2018 15th International Conference on Ubiquitous Robots (UR)
Hawaii Convention Center, Hawai'i, USA, June 27-30, 2018

# Driverless Car: Autonomous Driving Using Deep Reinforcement Learning In Urban Environment

Abdur R. Fayjie[*1], Sabir Hossain[*1], Doukhi Oualid[*2], and Deok-Jin Lee[*3]

*Abstract*— **Deep Reinforcement Learning has led us to newer possibilities in solving complex control and navigation related tasks. The paper presents Deep Reinforcement Learning autonomous navigation and obstacle avoidance of self-driving cars, applied with Deep Q Network to a simulated car an urban environment. The approach uses two types of sensor data as input: camera sensor ~~and~~ + laser sensor in front of the car. It also designs a cost-efficient high-speed car prototype capable of running the same algorithm in real-time. The design uses a camera and a Hokuyo Lidar sensor in the car front. It uses embedded GPU (Nvidia-TX2) for running deep-learning algorithms based on sensor inputs.**

## I. INTRODUCTION

The autonomous car has been in the headlines for a decade and still continues to dominate auto headlines [1]. The autonomous car has attracted the researchers, robotics communities and the automobile industries. Human driving is accident-prone. We are drunk, drowsy and tired. Our inefficiency to make better on-road decisions cause road accidents, property loss and deaths [2], [3]. The autonomous car brings us the possibilities to replace accident-prone human driver by providing comfort and safety.

The automated car concept was originated in 1920 [4], but it evolved truly in 1980 with CMU's Navlab and ALV projects. CMU's effort resulted in the ALVINN in 1989 [5], the first try in car autonomy which used a neural network to drive. The United States government invested in three projects for autonomous car which were demonstrated by US Army (Demo-I, Demo-III) and DARPA [6] (Demo-II) in the 2000s. From 2004 to 2007, DARPA organised 3 car autonomy challenges which got attention all over the world at that time [7]. The automobile industries like Ford, Mercedes, Volkswagen, Audi, Nissan, Toyota, BMW and Volvo came to play a bigger role in the autonomous car area by announcing their plans for it. In 2014, Google introduced Google car with sensor fusion technology (mainly camera and lidar sensor) [8]. One year later, Tesla brought their autonomous car in 2015. Nvidia has recently introduced a self-driving car with end-to-end convolutional neural network. In 2017, Audi stated "latest A8 would be autonomous at up to speed of 60 km/h using Audi AI [9]. The driver doesn't require safety checks such as frequently gripping the steering wheel." The Audi A8 is claimed to be the first production car to reach level 3 autonomous driving and Audi would be the first manufacturer to use laser scanners in addition to cameras and ultrasonic sensors for their system. On the 7th November 2017, