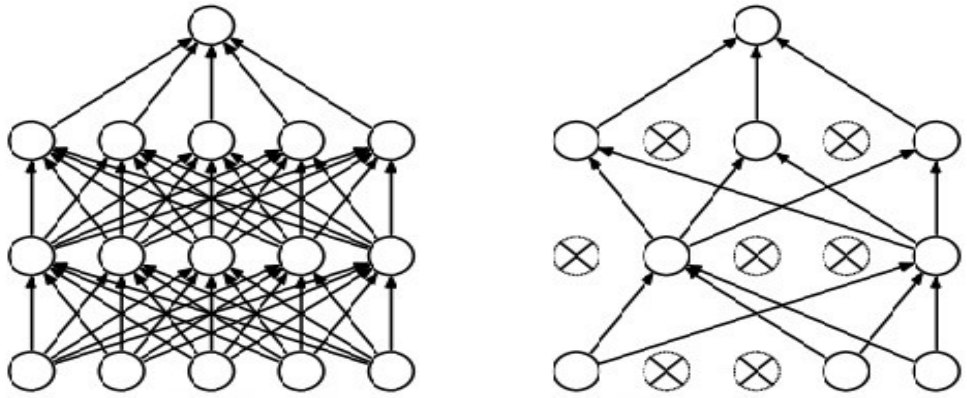


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2.7: Gated Recurrent Unit.

2.2.3. Overfitting in Artificial Neural Networks

One of the main challenges faced by deep neural networks is the phenomenon of overfitting, where the predictions are based on specific features in the neural network, which makes these predictions very restrict to these features. Thus, any new inputs that may belong to that class but do not fire the neurons corresponding to these features are most probably are going be wrongfully classifies. To overcome such problem, a predefined ratio of the neurons in a hidden layer are randomly dropped per each iteration of the training phase, so that, the neural network is forced to find alternative paths to the same prediction and reduce the dependency on specific features. This approach is known as Dropout and has shown good improvement in the predictions provided by neural networks [47], it is shown in Figure 2.8.



a-Standard Neural Network b-After applying dropout

Figure 2.8: Illustration of dropout in artificial neural networks [47].

2.2.4. Training Artificial Neural Networks

Similar to humans' brains, where the topology of the biological neural network and the conductivities of the synapses define the decisions made by the brain, ANNs also rely of the distribution of the neurons and the weights among them to make the required decision. Two identical neural networks can be used in completely different task, in which different decisions are made, by using different weights values among their neurons. The value of a weight between two neurons defines the type of the effect, the output of the neuron in the previous layer over the neuron in the next one, as well as the significance of that effect on the output of the neuron in the later layer [27, 48].

Backpropagation has a key-role in the popularity of neural networks, as the performance of these networks is significantly improved when this technique is used to update the value of the network's weights. In order to update the weights of the ANN, backpropagation requires three values, as shown in Equation 2.1, which are the rate of change of the network's output, with respect to the loss being updated $\frac{\partial o}{\partial w}$, the error E between the output of the network and the one actually required from it and the learning rate L [49].

$$\widehat{w} = w - \frac{\partial O}{\partial w} \times E \times L \tag{2.1}$$

Regardless of the type of error function used by the neural network, such as the cross-entropy and Mean Squared Error (MSE) functions, these functions calculate a single value that represents the difference between the output of the neural network, using the current weights values, and the values required from the network. The output of the neural network is collected by processing a batch of sample inputs, from the training dataset, using the forward pass of the neural network, while the actual outputs are collected directly from the training dataset, or by processing the inputs using predefined functions. The calculated error value is then used by in the backpropagation. However, as large error values can produce large delta values, for weights updates, a learning rate is used to control the delta values in lower ranges. This control of the delta values ensures the avoidance of exploding weight values, so that, the weights values that produce the minimum error can be discovered [50].

By calculating the rate of change of the output error, with respect to the weight values, three possible values can be produces [51, 52], which are:

i. A positive value, which indicate that increasing that weight value increases the error. Thus, the weight value must be decreased by the calculate delta value, in order to decrease the difference between the outputs of the neural network and the required ones.
ii. A negative value that indicates that the error is decreased by increasing the value of that weight. Thus, the current weight value must be increased by the calculated delta value, in order to reduce the error value and produce more accurate outputs.
iii. A zero value, which indicates that no change is required to the current value of the weight.

According to these possible values and by using the formula shown in Equation 2.3, the values of the weights in the neural network can be updated in order to reduce the difference between the predictions of the neural network and the actual output that is required to achieve the task of the ANN. However, according to the need of learning rate, to reduce the delta value used to update the weight values, the optimal performance of the neural network,

produced by minimizing the error through updating the weights, calculating the optimal weights values require multiple iterations, i.e. epochs [53].

2.3. 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 [54]. 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 [55]. 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 [56].

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 [57]. 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 [58, 59].

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 [18, 60]. 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 [61, 62]. Thus, DQN has been widely used in approximating the functions of complex environments, such as those faced by autonomous vehicles drivers.

2.4. ARTIFICIAL INTELLIGENCE AND BEAMFORMING IN 5G NETWORKS

As mentioned earlier, the main aim of a new generation of cellular networks is to increase the bandwidth available for each user. This aim is met mainly by the 5G network by increasing the frequency of the carrier signal to use the frequencies up to 300GHz, so that, more bandwidth can be achieved using the same channel. Accordingly, the wavelength of the carrier is reduced to millimeter lengths, i.e. mmWaves, as the wavelength of the signals transmitted at 300GHz frequency is 1mm. Despite the additional bandwidth that becomes available when such frequencies are used, these waves are more sensitive toward obstacles in the environment. To solve this problem, Small Cells (SC) are being distributed in the environment to relay communications with the clients and reduce the effect of the obstacles in the environment on the links.

Another important approach that is used in 5G networks to provide more bandwidth to each client is the use of MIMO, in which each BS can be equipped with up to one-hundred ports, each is connected to a dedicated antenna. Compared to only a dozen antennas in the 4G networks, the use of MIMO can significantly increase the bandwidth available for each client, as the number of clients that share the bandwidth of a single port is reduced. However, such an increment in the number of antennas declines the use of omnidirectional antennas, which is the case in 4G networks, as the interference among these signals becomes a serious issue that limits the ability to use such networks. To overcome this problem, beamforming is used to control the direction and timing of each packet being transmitted, so that, interference among the signals is avoided and the best link with the clients is established.

A packet that is transmitted from a BS in 5G using beamforming may or may not be transmitted using the same antenna that is used for the same client in the previous packet, even if the position of the client is static. Selecting the antenna that the packet is being transmitted at relies on the overall status of the environment, including all clients, obstacles and the BS itself. To handle the making of such complex decisions, AI techniques, especially Machine Learning (ML), are being used in different approaches. The method used in [8] uses ML to provide adaptive beamforming for 5g networks but this method adapts to changes that occur only in certain parameters, such as weather conditions. Accordingly, if a change occurs in the environment that is not of these parameters, the method fails to adapt it and the quality of the services provided to the clients can be affected. Additionally, RL has been used in [9] to control the beamforming and has been able to significantly improve the network, compared to the use of standard link adaption methods. However, as the agent is pretrained using data that represent certain scenarios, this method fails to take into consideration any additional factors that can affect its performance, e.g., changes in the positioning or shapes of the obstacles in the environment.

3. METHODOLOGY

3.1. OVERVIEW

The proposed method uses a RL agent to govern beamforming in the 5g network by collecting information about the client from all base stations in order to select the antenna that is expected to maximize the bandwidth between the cellular network and the client. As shown in Figure 3.1, when a client requests a connection to the network, the signal between the device and each antenna in the network is measured and delivered to a centralized server. This server then uses the RL agent to predict the bandwidth at each antenna and use the one with the highest expected bandwidth. When the connection is established, the actual bandwidth is measured and used to update the neural network, so that, it can handle any variations in the environment, without the need to manually update its parameters.

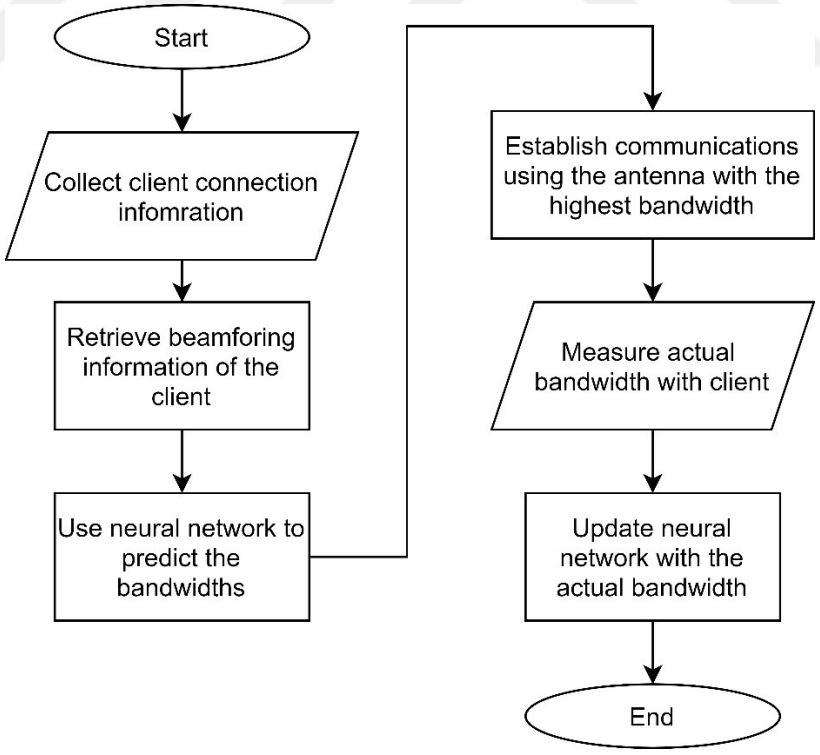


Figure 3.1: Overview of the proposed methodology.

3.2. DATA COLLECTION AND CLIENT REPRESENTATION

When an antenna receives a signal from the client, the strength of the signal is measured and delivered to the server that is designated for beamforming management. Another piece of information is appended to the received signal strength value, which indicates whether that antenna has been used by the beamforming coordination server to establish communications with that client at that time instance or not. Accordingly, for a system with N BSs and M antenna in each BS, a total of $2 \times N \times M$ values are collected for each packet received from that client.

In addition to the possible variations in the environment, the mobility of the client poses another challenge towards the beamforming task, which is represented by the need to predict which antenna has the ability to maintain communications with the client based on their movement. By considering such scenarios, the proposed method can favor one antenna over another, based on the behavior of the client, in terms of movement, and the environment that the client is moving in, e.g., the obstacles in that environment. To provide the neural network with such representation, historical data is provided, in addition to the data collected from the current time instance. Hence, for each antenna selection, the proposed method provides the neural network with $100 \times 2 \times N \times M$ values, which represent the measures collected by all the antennas that are in the network for the latest 100 time-instances.

3.3. ANTENNA SELECTION USING RL

When a packet is received from the client, the antenna that is designated to reply that packet, i.e., establish communications with the client, is selected by passing the data that represent the client to the neural network that the RL agent uses to predict the bandwidth of each antenna, if selected. The output of the neural network, which represents the normalized expected bandwidth for each antenna in each BS, is then used to select an antenna. However, to allow the proposed method to maintain exploration during the operation but avoid frequent use of antennas with low expected bandwidth, the probability of using an antenna is equal to the normalized predicted bandwidth, which is normalized to the maximum bandwidth of the antenna. Accordingly, the chances that the proposed method attempt antennas that are predicted to have higher bandwidths are significantly higher than antennas that are expected

to have lower bandwidths. To achieve such a selection, the proposed method uses the approach shown in Figure 3.2, in which the ID of an antenna is repeated *t* times, which represents *100×normalized bandwidth* of the antenna. Then, an antenna is selected randomly from the generated list. According to this approach, the high frequency of IDs of the antennas that are predicted to have high bandwidth increases the chances of selecting such an antenna, whereas the absence of the antennas that have not received the packet, i.e., cannot establish communications with that client, eliminates the probability of selecting such an antenna, which may interrupt the communications with the client.

Input: Predictions of the neural network. **Output:** The selected antenna.	
Step1:	P ← Predictions of the neural network. //Read the predictions of the neural network.
Step2:	A = [] for i in range(Len(P)): //For each antenna in the system. for j in range(100×P[i]+1): //Repeat for 100×predicted normalized bandwidth. A.append(i) //Add the antenna ID to the list
Step3:	S ← Select an antenna from A randomly.
Step4:	Return S

Figure 3.2: Antenna selection algorithm.

3.4. STRUCTURE OF THE RL NEURAL NETWORK

In order to predict the normalized bandwidth of each antenna, a neural network is implemented for the RL agent in the proposed method, which is responsible for selecting the antenna. The implemented neural network, shown in Figure 3.3, consists of three hidden layers, in addition to the input and output layers. The first two hidden layers are followed by dropout layers, whereas the third one is not followed by a dropout layer to avoid affecting the values in the output layer. All hidden layers use ReLU activation function, whereas the

output layer uses Sigmoid function, as this function limits the output to the interval [0, 1], which represents the normalized bandwidth value. The input layer is set to handle $100 \times 2 \times N \times M$, as described in Section 3.2, whereas the output layer contains only $N \times M$ neurons, as each neuron represents the normalized bandwidth that is predicted to be achieved by the corresponding antenna if used to establish communications with the client.

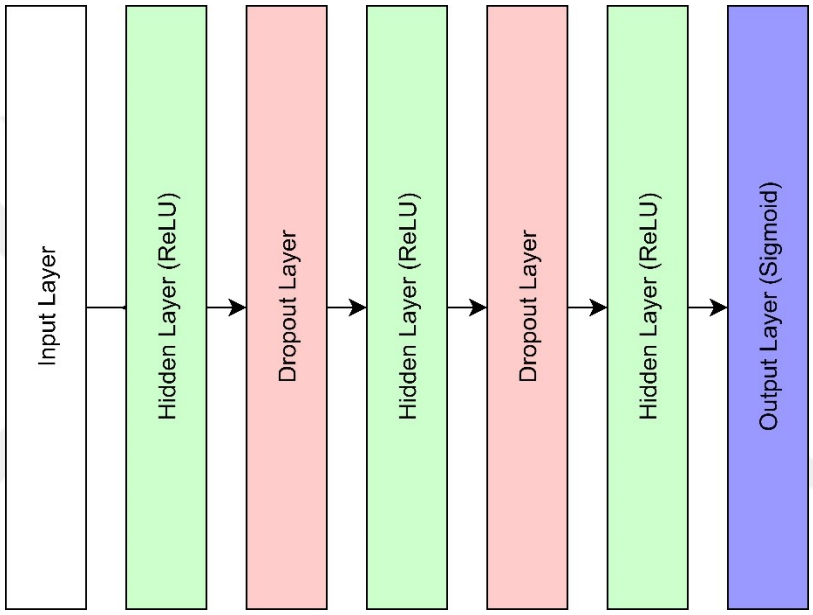


Figure 3.3: Structure of the neural networks implemented for the RL agent.

Different types of neurons are evaluated in the proposed method, in order to recognize the one that is suitable for the beamform coordination application. The selected types are either CNN or RNN neurons, according to their ability to handle multi-dimensional inputs. According to this ability, these neurons can predict the behavior of the client and the antenna suitable for that behavior by processing the historical information collected from the different antennas. Eventually, the outputs of the neural network are used to select the suitable antenna, based on the approach described in Section 3.3.

3.5. TRAINING THE NEURAL NETWORK

One of the most important features of the proposed method is its ability to adopt to the changes that may occur in the environment by simply measuring the actual bandwidth that is

achieved between the network and the client, when established using the selected antenna. This feature is inherited from the use of RL in the proposed method, in which the predictions of the neural network are updated in realtime. Such updates allow the proposed method to accommodate to any changes that may occur in the environment, combined with the antenna selection approach based on the probability of achieving high bandwidth. To update the neural network, the predictions of the network are collected. Then, the actual bandwidth is calculated for the antenna, as shown in Equation 3.1. This value is then normalized and placed at the output correspondent to the antenna that is actually selected to establish the communications. Hence, if any changes occur to the bandwidth, according to any changing variables in the environment, the neural network is updated automatically, which allows it to recognize better alternatives to the selected antenna to maintain high bandwidths.

$$R = PDR \times \frac{b}{B} \tag{3.1}$$

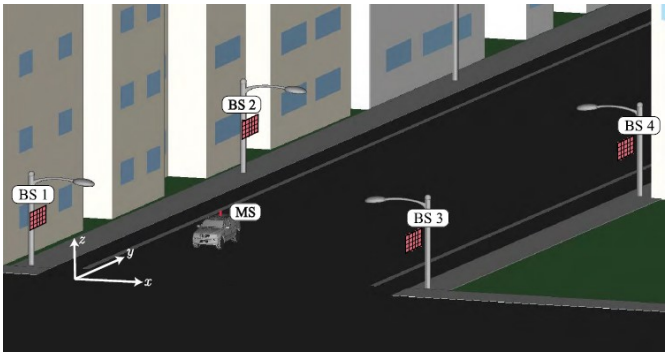
where,

R is the calculated reward value, based on the packet delivery rate (PDR) and the actual bandwidth b to the bandwidth of the antenna B.

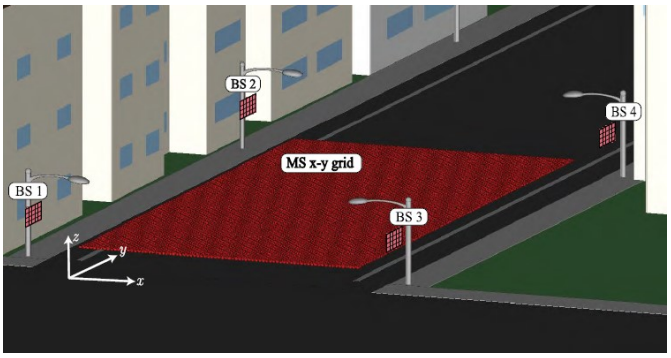
4. EXPERIMENTAL RESULTS

In order to evaluate the performance of the proposed method, the environment simulated by Alkhateeb et al. [13], which is available at [63]. As shown in Figure 4.1, this setup simulates a mobile client in a vehicle moving among four base stations. Then, a bus interrupts the environment, which requires changes in the beamforming decisions made by the decision-making method. The proposed method is implemented using Python programming language [64], where the Tensorflow library [65] is used to implement, train and use the neural network employed by the RL agent. All experiments are conducted using an Intel Core i7 processor running at 2.4 GHz with 16 GB of Random Access Memory (RAM). Additionally, the computer also contains an Nvidia Graphical Processing Unit (GPU), which has the ability to parallelize and accelerate the computations of the Tensorflow library, which are required to train the neural networks and compute their outputs during runtime.

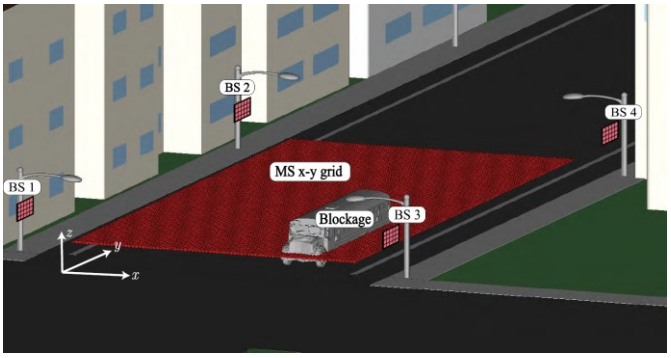
.



(a)



(b)



(c)

Figure 4.1: Illustration of the simulated scenario. (a) The vehicle with the moving client. (b) The grid covered by the four base stations. (c) A bus interrupting the environment.

4.1. EXPERIMENT A – USING CNN

In this experiment, convolutional neurons are used in the model described in Section 3.4, where each neuron uses a 2×2 filters and followed by a 2×2 max-pooling layer to reduce the dimensionality of the array forwarded to the following layer. As shown in Figure 4.2, the

proposed method using CNN has been able to achieve high bandwidth rates, compared to the maximum bandwidth that the antenna can achieve. The proposed method in this experiment has been able to the learn the correct assignment that can increase the bandwidth of the established link, by choosing the suitable antenna. These results validate the ability of using RL and CNN to address the beamform coordination online, i.e., without the need to collect any prior data. Hence, the proposed method can also update its performance when any permanent changes are proposed to the environment, as the actual bandwidth achieved is used to train the neural network and update its predictions, which in return updates the decisions made for antenna assignment. Moreover, the ability to use an antenna with a competitive bandwidth has accelerated the learning process, as multiple antennas are evaluated until the best one is recognized based on the conditions of the environment and the client.

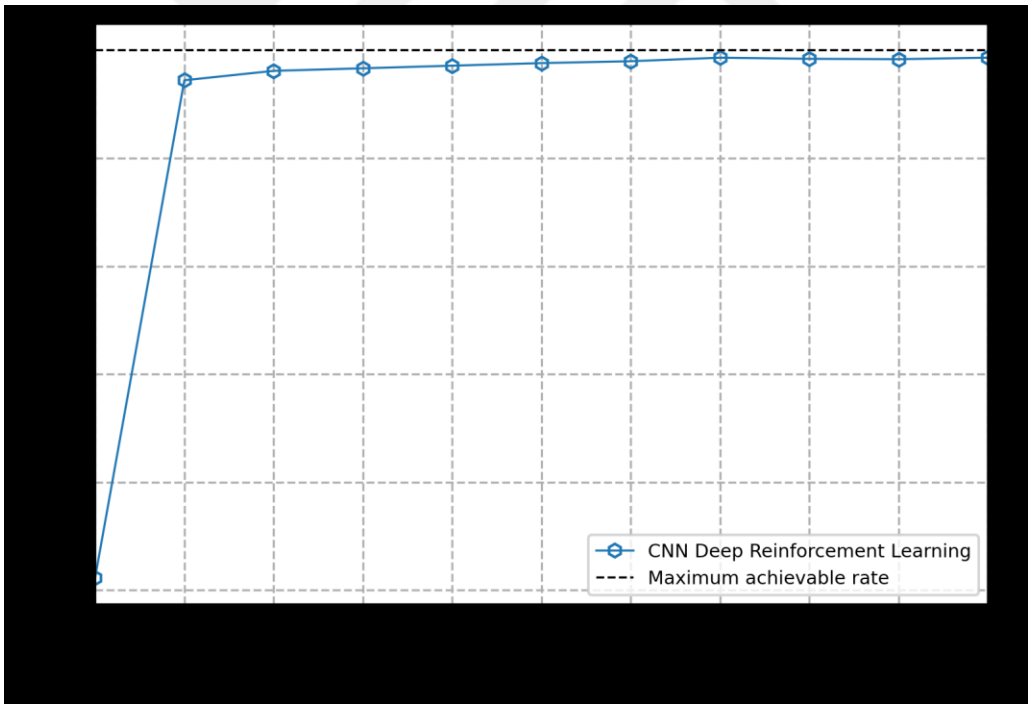


Figure 4.2: Bandwidth achieved by the CNN without interruption.

Additionally, the proposed antenna selection method has shown the ability to improve the decision by trying competitive antennas when the communications are interrupted, as shown in Figure 4.3. This update is a result of updating the predictions of the neural network to reduce the bandwidth of the selected antenna, as its actual bandwidth is reduced. Hence, the

antenna, with a similar or better predicted bandwidth, compared to the antenna being currently used is selected for to maintain communications and bandwidth. Moreover, these results also illustrate the ability of the proposed method to rapidly adopt to any changes in the environment, illustrated by its ability to increase the bandwidth when the bus has interrupted the communications. In addition to the achievable bandwidth, the use of the CNN in the proposed method has consumed an average of $0.31\mu s$ per each prediction, which is measured as a representation of the complexity of the model.

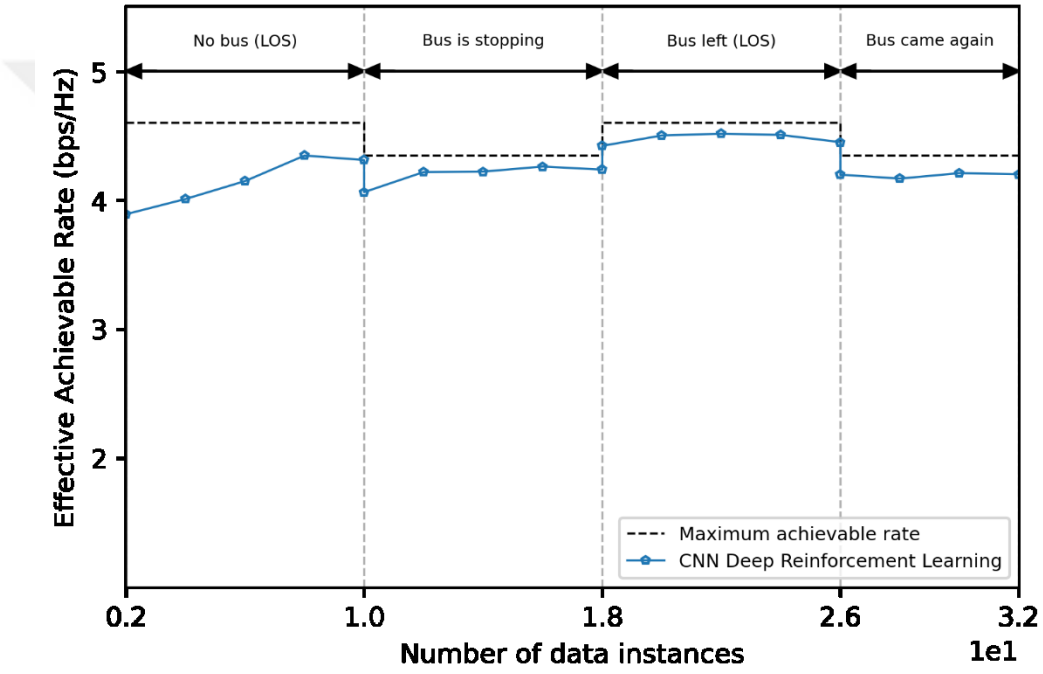


Figure 4.3: Performance of the proposed method using CNN neural network with the bus interrupting the environment.

