JUNAID GIRKAR
60004190057
TE COMPS A4

EXPERIMENT - 6
CASE STUDY

Paper Link: https://ieeexplore.ieee.org/abstract/document/9325622

Introduction:
This paper is a comparative study for deep reinforcement learning with CNN, RNN, and LSTM in autonomous navigation. For the comparison, a PyGame simulator has been used with the final goal that the representative will learn to move without hitting four different fixed obstacles. Autonomous vehicle movements were simulated in the training environment and the conclusion drawn was that the LSTM model was better than the others.

Approach:
The research is wholly based on reinforcement learning which is a machine learning training method based on rewarding desired behaviors and/or punishing undesired ones. This method assigns positive values to the desired actions to encourage the agent and negative values to undesired behaviors. This programs the agent to seek long-term and maximum overall reward to achieve an optimal solution.

These long-term goals help prevent the agent from stalling on lesser goals. With time, the agent learns to avoid the negative and seek the positive. This learning method has been adopted in artificial intelligence (AI) as a way of directing unsupervised machine learning through rewards and penalties.

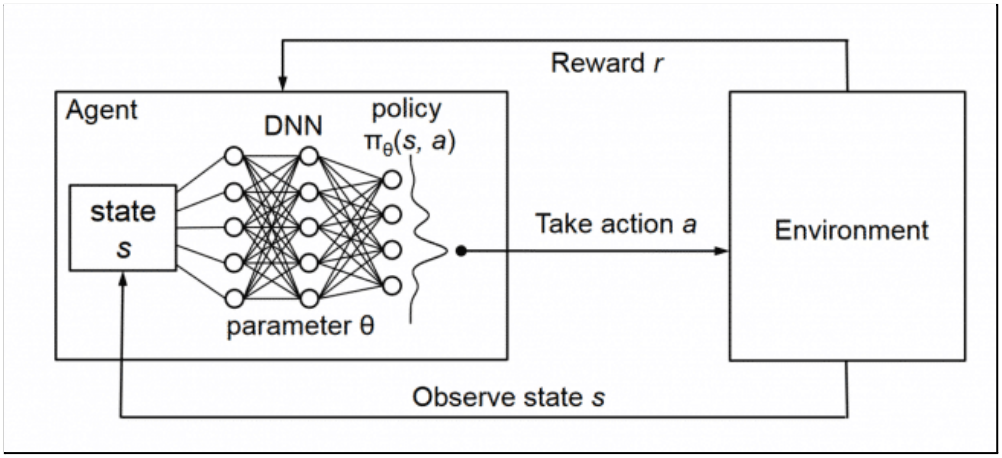
The main advantage of reinforcement learning in this scenario is that unlike deep learning (DL) algorithms, it does not require a data set during
the training phase, increasing its popularity and making it more suitable.

The PyGame simulator interface consists of an agent that learns to move without hitting 4 different randomly positioned obstacles and edges limiting the area. In addition, the paper presents a model-free, off policy approach in this study.

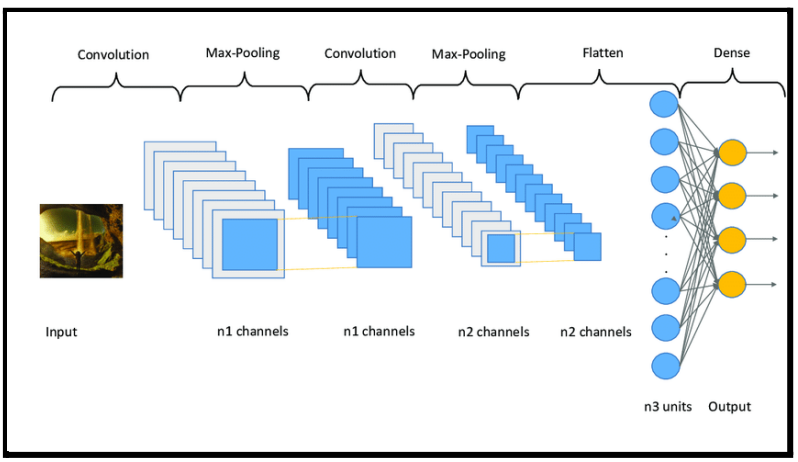
During the research, 4 algorithms were compared. They are:

1) **Deep Q-Network**:It trains on inputs that represent active players in areas or other experienced samples and learns to match those data with desired

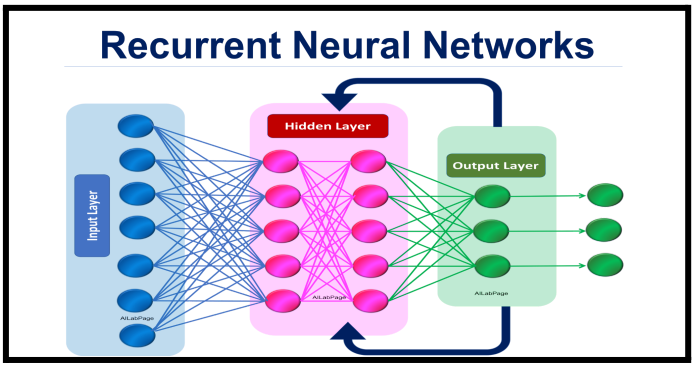
outputs. This is a powerful method in the development of artificial intelligence that can play games like chess at a high level, or carry out other high-level cognitive activities – the Atari or chess video game playing example is also a good example of how AI uses the types of interfaces that were traditionally used by human agents.



2) **CNN:** A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.
The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.
The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.
The main purpose of the convolution process is to extract the feature map from the input data.

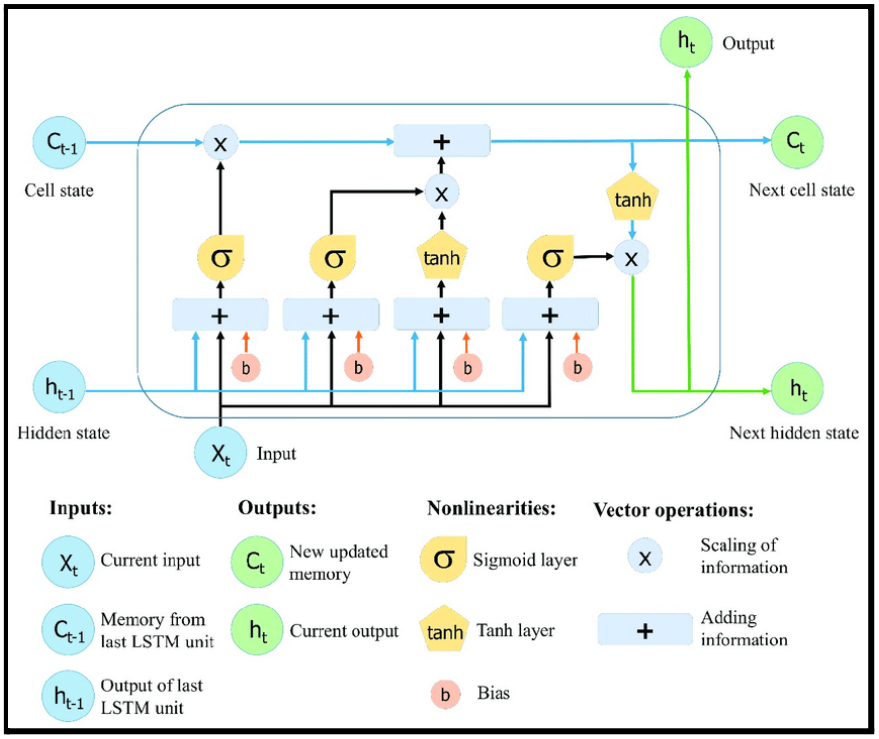


3) **RNN:** Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other, but in cases like when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is the Hidden state, which stores some information about a sequence.



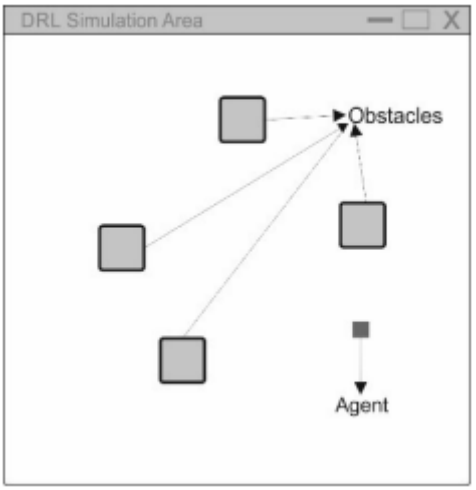
4) **LSTM:** Long Short Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step. LSTM was designed by Hochreiter & Schmidhuber. It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information. As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying on the basis of time-series data. LSTM has a chain structure that contains four neural networks and different memory blocks called cells. Information is retained by the cells and the memory manipulations are done by the gates. There are three gates – Forget gate, Input gate and the output gate.

