iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VI **Month of publication:** June 2021

DOI: https://doi.org/10.22214/ijraset.2021.35753

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VI Jun 2021- Available at www.ijraset.com

Self-Driving Car using Deep-Q-Networks

Rajath C V[1], Yugant Soni[2], Santhosh B[3]
[1, 2, 3]*Department of Telecommunication Engineering, Dayananda Sagar College of Engineering* [1]rajath2608@gmail.com

Abstract: ***A autonomous car is also called a self-driving car or a robot car. As for the history of self-driving cars, radio technology was used to control the tests, which began in 1920, and later in 1950, the tracks were finally put in place. The present-day individual is habituated to automation technology and the use of robotics in areas such as agriculture, medication, transportation, IT industry, etc. In the recent decades, the automotive sector has come to the forefront of researching private car technologies.***
The independent Level-3 standard was out in 2020. Everyday automotive technology researchers solve challenges. The prime intention of the project is to create a self-driving car using Deep-Q-Networks, thus enabling the car to make decisions based on the spontaneously occurring events.
Independent vehicles require data and are regularly updated, therefore IoT and AI can assist in allocating device data to the machine.
Keywords: Autonomous vehicles, Deep-Q-Network, Reinforcement Learning, Markov Decision Process, Bellman Equation

I. INTRODUCTION

Technology has always been striving to make human lives simpler from the very beginning of time. The very objective of technology is to make human lives comfortable, effortless and straightforward along with improving the quality of human lives. Travel or quick and personal transport is one such area which is extremely important to help achieve this goal. Development and enhancement of motor vehicles began from as early as the 17th century and has been constantly improving in multiple areas including comfort, performance and technological expertise. Self-driving cars have always been an area of tremendous interest and engineers have been working on it for many years.
These autonomous or self-driving cars would make technology ever so closer to reaching its ultimate goal that is improved quality and simpler human life.
Self-driving or autonomous cars are those which are adept at recognizing and distinguishing its surrounding and driving carefully and securely with minimal human input or even in the absence of any input from the driver which means that they can successfully transport people or goods from their source to destination safely. The driving agent, which is trained using Deep-Q-Networks along with Reinforcement learning, employs reward and punishment signals to train agents.

II. LITERATURE SURVEY

A. Deep CNN-based Real Time Traffic Light Detector for Self-Driving Vehicles: Zhenchao Ouyang et al [1]

This paper talks about using a model which identifies all potential traffic lights and uses a Convolution Neural Network (CNN) to categorize the results attained. The proposed model was able to achieve a high accuracy of about 99% by incorporating the detector module with on NVidia Jetson TX1/TX2. The paper talks about how the data set consisting of the traffic lights was obtained using Heuristic ROI detector and proposal based detection model, entering the data set into CNN models such as AlexNet/GoogleNet.

B. Toward a Brain-Inspired System: Deep Recurrent Reinforcement Learning for a Simulated Self-Driving Agent: Jieneng Chen et al [2]

This paper talks about methods of simulating the different behaviors of the human brain using the modified DRQN model which is based on the deep-Q-network architecture. It talks about how the driving agent was trained to respond only to visual signals. The project used Reinforcement learning where the actions included left, right, going forward, going backwards and braking, and when the driving agent performs the required task, it sends reward signals, however when it fails to perform the task as expected it sends punishment signals. This paper proposes a deep recurrent reinforcement learning network to resolve problems encountered in the simulation. But this brain inspired model can perform better and provide more stability with the use of trial and error method.

C. Robust Lane Detection Using Multiple Features: Tejas Gupta et al [3]

This paper talks about the use of monocular RGB images as a method for ego-lane detection. It also talks about an ego-lane detection algorithm which executes equally well even if the path is curvy, low lit or filled with obstructions. The goal is to employ numerous low-level visual features in order that the detector doesn't fail when an incorrect result is given by a visual feature. An estimator is utilized for renewing the model after each step so as to not extract wrong or noisy images. Stress also impacts muscles. The proposed method's performance is evaluated on the KITTI dataset. A unique assessment metric is proposed that is comparatively more apt for ascertaining lane detection outcomes.