4.2. EXPERIMENT B – USING RNN

In this experiment, two typos of RNN are evaluated, which are the LSTM and GRU. As shown in Figure 4.4, the GRU has been able to achieve higher bandwidth, with slightly faster learning, illustrated by its ability to improve the bandwidth faster. Additionally, the LSTM network has shown fluctuations in the achieved bandwidth, which indicates that it has outputted similar predictions for antennas that have achieve lower actual bandwidth, compared to the predicted bandwidth. Despite the reduction in the bandwidth at such conditions, these fluctuations and their return to high bandwidths illustrate the ability of the

proposed method to rectify any errors in the predictions and adopt to any changes in the environment.

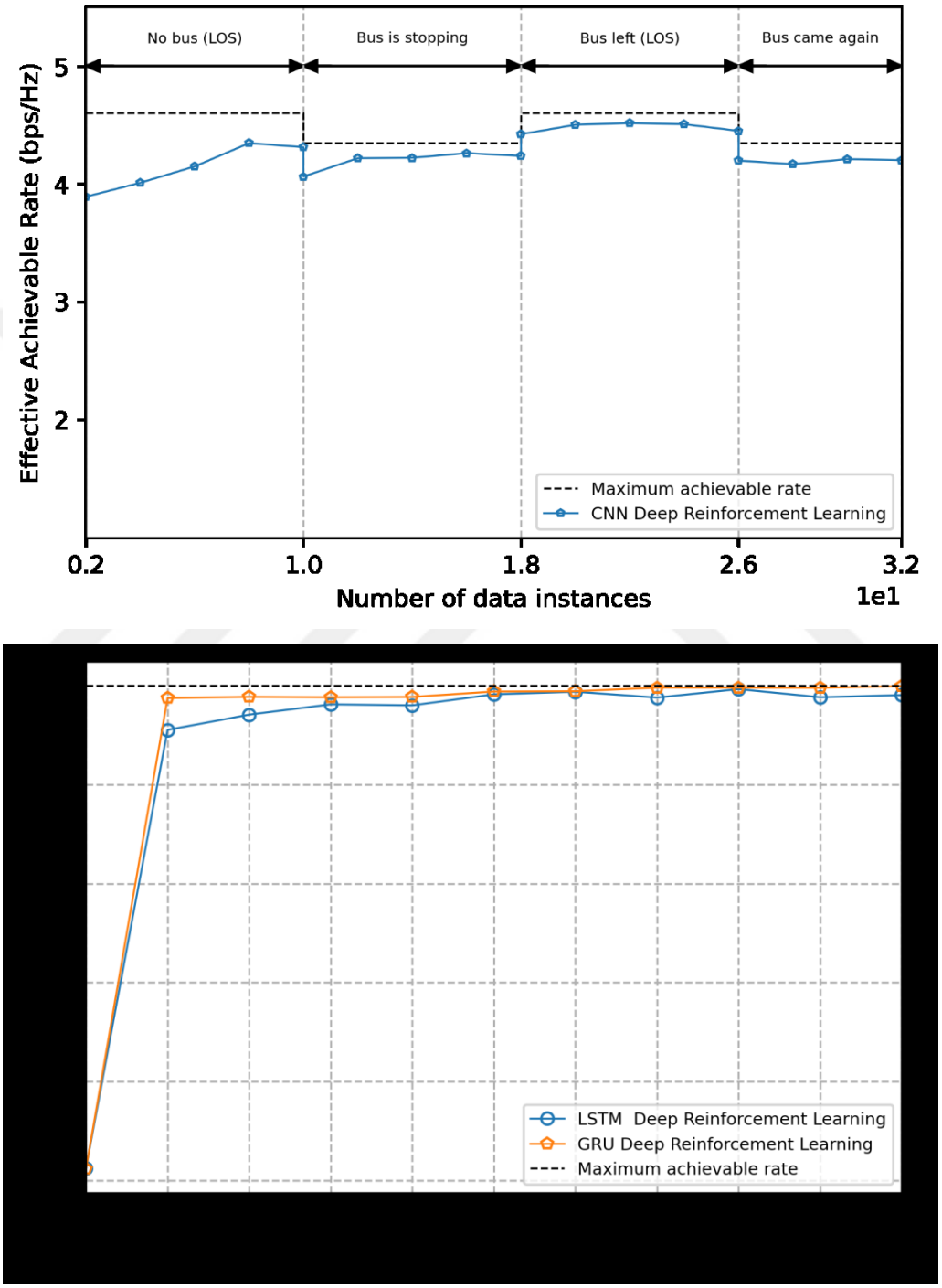


Figure 4.4: Bandwidth achieved by the RNNs without interruption.

The faster learning of the GRU is a result of its lower complexity, compared to the LSTM, which is also the reason behind providing faster predictions, with only $0.28\mu s$, compared to

0.42μs required by the LSTM. The longer time required by the LSTM is a result of using more computations in its structure to govern the flow of the data, which despite the ability to the GPU to parallelize these computations, still affect the execution time.

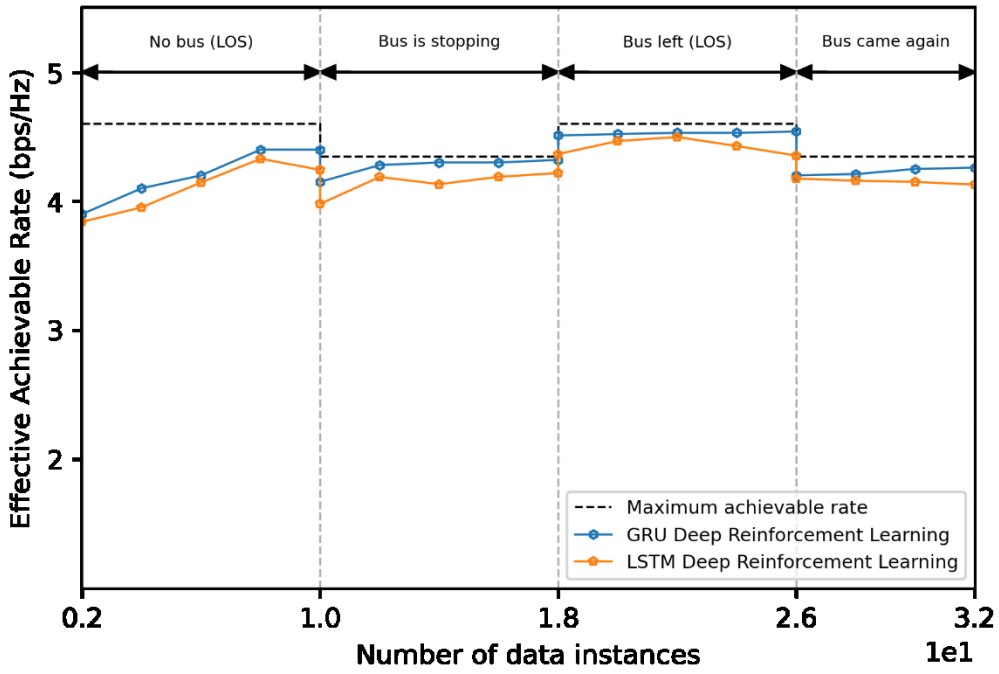


Figure 4.5: Bandwidth achieved by the RNN models with the bus interrupting the environment.