With LSTMs, there is no need to keep a finite number of states from beforehand as required in the hidden Markov model (HMM). LSTMs provide us with a large range of parameters such as learning rates, and input and output biases. Hence, no need for fine adjustments. The complexity to update each weight is reduced to O(1) with LSTMs, similar to that of Back Propagation Through Time (BPTT), which is an advantage.



SIMULATION:

In this work, PyGame library was used as a robot simulation environment. 4 different obstacles were placed randomly in a 360 * 360 pixel area and the agent was allowed to float within the specified area without hitting these obstacles as shown below.



The agent loses -150 points when hitting obstacles during the learning phase and -50 points when it hits walls. It gets +2 points for every step where it does not hit walls and obstacles.

Four different actions in this simulation are shown in the table below.

Num.	Action
0	go to the left
1	go to the right
2	go to the up
3	go to the bottom

For the model training, python was used as the software language and Keras, a deep learning library was used to create the neural networks. Mean Squared Error was used as the Loss function. Linear was preferred as the activation function. Sigmoid function is used as the activation function in the output layer. The training took 5 hours for CNN, 18 hours for RNN and 35 hours for LSTM on a standard equipped (core i5 Processor and 8GB RAM) computer

CONCLUSION:

Multiple deep learning algorithms were separately tested on the PyGame simulation interface and the conclusions were drawn. The first conclusion was that even though deep reinforcement learning (DRL) models provide fast and safe solutions for autonomous vehicles, their training time is very high. After training it was observed that RNN and LSTM,which are generally used to solve language processing problems, can also be successful in such autonomous navigation problems. The second conclusion was that while the LSTM model took the maximum time to train, it showed the highest success in the success-episode graphics. The paper then concludes with saying that this is a very rich field in terms of future research prospects.

Comparative Study for Deep Reinforcement Learning with CNN, RNN, and LSTM in Autonomous Navigation

1st Ziya TAN
Erzincan Binali Yıldırım University
Erzincan, Turkey
ziyatan@erzincan.edu.tr

2nd Mehmet KARAKOSE
Firat University
Elazığ,Turkey
mkarakose@gmail.com

2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI) | 978-1-7281-9675-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICDABI51230.2020.9325622

***Abstract*— Reinforcement learning algorithms are one of the popular machine learning methods in recent years. Unlike deep learning (DL) algorithms, it does not require a data set during the training phase, increasing its popularity. Today, it offers successful results especially in the navigation of autonomous robots and in solving complex problems such as video games. The feedback process is also known as a reward or called a penalty. Given Agents and environment, it is determined which action to take. In this article, the performance of three different DL algorithms has been compared using the PyGame simulator. In the simulator created using CNN, RNN and LSTM deep learning algorithms, it is aimed that the representative will learn to move without hitting four different fixed obstacles. While creating the training environment, the movement of an autonomous robot in the field without getting stuck in obstacles was simulated. Separate results of each algorithm were reported in the training results. As a result of these reports, it has been observed that the LSTM algorithm is more successful than the others.**

Keywords—Deep Reinforcement Learning, Autonomous Navigations, Deep Q-Learning, PyGame

I. INTRODUCTION

Artificial Intelligence [1] has an important role in making the agent autonomous. It allows the agent to decide his actions in a setting. Artificial intelligence has started to be applied in many areas from image processing to the health sector. One of the sub-branches of artificial intelligence is machine learning. Machine learning [2] has three different methods. Supervised learning is unsupervised learning and reinforcement learning [3]. Supervised learning is a learning method that is generally used by using a labeled data set to solve classification and regression problems. Unsupervised learning is also a learning method that predicts outcome using unlabeled data. Reinforcement learning, on the other hand, is a learning model that benefits from the reward system obtained from the interaction of the agent with the environment. In reinforcement learning, the agent needs to interact with the environment to maximize his reward [4].