D. Detecting Unexpected Obstacles for Self-Driving Cars: Fusing Deep Learning and Geometric Modelling: Sebastian Ramos et al [4]

This paper proposes a replacement deep learning-based obstacle detection framework. The paper talks about a type of convolutional network which is employed to forecast the pixel-wise semantic classification of Free-space, unexpected obstacles on the path. The paper talks about the introduction of a deep learning-based approach to spot minute and unpredictable obstructions. The paper demonstrates a way in which CNNs can be used to simplify information present in the training set and thereby solve one of their major setbacks: managing outliers and therefore the "open world". Thus new and earlier unnoticed objects can be effectively managed and an appropriate background class was modelled.

III.METHODOLOGY

A. Software used

1) *Python:* Python is an open source high level language which can be used for web development, data analytics, data visualization, machine learning etc
2) *Unity Simulator:* The simulator used is the Unity Simulator for self-driving cars. Unity is a real time 3D rendering platform used to efficiently create simulation environments. The fundamental elements of the simulation environment are Vehicle dynamics such as friction experienced by the car, Dynamic elements like pedestrians or other cars and Parameters such as weather conditions, trees and so on.

B. Packages used to build the stress detector

1) *NumPy:* Python has a module called NumPy. It is a multidimensional array object library with a set of array processing routines.
2) *Torch:* Torch is python module that provides Tensor computation with stronger GPU capabilities. It includes some of the components such as Torch.nn and Torch.autograd which is implemented
3) *OS:* OS package is used in Python to interact with the operating system.

C. Reinforcement Learning

A field in machine learning which is focused on making an agent take intelligent decisions, in the environment they are placed in, in the best way possible so that the agent can reach a state where it gets the maximum reward as opposed to ending up in a state that gives the agent the least reward. This reward can be positive or negative and depends on the state that the agent is in, and the agent always strives to reach the region of maximum reward and avoid the state of minimum reward at all costs.

D. Markov Decision Process

The Markov Decision Process, in reinforcement learning, is used to deal with State transitional probabilities. A state, in reinforcement learning, refers to the present position of the training agent with respect to its environment. The agent while undergoing the state transition, always has the chance or probability of moving to another direction completely, which is referred to as State Transitional Probability. The Markov Property states that the present state of the object is not related to the previous state of the agent, meaning the current state the agent has reached does not depend on the state the agent was present in, before transitioning to this state.

E. Bellman Equation

The Bellman Equation is a mathematical enhancement or optimization technique. The appropriate Bellman equation can be formed by adding new state variables to the equation. The first decision to be made for using the Bellman equation is the type of objective function required. For the proposed method, the objective function is to maximize the reward obtained by the driving agent. the Bellman equation provides the optimal solution which is basically a rule which decides what action or values the control variables should hold for each state or when the state changes.

F. Deep Q Learning

To help AI agents in operating in environments with distinct action spaces, Deep Q-Learning is used. It uses Q-values to provide the desired results. The Q value is the value of the respective action-state pair. A memory table is created in Q-learning to store Q-values for all conceivable combinations of the state and the related action. This value is fed into the Deep Q Learning Network, and it gives the required action to be taken, combining the reward from the current action-state combination with the greatest Q value at the following state.

IV. IMPLEMENTATION

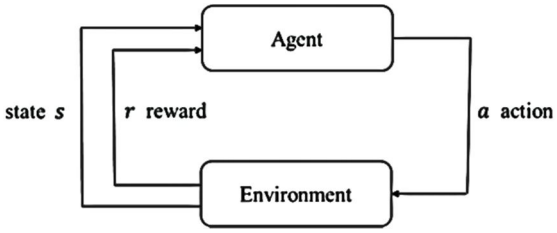


Fig. 1 Deep-Q-Network using Reinforcement Learning

The implementation is performed by training the driving agent by employing Deep-Q-Networks using Reinforcement Learning. The agent performs some actions in the environment, based on which it attains new states and corresponding rewards. In order to show the working of the Deep-Q-Network algorithm, we first keep an empty black screen, which is our map, as seen in Fig.2, and make the driving agent move around the map. The agent moves around the map in order to familiarize itself with its surroundings.