5. DISCUSSION

The summary of the results collected from the conducted experiments, shown in Figure 5.1, show that the GRU has achieved the highest bandwidth among the evaluated neural network units. Combined with the lower execution time achieved by the GRU, shown in Figure 5.2, the results show that this unit is most suitable for beamforming coordination. This performance is according to the ability of GRU units to efficiently consider historical data, compared to the more complex computations required by the LSTM, which requires additional execution time, and the limited ability of CNN to consider only the values that are in the same filter. Additionally, the comparison shown in Figure 5.1 illustrate the improvement achieved by using RL instead of regression approach, by achieving higher bandwidth using CNN, which is also used by the regression-based method proposed by Alkhateeb et al. [13].

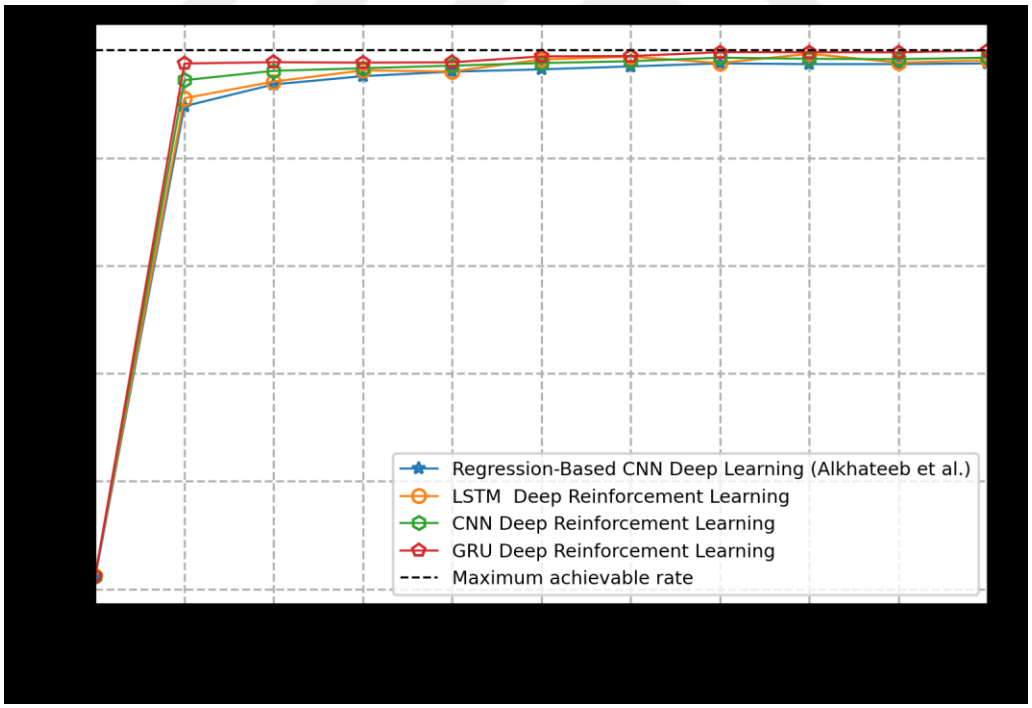


Figure 5.1: Comparison of the bandwidths achieved by the different neural networks in the proposed method.

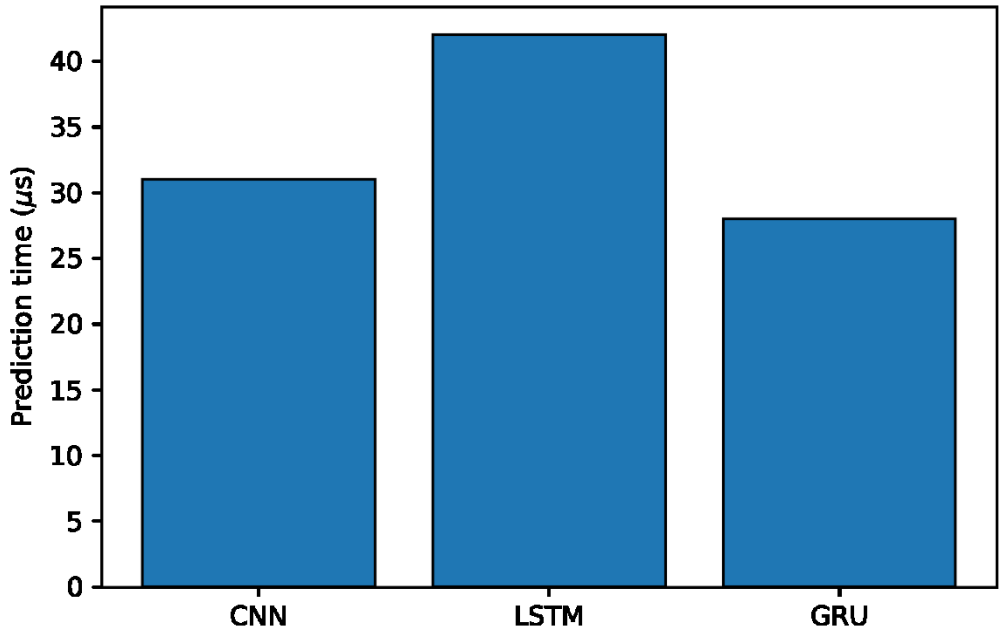


Figure 5.2: Illustration of the average prediction time required by the neural networks in the proposed method.

In addition to the higher bandwidth achieved by the proposed method, the proposed antenna selection method has allowed the method to adopt faster than the method proposed by Alkhateeb et al. [13], as the best antenna is selected by attempting several competitive antennas that are predicted to achieve the highest bandwidth. As shown in Figure 5.3, the method proposed by Alkhateeb et al. [13] has almost linear improvement, based on the behavior of the regression approach. Alternatively, the proposed method has shown faster adoption to the changes and faster reach to high bandwidths as it uses several antennas with similar predicted bandwidth and emphasizes the one that achieves higher bandwidth than the others.

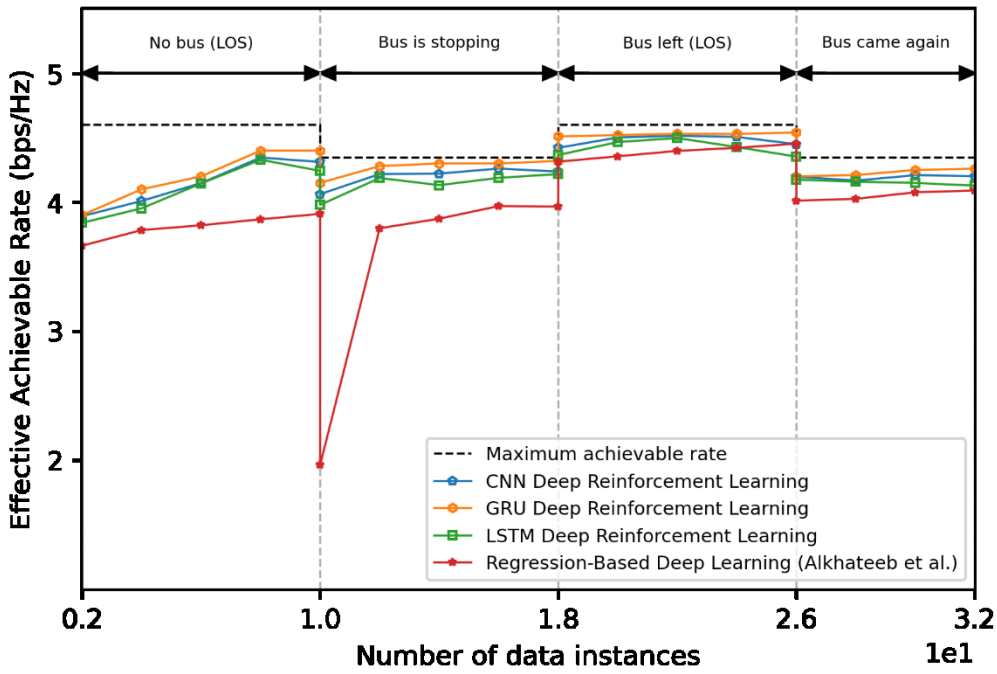


Figure 5.3: Comparison of the achieved bandwidths when the bus interrupts computations.

Another important behavior that is shown by the proposed method during the evaluation is its ability to maintain previous knowledge in future decisions. This behavior is illustrated in Figure 5.4 by the green and red horizontal lines, which mark the bandwidth of the network when the bus leaves and returns for the second time. These bandwidths are larger than the ones achieved by the network at the same conditions in the previous time, which illustrates the ability of making use of the decisions made in similar previous scenarios.

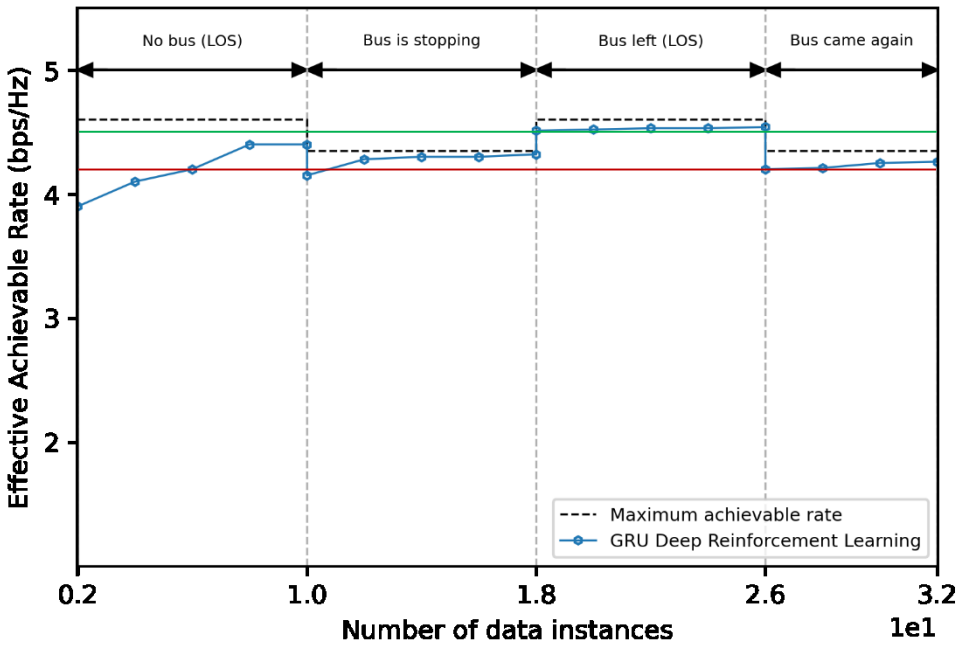


Figure 5.4: Illustration of the ability of the proposed method to make use of previous knowledge to improve communications.

6. CONCLUSION

With the rapidly growing number of devices accessing cellular mobile networks and the increasing demand on bandwidth by each of these devices, the existing generation of these networks has not been able to satisfy these requirements. For the next generation of networks, the range of the frequency must be extended beyond the mmWave range, which imposes the need to handle the challenges presented by the lack of ability of these waves to travel through obstacles, such as walls and trees. Combined with the use of MIMO technology and the significantly higher number of antennas to handle the larger number of the clients and reduce the loading per each antenna, beamforming is being used to direct the wave via the best route to reach the client.

With the high complexity required to make the appropriate decision on which antenna to use to establish communications with the client, recent techniques have resorted to the use of different AI methods. However, the use of classification and regression approaches does not allow the methods to adapt to changes in the environment, unless new training data are collected and used to train the ML method. Hence, a new method is proposed in this study to handle beamforming in 5G networks using RL. The use of this approach allows the proposed method to reevaluate its decisions regarding the selected antennas. This method measures the actual bandwidth achieved by the selected antenna and use the measured value to update the neural networks that is used to predict the bandwidth of each antenna in the environment.

The proposed method has been able to improve the bandwidth of the network by improving the bandwidth provided to the client, compared to existing method that relies on regression approach for the training of the neural network. Additionally, several types of neural networks are evaluated in this study, in which the results show that the GRU has achieved the best performance, with the highest bandwidth for the client and the least execution time per each prediction. Moreover, the results show that the proposed method has been able to reach higher bandwidths faster than existing methods, according to the antenna selection method proposed in this study. This method allows the beamforming coordination to select competitive antennas that have competitive bandwidths and update their actual bandwidth, which allows more accurate decision making.

In future work, the ability of the proposed method to handle an enormous number of clients is going to be evaluated. Despite the ability of RL to handle such complex decisions, the interference among the beams can affect the bandwidth of each client. Hence, this evaluation must be conducted and compared to the results of the experiments conducted in this study, in order to evaluate the ability of applying the proposed method is such environments.