Today, mobile robots are now in every aspect of our lives. Especially with the development of industry4.0, the need of factories for robot workers has increased [5]. Robots in factories are not very risky in terms of security as they work in fixed environments, but mobile robots in crowded and narrow environments where people in airports or shopping malls are at risk. This situation can sometimes result in unwanted accidents. Therefore, it should be ensured that mobile robots move safely at the maximum level in mobile environments. The level of security is directly proportional to how well the robots are trained.

Nowadays, deep learning and reinforcement learning have achieved many successes. Deep learning and reinforcement learning were used together in areas such as games in which pixels are learned, agent movements, computer vision, autonomous robots, health, finance and industry [6].

Deep reinforced learning (DRL) has peaked in popularity, especially with developments in the last 6 years. Impressive results have been obtained in studies that combine artificial neural networks and Reinforcement learning after 2010. The most important of these is the DeepMind team's development of an RL-agent that can play the Atari game. Using his points from the game as a reward, the agent was able to choose his next move. This approach is known as Deep Q-Network (DQN) [4]. The second successful work in the same years was AlphaGo [7] . They developed a DRL agent that can beat the world champion in the game of Go.

One of the challenges of reinforcement learning is to strike a balance between exploration and exploitation. The agent must explore the environment sufficiently to learn fully and then make a decision by making use of his experience. In off-policy algorithms such as DQN, the agent uses a value between [0,1] and in random action selection. In the following episodes, the epsilon value is reduced slightly, allowing it to benefit from experience rather than random selection [8].

Recurrent neural networks [9] (RNN) are an artificial neural network model in which connections between nodes form a directed loop. In RNNs, the main purpose is to use sequential information. Long Short Term Memory [10] (LSTM) has shown successful results with RNN in many NLP tasks. In this study, RNN and LSTM were tested not in conventional language processing problems, but in a different problem.

We use PyGame [11] simulator, a python library, to train our agent. PyGame offers a useful interface preferred by many game developers. In the interface created using PyGame, our agent learns to move without hitting 4 different randomly positioned obstacles and edges limiting the area. In addition, paper presents a model-free, off policy approach in this study.

In this study, different DRL algorithms were trained and the results were compared to address the robot navigation problem, especially in rough terrain. As a result of the trainings, Episode-Reward chart and average reward values are shown. When the results are examined, it is seen that the LSTM architecture is more successful.

The structure of the article is organized as follows: In Section II, the algorithms and DRL approach used in our

study are explained, in section III the results of the simulation are stated. Finally, the result is given in section IV.

II. PROPOSED APPROACH

In this section, the algorithms and DRL approach used in our study are explained.

A. Deep Q-Network

Deep Q-Network(DQN) is the first algorithm that uses the image as input and outputs an action [12]. It uses CNN to process the image in the first step and then calculates Q (s, a) for possible actions. The main purpose of an agent is to collect rewards from the environment he is in and to maximize his score.

The DQN agent learns a reward maximization strategy using DL. The DQN agent uses Q-learning [13], a modeless RL technique used to select an optimal policy in a Markov decision making process. Q function is the future weighted evaluation of the system with a fixed policy system. The architecture of DQN is shown fig.1.

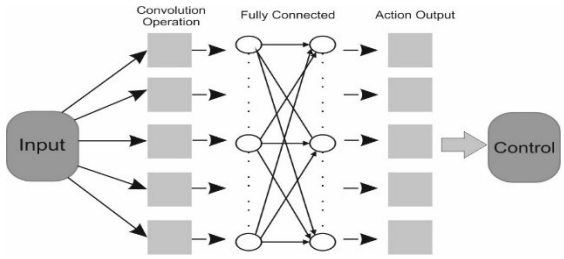


Fig. 1. The Deep Q-Network architecture

In conjunction with DQN-Agent, a variant of Q-learning, also called deep Q-learning, is used. Function Q is defined as the expected future cumulative weighted reward as shown in formula 1:

$$Q(s,a) = r + \gamma maxQ'(s',a') \quad (1)$$

TABLE 1. NOTATIONS

Notation	Meaning
NewQ(s,a)	New Value
Q(s,a)	Current Q Value
α	Learning Rate
R(s,a)	Reward For Taking That Action At That State
γ =	Discount Rate
Max Q'(s',a')	Maximum Expected Future Reward
State (s)	The Parameter That Defines The Current Region
a	Current Action
a'	Next Action
s'	Next State

The flow chart of the Deep Q-Learning [14] algorithm is shown in fig. 2.

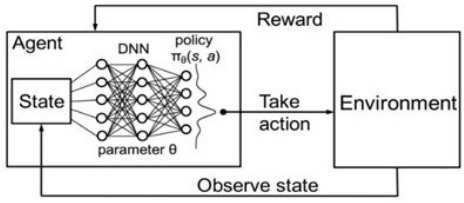


Fig. 2. The Deep Q-Learning algorithm process

The agent updates the variables of the model every iteration. The pseudo code of the Deep Q-Learning algorithm used at this stage is shown in table 2.

TABLE 2. PSEUDO CODE OF DEEP Q-LEARNING ALGORITHM

1	Initialize Replay Memory capacity
2	Initialize network with random weight
3	for each episode:
	-Initialize the starting state
	-For each time step
	-Select an action
	-Via exploration or Exploitation
	-Execute selected action in an emulator
	-Observe reward and next state
	-Store experience and replay memory
	-Update network weights

B. CNN

The main purpose of the convolution process is to extract the feature map from the input data [15].

At this stage, the symmetry of the filter to be applied to the two-dimensional data with respect to the x and y axis is taken. While the filter used is hovered over the input image depending on the step length, the overlapping values at each step are multiplied one by one and the sum of all values is recorded as the value of the output matrix. The simplicity of the process varies according to the number and format of the input data.

CNNs are basically composed of these layers. Convolution layer, pooling layer and fully connected layer.

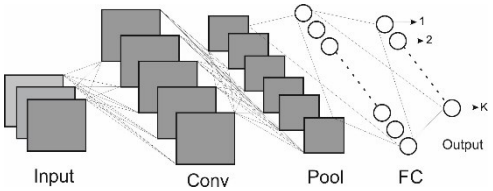


Fig. 3. A classic CNN (Convolutional Neural Network) architecture

A classic CNN (Convolutional Neural Network) architecture is shown in fig. 3.

C. RNN

The RNN (Recurrent Neural Network) neural network model is structurally different from other artificial neural networks. One of the most important differences is that it has a temporary memory. This memory has short memory. Therefore, they cannot remember very far, they can remember in the recent past *[16]*.

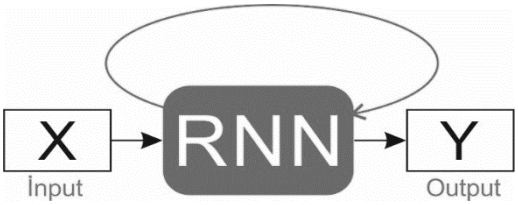


Fig. 4. RNN structure network

The fig.4 shows the structure of an RNN network. Its specific feature is that the output from the previous step is the input data for the current step. Networks that remember information are included in the hidden layer. Thanks to the structure in this layer, repetition takes place. Today, this structure is frequently used in natural language processing, translation operations and data mining.

D. Long Short Term Memory (LSTM)

LSTM was developed by Sepp Hochreiter and Juergen Schmidhuber in 1997 to find a solution to the vanishing gradient problem [17]. LSTM has reached a wide usage area with the contribution of many people along with the studies carried out later. Fig. 5. shows the LSTM architecture.

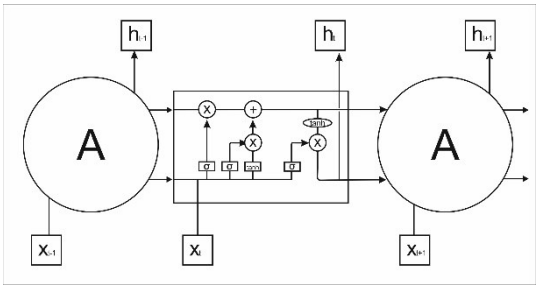


Fig. 5. The LSTM architecture

E. Exploration or Exploitation

The agent will follow the path of discovery in the first stage. Because there is an environment, he does not know. It will benefit from its experience as it does enough exploration in the later stages. The parameter Ɛ –Greedy plays an important role in determining these experiences.

F. Ɛ -Greedy

In this study, Ɛ –Greedy was determined as the behavior policy. The value of was initially set at 1. This value will decrease at the rate determined in the following steps and determine the action of the agent.

$$a = \begin{cases} a, 1-\varepsilon \\ random\ action, \varepsilon \end{cases} \quad (2)$$

The Ɛ in (2) indicates the probability of exploration of the agent in the learning phase.

G. The Agent-Environment Interaction

DQL is a closed loop learning method responsible for learning a task determined by an agent's trial and error method [18]. It is not clear what to do during the learning phase, instead, he learns what to do from his own experience. At each step the current situation of the environment is observed and chooses an action according to the policy. As a result of the selected action, the awards received until the end of the episode are collected. The figure shows the interaction of an agent with the environment.

Each reward collected is the learning criterion for the agent performing the actions. In the collection of the awards, the representative acts according to the policy (π). Therefore, when formulating the RL problem, we must define a policy that brings the maximum reward from the environment. Reinforcement learning works based on Markov Decision Process (MDP). MDP states that the current state (s) depends only on the last state (s') and does not depend on all past situations [19].

The basic philosophy of reinforcement learning algorithms is to organize a reward or punishment of an action in a method to ensure that a desired outcome occurs [20]. Fig. 6 shows the interaction of an agent with the environment.

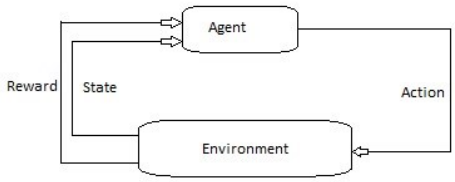


Fig. 6. An RL agent's interaction with the environment

Actions (a) = The sequence of actions the agent takes at each step. Generally these actions are defined as forward, right, left and backward, but in some cases they can also move linearly and angularly.

State (s): It is defined as the location of an agent in the environment.

Reward (R): The reward function describes how well the representative learns from his interaction with the environment. It allows him to move efficiently and safely in his next steps.

Environment: It is defined as the area where the representative learns his / her duty. It performs an action within this area, receives a reward for that action, and determines the next step. There are two types of environments. In a deterministic environment, the reward function and transition model are deterministic, if the representative acts with repetitive actions, he receives the same reward. In the next step, there is uncertainty regarding the effect of the action in the stochastic environment.

Policy (π): Policy determines an agent's behavior. Provides information for a representative to take a specific action in the current situation. The agent's goal is to choose the policy in which he can get the maximum reward [21].

Discount factor (γ): Used in the agent's reward function. The purpose of the discount factor focuses on instant rewards.

The discount factor is used to make the reward not forever and limited.

III. SIMULATION RESULTS

A. Reinforcement Learning for the PyGame

Pygame is a Python library built on the Simple Direct Media Layer (SDL) library to prepare interactive games in the Python programming language.

In this work, we used the PyGame library as a robot simulation environment. 4 different obstacles were placed randomly in a 360 * 360 pixel area and the agent was allowed to float within the specified area without hitting these obstacles.

As shown in the fig. 7, four different obstacles are placed and the agent learns to move without hitting the obstacles.

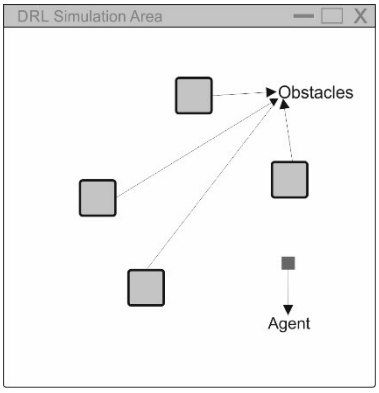


Fig. 7. PyGame simulation environment

The agent loses -150 points when hitting obstacles during the learning phase and -50 points when it hits walls. It gets +2 points for every step where it does not hit walls and obstacles.

Four different actions in in this simulation are shown in table 3.

TABLE 3. ACTIONS OF ENVIRONMENT

Num.	Action
0	go to the left
1	go to the right
2	go to the up
3	go to the bottom

B. Training

Python software language is used in this study. Keras, a deep learning library, was used to create artificial neural networks. In addition, in our study, LSTM, RNN and CNN Deep Learning algorithms were programmed separately and the results were reported.

The artificial neural network used in the Deep Q-Learning algorithm consists of 48 single-layer neurons. Mean Squared Error [22] was used as the Loss function. Linear [23] was preferred as the activation function. Sigmoid function [24] is used as the activation function in the output layer. The training took 5 hours for CNN, 18 hours for RNN and 35 hours for LSTM on a standard equipped (core i5 Processor and 8GB RAM) computer.

The parameters used in the training phase of the Deep Q-Learning algorithm are shown in Table 4.

TABLE 4. PARAMETER OF DEEP Q-LEARNING

Parameter	Value	
Episode	200	
Learning Rate	0,01	
Gamma	0,95	
Exploration Rate	Exploration Max	1
	Exploration Min	0.1
	Decay Rate	0.95

The following reward-episode graphics were formed as a result of the training.

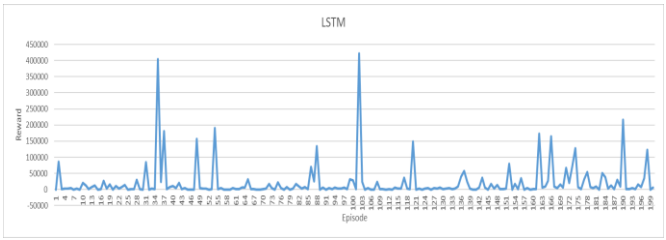


Fig. 8. Result of training LSTM algorithm

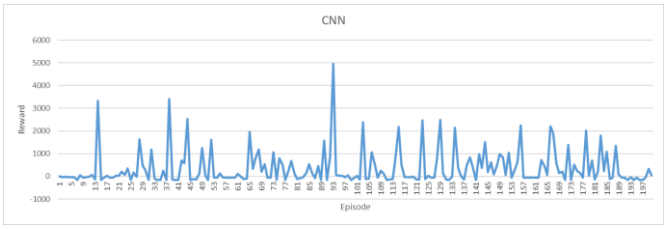


Fig. 9. Result of training CNN algorithm

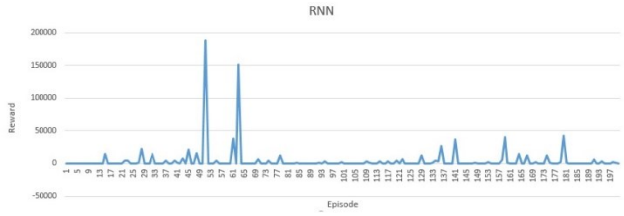


Fig. 10. Result of training RNN algorithm

When the reward-episode graphics obtained as a result of the training are examined; The results of the LSTM algorithm shown in fig. 8 were found to be more successful than the other two algorithms. In the CNN algorithm shown in Fig. 9, it was seen that he collected approximately 5000 reward points during 200 episodes. In Fig. 10, it is revealed that the RNN algorithm is more successful than the CNN algorithm.

When Table 5 is examined, we can see the results of reward averages in parallel with the reward-episode graphs.

TABLE 5. AVERAGE OF AWARDS RECEIVED

Name of Algortihm	Reward Mean
CNN	363
LSTM	21.555
RNN	3.922

IV. Conclusion

This article provides an insight into the results of deep reinforcement learning and deep learning algorithms applied to problems involving the motion control of autonomous robots in disabled terrains. In particular, the performance of our agent in the PyGame interface was evaluated using popular deep learning algorithms such as CNN, RNN and LSTM separately.

DRL can provide fast and safe solutions for autonomous vehicles. However, some disadvantages of deep reinforcement learning should be taken into account. First, training the agent in a simulator can take a lot of time for a particular problem, the other depends on the agent's function with the reward.

As a result, all three deep learning algorithms used were trained in 200 steps and the results were observed. In the results of the training, it was observed that RNN and LSTM, which are generally used to solve language processing problems, can also be successful in such problems. It was determined that the LSTM deep learning algorithm completed its training in a longer time compared to the other two algorithms. In the next stages, different results can be obtained with different simulators and different deep learning models.

In summary, DRL, the topic that attracted the attention of researchers, how to increase its success in autonomous robotic systems will maintain its popularity in the near future. Studies on this subject are followed with curiosity by many people.

References

[1] S. Haykin, "Neural networks: A comprehensive foundation," *network,* 2004.

[2] N. Goyal, J. K. Sandhu and L. Verma, "Machine Learning based Data Agglomeration in Underwater Wireless Sensor Networks," *International Journal of Management, Technology And Engineering,* vol. 6, no. 9, pp. 240-245, 2019.

[3] Z. Tan, *Derin Öğrenme Yardımıyla Araç Sınıflandırma,* Elazığ: Fırat Üniversitesi-Fen Bilimleri Enstitüsü, 2019.

[4] M. M. Ejaz, T. B. Tang and C.-K. Lu, "Autonomous Visual Navigation using Deep Reinforcement Learning: An Overview," *IEEE Student Conference on Research and Development,* 2019.

[5] F. Zhang, J. Leitner, M. Milford, B. Upcroft and P. Corke, "Towards Vision-Based Deep Reinforcement Learning for Robotic Motion Control," in *In Proceedings of the Australasian Conference on Robotics and Automation* , 2015.

[6] K. Zhang, F. Niroui, M. Ficocelli and G. Nejat, "Robot Navigation of Environments with Unknown Rough Terrain Using Deep Reinforcement Learning," *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR),* pp. 1-7, 2018.

[7] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre and V. Den, "Mastering the game of Go with deep neural networks and tree search," *nature,* vol. 529, p. 484, 2016.

[8] R. Güldenring, "Applying Deep Reinforcement Learning in the Navigation of Mobile Robots in Static and Dynamic Environments," 2019.

[9] A. M. Schäfer, Reinforcement learning with recurrent neural networks, Osnabruck, 2008.

[10] Z. Qun, L. Xu and G. Zhang, "LSTM Neural Network with Emotional Analysis for Prediction of Stock Price," *Engineering letters,* vol. 25, no. 2, 2017.

[11] «Pygame,» [Çevrimiçi]. Available: www.pygame.org. [Erişildi: 10 Ekim 2020].

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg and D. Hassabis, "Human-level control through deep reinforcement learning," *nature,* pp. 529-533, 2015.

[13] M. Hüttenrauch, A. Šošic and G. Neumann, "Deep Reinforcement Learning for Swarm Systems," *Journal of Machine Learning Research,* pp. 1-31, 2019.

[14] J. Fan, Z. Wang, Y. Xie and Z. Yang, "A Theoretical Analysis of Deep Q-Learning," *2nd Annual Conference on Learning for Dynamics and Control,* vol. 120, pp. 1-4, 2020.

[15] P. Li, M. A. Aty and J. Yuan, "Real-time crash risk prediction on arterials based on LSTM-CNN," *Accident Analysis & Prevention,* 2020.

[16] X. Li, L. Li, J. Gao, X. He, J. Chen, L. Deng and J. He, "Recurrent Reinforcement Learning: A Hybrid Approach," *arXiv:1509.0344,* 2015.

[17] F. Rundo, "Deep LSTM with Reinforcement Learning Layer for Financial Trend Prediction in FX High Frequency Trading Systems," *Applied Sciences,* vol. 20, no. 9, p. 4460, 2019.

[18] S. Ha, J. Kim and K. Yamane, "Automated Deep Reinforcement Learning Environment for Hardware of a Modular Legged Robot," *15th International Conference on Ubiquitous Robots,* pp. 348-354, 2018.

[19] A. Ramaswamy, "Theory of Deep Q-Learning: A Dynamical Systems Perspective," *arXiv:2008.10870v1,* 2020.

[20] R. S. Sutton and A. G. Barto, Reinforcement Learning:An Introduction, London: MIT Press, 2015.

[21] S. Bhagat, H. Banerjee, Z. T. H. Tse and H. Ren, "Deep Reinforcement Learning for Soft, Flexible Robots:Brief Review with Impending Challenges," *Robotics,* 2019.

[22] "Mean_squared_error," [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_error. [Accessed 10 8 2020].

[23] P. Jain, "Complete Guide of Activation Functions," [Online]. Available: https://towardsdatascience.com/complete-guide-of-activation-functions-34076e95d044. [Accessed 10 8 2020].

[24] P. Ramachandran, B. Zoph and Q. V. Le, "Swish: A Self-Gated Activation Function," *arXiv:1710.05941v1 [cs.NE],* 2017.