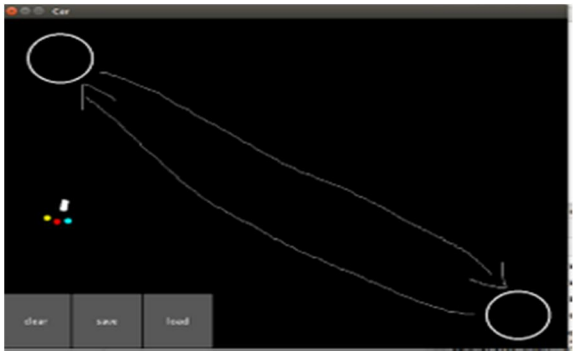


Fig. 2 The empty map for the agent to travel

Once the agent is able to traverse back and forth through the entire map, we begin to draw obstacles around the black screen as illustrated in Fig. 3. Once the agent comes in contact with these obstacles it decides whether it should turn 20 degrees to the left or 20 degrees to the right or continue to not turn at all. In order to make sure that the agent learns from the mistakes it makes we give rewards and penalties. The penalties are -5 whenever the agent runs into an obstacle created by us and -1 if the agent gets further away from the objective. The agent is given a reward of +1 if it gets closer to the objective. We provide five sensors to our agent which include sensor red, yellow and blue, and the other two are positive and negative orientations which tells us about the degree of the agent.

Fig. 3 The map with obstacles to train the agent

In the next step we connect our trained agent into the Unity simulator. In order to connect it to the Unity simulator and make our agent familiar to the environment provided by Unity, we use our three sensors to get the images of our surroundings for every position the car is in. Hence our three sensors provide us images of what is in front of the agent, left of the agent and to the right of the agent. The orientation sensor provides us information about the degree position of the car for each position. Hence the dataset consists of images which are captured by our sensor and also a CSV file which consists of the steering angle, breaking and the speed which is collected by manually driving the car around the car around the Unity simulator.

This data collected is now processed using the Deep-Q network algorithm and a model file is created .Once the model file is created, we have to connect the model file to the simulator and the car will run on its own. In order to connect to the simulator we use Anaconda. It is a tool that is used to run python codes. Anaconda is very effective as it allows us to run python libraries and dependencies with great ease. Once the dependencies and libraries are installed using anaconda, the Unity simulator is started, and the car begins to run autonomously by the trained model.

V. RESULT

Fig. 4 Fully trained agent driving as required

Fig. 4 shows the agent running smoothly in the Unity simulation environment. It is also seen that the agent has been successfully linked to the Unity simulator. The agent successfully performs all the required actions such as following the lanes and avoiding obstacles as it goes forward in the environment. It is able to maneuver through the simulation environment, and thus receive the positive rewards. We can notice that once the dataset has been collected and the Deep-Q-Learning algorithm is applied to it, the agent is able to travel around the environment simultaneously and it can perform behavioral cloning with a high level of accuracy.

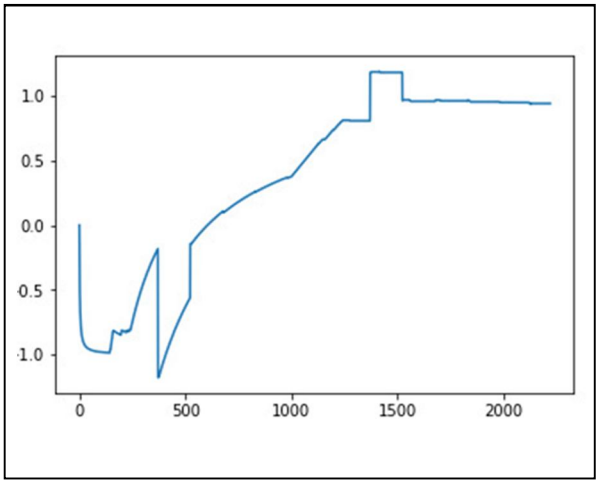


Fig. 6 Graph showing reward obtained for each step

Fig. 6 shows the rewards obtained by the driving agent for every action it performed. It is seen from the graph that the agent receives a reward of +1 whenever it gets closer to the objective and -1 when it moves farther away. The results becomes stable and consistent after 1500 time steps.