REFERENCES

[1] M. Giordani, M. Mezzavilla, and M. Zorzi, "Initial access in 5G mmWave cellular networks," *IEEE Communications Magazine,* vol. 54, pp. 40-47, 2016.

[2] S. A. Busari, S. Mumtaz, S. Al-Rubaye, and J. Rodriguez, "5G millimeter-wave mobile broadband: Performance and challenges," *IEEE Communications Magazine,* vol. 56, pp. 137-143, 2018.

[3] X. Liu, Q. Zhang, W. Chen, H. Feng, L. Chen, F. M. Ghannouchi*, et al.*, "Beam-oriented digital predistortion for 5G massive MIMO hybrid beamforming transmitters," *IEEE Transactions on Microwave Theory and Techniques,* vol. 66, pp. 3419-3432, 2018.

[4] A. Nordrum and K. Clark, "Everything you need to know about 5G," *IEEE Spectrum,* vol. 27, 2017.

[5] B. Yang, Z. Yu, J. Lan, R. Zhang, J. Zhou, and W. Hong, "Digital beamforming-based massive MIMO transceiver for 5G millimeter-wave communications," *IEEE Transactions on Microwave Theory and Techniques,* vol. 66, pp. 3403-3418, 2018.

[6] T. Maksymyuk, J. Gazda, O. Yaremko, and D. Nevinskiy, "Deep learning based massive MIMO beamforming for 5G mobile network," in *2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS)*, 2018, pp. 241-244.

[7] M. L. Memon, M. K. Maheshwari, N. Saxena, A. Roy, and D. R. Shin, "Artificial intelligence-based discontinuous reception for energy saving in 5G networks," *Electronics,* vol. 8, p. 778, 2019.

[8] C. Liu and H. J. Helgert, "An Improved Adaptive Beamforming-based Machine Learning Method for Positioning in Massive MIMO Systems," *International Journal On Advances in Internet Technology,* vol. 6, pp. 1-12, 2020.

[9] F. B. Mismar, B. L. Evans, and A. Alkhateeb, "Deep reinforcement learning for 5g networks: Joint beamforming, power control, and interference coordination," *IEEE Transactions on Communications,* vol. 68, pp. 1581-1592, 2019.

[10] J. Fu, K. Luo, and S. Levine, "Learning robust rewards with adversarial inverse reinforcement learning," *arXiv preprint arXiv:1710.11248,* 2017.

[11] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine,* vol. 34, pp. 26-38, 2017.

[12] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866,* 2017.

[13] A. Alkhateeb, S. Alex, P. Varkey, Y. Li, Q. Qu, and D. Tujkovic, "Deep learning coordinated beamforming for highly-mobile millimeter wave systems," *IEEE Access,* vol. 6, pp. 37328-37348, 2018.

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

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

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

[17] 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.

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

[19] 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.

[20] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Advances in Neural Information Processing Systems*, 2015, pp. 1117-1125.

[21] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in neural information processing systems*, 2014, pp. 2924-2932.

[22] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning*, 2013, pp. 1058-1066.

[23] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853,* 2015.

[24] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence,* vol. 39, pp. 1137-1149, 2017.

[25] M. Turčaník and M. Javurek, "Hash function generation by neural network," in *2016 New Trends in Signal Processing (NTSP)*, 2016, pp. 1-5.

[26] M. Turčaník, "Hash function generation based on neural networks and chaotic maps," in *2017 Communication and Information Technologies (KIT)*, 2017, pp. 1-5.

[27] N. Abdoun, S. El Assad, R. Assaf, O. Déforges, M. Khalil, and S. Belghith, "Design and implementation of robust Keyed Hash functions based on Chaotic Neural Network," 2018.

[28] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," *International Journal of Artificial Intelligence and Expert Systems,* vol. 1, pp. 111-122, 2011.

[29] D. Maclaurin, D. Duvenaud, and R. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *International Conference on Machine Learning*, 2015, pp. 2113-2122.

[30] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience,* vol. 10, p. 508, 2016.

[31] K. B. Nahato, K. N. Harichandran, and K. Arputharaj, "Knowledge mining from clinical datasets using rough sets and backpropagation neural network," *Computational and mathematical methods in medicine,* vol. 2015, 2015.

[32] B. Kayalibay, G. Jensen, and P. van der Smagt, "CNN-based segmentation of medical imaging data," *arXiv preprint arXiv:1701.03056,* 2017.

[33] Y. Lu, S.-C. Zhu, and Y. N. Wu, "Learning frame models using cnn filters," *arXiv preprint arXiv:1509.08379,* 2015.

[34] G. Tolias, R. Sicre, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," *arXiv preprint arXiv:1511.05879,* 2015.

[35] J. Xu, W. Peng, T. Guanhua, X. Bo, Z. Jun, W. Fangyuan*, et al.*, "Short text clustering via convolutional neural networks," 2015.

[36] D. Chen, J. Bolton, and C. D. Manning, "A thorough examination of the cnn/daily mail reading comprehension task," *arXiv preprint arXiv:1606.02858,* 2016.

[37] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329,* 2014.

[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104-3112.

[39] P. Liu, X. Qiu, and X. Huang, "Recurrent neural network for text classification with multi-task learning," *arXiv preprint arXiv:1605.05101,* 2016.

[40] A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer, G. Neubig, and N. A. Smith, "What do recurrent neural network grammars learn about syntax?," *arXiv preprint arXiv:1611.05774,* 2016.

[41] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[42] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," 1999.

[43] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, *et al.*, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078,* 2014.

[44] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, 2016, pp. 324-328.

[45] B. Athiwaratkun and J. W. Stokes, "Malware classification with LSTM and GRU language models and a character-level CNN," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 2482-2486.

[46] R. Dey and F. M. Salemt, "Gate-variants of gated recurrent unit (GRU) neural networks," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 1597-1600.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* vol. 15, pp. 1929-1958, 2014.

[48] I. K. Sethi and A. K. Jain, *Artificial neural networks and statistical pattern recognition: old and new connections* vol. 11: Elsevier, 2014.

[49] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, ed: Elsevier, 1992, pp. 65-93.

[50] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon*, et al.*, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*, ed: Elsevier, 2019, pp. 293-312.

[51] M. Cilimkovic, "Neural networks and back propagation algorithm," *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin,* vol. 15, 2015.

[52] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature,* vol. 521, p. 436, 2015.

[53] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks,* vol. 61, pp. 85-117, 2015.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] 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.

[63] S. A. A. Alkhateeb, P. Varkey, Y. Li, Q. Qu and D. Tujkovic. (2018). Available: https://www.deepmimo.net/

[64] M. F. Sanner, "Python: a programming language for software integration and development," *J Mol Graph Model,* vol. 17, pp. 57-61, 1999.

[65] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean*, et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265-283.