Deep-Q Learning for Autonomous Driving System in Simulation Environment

[1]Udit Gavasane, [2]Amod Sahasrabudhe, [3]Rohan Tergaonkar, [4]Kalyani Shinde, [5]Prof.Mrs. S.S. Dixit

[1]B.E. Student [2]B.E. Student [3]B.E. Student [4]B.E. Student [5]Professor

[1]Department of Information Technology
[1]Pune Vidyarthi Griha's College of Engineering and Technology, Pune, India

Abstract: Due to increase in traffic accidents by negligent driving by human beings, there is a need for safe and comfortable driving experience. For this purpose, development of a simulation for autonomous driving car which later can be developed into a physical model after the simulation training has been successful. This project deals with simulation results of an autonomous car learning to drive on a single-lane highway or a road with lane markings. Learning is performed by using Deep Q algorithm. For a given input image of the street captured by the camera which is installed at the front of the car, the Deep Q Network computes the Q values (rewards) corresponding to the actions available to the autonomous driving car. These actions are discrete angles through which the car can steer for constant speed. The autonomous driving system in the car enforces action that has the highest reward.

***Index Terms* – Artificial Intelligence, AI, Self-driving Cars, Autonomous Driving cars, Reinforcement Learning**

INTRODUCTION

Artificial Intelligence (AI) is a scientific field that has gained a huge popularity, attraction and fascination outside the scientific community as well, because the idea of intelligent machines ignites people's imagination. The whole idea of a machine doing a human's job replicating the human itself is quite interesting. When it comes to performing a hardcore human job like driving, cooking, household chores etc. and relating them with the machine by making them help us do the following activities it becomes quite a task for a developer to such applications. There are various subfields of AI which deal with such complex scenarios namely, Supervised Machine Learning, Unsupervised Machine Learning and Reinforcement Learning. The first two that is Supervised and Unsupervised Machine Learning are currently very popularly used on a large scale. Reinforcement Learning (RL) still is a fast-developing algorithm in the current scenario.

I. EXISTING SYSTEM

Autonomous Driving cars are being implemented on a medium scale in the United States of America, Japan, Canada and Holland. Tesla [1] is one of the leading companies which has proposed the concept of autonomous driving cars with various sensors and high-performance cameras. The simulation environment used by current automobile giants like Tesla use software provided by NVDIA. Google began its development in this field of autonomous driving in 2009. By 2017, Google trained its model, WAYMO [2] for 3 million miles in its 4 different states. Volvo [6] has stepped into the self-driving field since 2006 and it's first car using supervised learning algorithm was out in the market in 2017. Volvo also plans to use unsupervised learning algorithms in developing their cars by 2021. Audi, BMW [7] and Nissan are expecting their concept car of autonomous driving cars in the upcoming decade. NVDIA is also working independently on their project called NVDIA Drive AGX which is a computing platform that serves as a brain for autonomous vehicles. They have published a paper using end to end learning which the authors have described in [5]. The drawback of current autonomous driving system is that it is quite expensive as it relies on specialized infrastructure and equipment like Light Detection and Ranging (LIDAR) and Laser Range Finder (LRF) for detection of the lane

II. PROPOSED SYSTEM

The car will be trained in a simulation environment provided by Donkey [3]. Donkey is a high-level self-driving library in python. It was developed with a focus on enabling fast experimentation and easy contribution. The Donkey Gym project is an open AI gym wrapper around the self-driving sandbox donkey simulator. The reason behind using the simulation environment is to avoid loss of life, saving time and saving money. The camera fitted on the car will provide images which are resized and then converted from RGB to Greyscale. The reinforcement learning algorithm uses these images to perform suitable action in a situation to assist the car stay on track. To train the car to stay on track and for every correct action, a hypothetical reward is assigned. If the agent performs an incorrect action, the episode is restarted. In this manner the car leans to drive on the road. The focus of the study is to enable the car to drive by detecting road features using single front camera vision. The objective of this paper is to make modifications in the existing system to make it cost effective but not compromise with its accuracy. In simple words without using other expensive sensors, making use of only one camera the agent learns to drive on its own with the same accuracy as that of the autonomous driving car with fully loaded sensors.

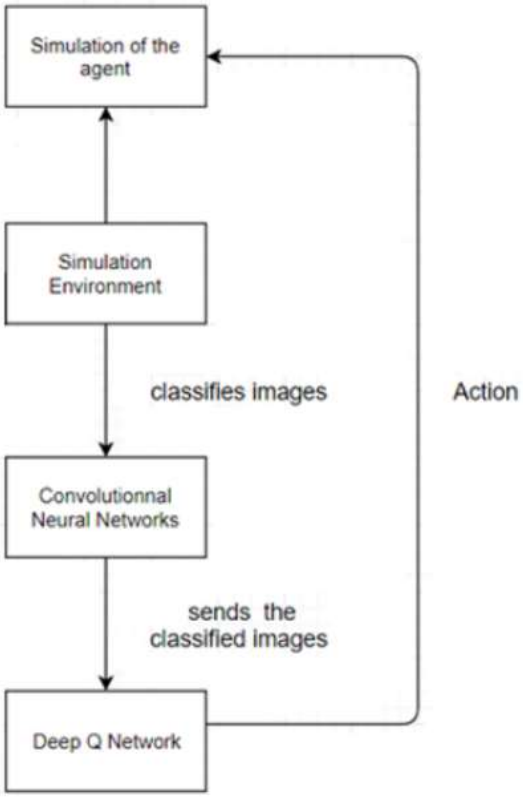


Fig1.Architecture Diagram

III. IMPLEMENTATION

1. Simulation Environment: - Donkey Gym

Donkey Gym project is an open AI gym wrapper around the Self Driving Sandbox donkey simulator. The name donkey came from the hard-working domestic animals namely donkeys because the ultimate aim of the use of donkey library is to build something useful. Donkey is minimalist and modular library for Python. It is developed for hobbyists and students with a focus on allowing fast experimentation and easy community contributions. Donkey Gym is built using Unity [4] software. The unity software is used to create virtual environment for the purpose of simulation. Donkey Gym uses the internal physics and graphics of this Unity Software, and connects to a donkey Python process to use our trained model to control the simulated Donkey. There are many ways to use this donkey gym software. The advantage of this software is to know and use the standard Donkey car drive/train/test cycle by treating it as virtual hardware. The data such as speed, throttle, steering angle etc. can collected. The model can also be driven and trained using commands as if you were using a real car.



Fig.2. Simulation Environment

2. Connection of Simulation Environment with Python Script

As reinforcement learning code is written in python, it should communicate with the simulation environment. The communication is done through the WebSocket protocol. WebSocket protocol as specified in allows two-way bidirectional communication between server and client. In this case, the python “server” can push messages directly to simulation environment (e.g. steering and throttle actions), and the Simulation Environment “client” can also push information (e.g. states and rewards) back to the python server.

3. Training the car: -Deep Q Learning

Deep Learning

Deep Learning is defined as the class of machine learning algorithms that exploits many layers of non-linear information processing for supervised or unsupervised feature extraction and transformation, and for pattern analysis and classification.

Deep Q Networks (DQN)

Deep Q Networks are deep learning models that combine deep convolutional neural networks with the Q-learning algorithm. Using a nonlinear function such as a neural network as a function approximator for the Q-function is unstable. An important contribution made by DQNs is the use of experience replay and a target network to stabilize the training of the Q action-value function approximation with deep neural networks. Q-network thus refers to a neural network function approximator of the Q-function:

$Q(s,a;\theta) \approx Q*(s,a)$

Experience replay

Experience Replay consists in maintaining a buffer of the old experiences and train the network against them. During the training a buffer is used where previous experiences are stored. An experience consists of an observed state and action pair, the immediate reward obtained and the next state observed. When the buffer is full, new experiences usually replace the oldest ones in the buffer. An agent's experience at a time step *t* is denoted by e_t and is a tuple (s_t,a_t,r_t,s_{t+1}) consisting of the current state s_t, the action a_t, the reward r_t and the next state s_{t+1}. The experiences for all the time steps are stored in a replay memory, over many episodes.

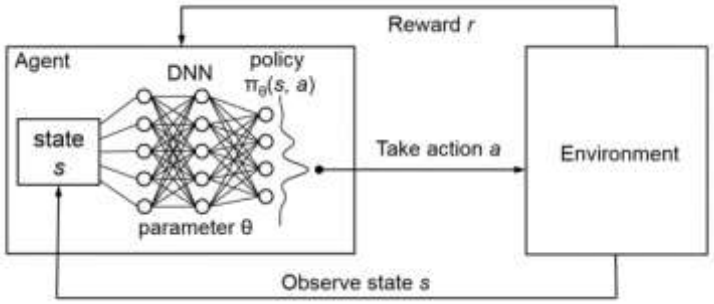


Fig.3. Deep Q Learning

State Space

The pixel images are taken by the front camera of the Donkey car, and perform the following transformations:

i. Resize it from (120,160) to (80,80)
ii. Turn it into grayscale
iii. Frame stacking: Stack 4 frames from previous time steps together
iv. The final state is of dimension (1,80,80,4)

Action Space

The Donkey car in the simulator takes continuous steering and throttle values as input. The throttle is set to a constant value of 0.7 for simplicity and to control the steering. The steering value ranges from -1 to 1. However, DQN can only handle discrete actions, so the steering value were discretized into 15 categorical bins.

Reward

Reward is a function of cross track error (cte) which is provided by the Unity environment. Cross track error measures the distance between the center of the track and car. The reward is given by the following formula:

reward = 1 - (abs(cte) / max_cte)

Where max_cte is just a normalizing constant so that the reward is within the range of 0 and 1. We terminate the episode if abs(cte) is larger than max_cte.

IV. RESULT

After training the DDQN for around 100 episodes on a single CPU and a GTX 1060 GPU the car starts navigating itself in a laned road with good accuracy. The entire training takes around 2 hours.



Fig.4 Result

V. FUTURE SCOPE

As of now, the agent follows the lane precisely by predicting the steering output at a constant speed. This has a limitation of driving the car at a constant speed. So, in the next stage of development of this model, speed can be incremented or decremented as per the condition of the road. For example, on a straight open road the car can accelerate whereas while turning the speed should be decremented. To achieve this, an appropriate reward function should be developed.
Currently the agent is driving in a virtual world. When it comes to the real world, there can be some situations which are unpredictable. Therefore, to train the model to face real world scenarios is a challenging task. One way to achieve this is to map the agent into a robust hardware prototype by utilizing the lane detection technique.

VI. CONCLUSION

The attempt made to make the car learn on its own using reinforcement learning is a success. No training dataset is given to the agent thus reducing the need of manually driving the car for hours to collect data. The agent is successfully training itself on its own in the simulation environment. The car learns to drive a complete track without drifting away from the lanes. The car Autonomous driving cars appear to be an important step in field of transportation and technology. Developments in autonomous cars is continuing and the software in the car is continuing to be updated.

VII. REFERENCES

[1] Tesla Autopilot 2020 [Online]. Available: https://www.tesla.com/autopilot
[2] Google Waymo 2020 [Online]. Available: https://www.waymo.com
[3] Donkey Car 2020 [Online]. Available: https://docs.donkeycar.com/guide/simulator/
[4] Unity 2020 [Online]. Available: https://unity.com/
[5] End to end deep learning for self-driving cars 2016 [Online]. Available: https://images.nvidia.com/content/tegra/automotive/images/2016/solutions/pdf/end-to-end-dl-using-px.pdf
[6] Autonomous Driving (Volvo) 2020 [Online]. Available: https://www.volvocars.com/en-kw/own/own-and- enjoy/autonomous-driving
[7] The path of Autonomous Driving (BMW) 202 [Online]. Available: https://www.bmw.com/en/automotive-life/autonomous-driving.html