VI. CONCLUSION

Every year, around 10 trillion kilometers are being covered using automobiles, and people spend billions of hours behind the wheel, which they could spend on other important things. Millions of people lose their life, every year, due to road accidents caused by human errors. All of this can be eliminated by employing self-driving cars, which saves, not only time, but also lives of millions of people.

The project shows that the agent was successfully trained to drive in the environment created in the Unity simulator. The training was done successfully by using Reinforcement learning and Deep-Q-Networks. Various equations and techniques have been applied, to help the self-driving automobile make better decisions. The agent recognizes the reward and punishment signals given to it during the course of the training. The reward points are collected in a batch, which the agent consults while making a decision. The guiding systems make use of neural networks to mimic the processes of biological decision-making systems.

REFERENCES

[1] Zhenchao Ouyang, Jianwei Niu, Yu Liu, Mohsen Guizani, " Deep CNN-based Real-time Traffic Light Detector for Self-driving Vehicles", IEEE Transactions on Mobile Computing, Volume: 19 , Issue: 2 , Feb. 1 2020

[2] Jieneng Chen, Jingye Chen, Ruiming Zhang and Xiaobin Hu, "Toward a Brain-Inspired System: Deep Recurrent Reinforcement Learning for a Simulated Self-Driving Agent", Front. Neurorobot., 28 June 2019.

[3] Tejus Gupta, Harshit S. Sikchi, Debashish Charkravarty, "Robust Lane Detection Using Multiple Features", 2018 IEEE Intelligent Vehicles Symposium (IV)

[4] Sebastian Ramos ; Stefan Gehrig ; Peter Pinggera ; Uwe Franke ; Carsten Rother, "Detecting Unexpected Obstacles for Self-Driving Cars:Fusing Deep Learning and Geometric Modeling", 2017 IEEE Intelligent Vehicles Symposium (IV)

[5] Wuttichai Vijitkunsawat, Peerasak Chantngarm, "Comparison of Machine Learning Algorithm's on Self-Driving Car Navigation using Nvidia Jetson Nano" 2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)

[6] A N Aneesh, Linu Shine, R Pradeep, V Sajith, "Real-time Traffic Light Detection and Recognition based on Deep RetinaNet for Self Driving Cars" 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)

[7] G. Meera Gandhi, Salvi, "Artificial Intelligence Integrated Blockchain For Training Autonomous Cars" 2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)

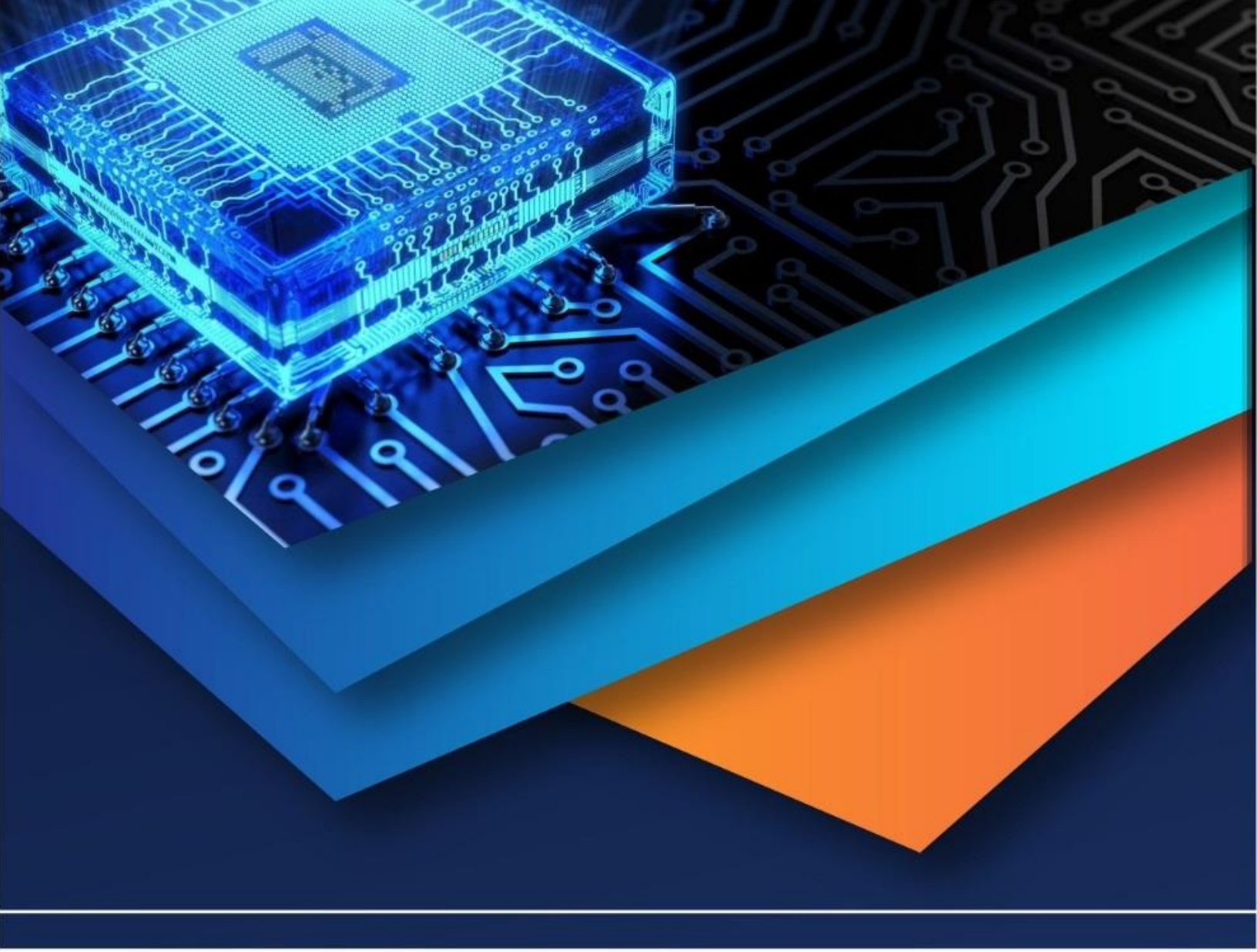
[8] Takafumi Okuyama, Tad Gonsalves, Jaychand Upadhay, "Autonomous Driving System based on Deep Q Learning" 2018 International Conference on Intelligent Autonomous Systems (ICoIAS)

[9] Yang Guan, Shengbo Eben Li, Jingliang Duan, Wenjun Wang, Bo Cheng, "Markov probabilistic decision making of self-driving cars in highway with random traffic flow: a simulation study", Journal of Intelligent and Connected Vehicles, 18 October 2018.

[10] Abdur R. Fayjie, Sabir Hossain, Doukhi Oualid, Deok-Jin Lee, "Driverless Car: Autonomous Driving Using Deep Reinforcement Learning in Urban Environment" 2018 15th International Conference on Ubiquitous Robots (UR)

[11] Michael G. Bechtel, Elise Mcellhiney, Minje Kim, Heechul Yun, "DeepPicar: A Low-cost Deep Neural Network-based Autonomous Car", 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)

[12] Michael Treml , José Arjona-Medina, Thomas Unterthiner, Rupesh Durgesh, Felix Friedmann, Peter Schuberth, Andreas Mayr, Martin Heusel, Markus Hofmarcher, Michael Widrich, Ulrich Bodenhofer, Bernhard Nessler and Sepp Hochreiter, "Speeding up Semantic Segmentation for Autonomous Driving", 29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY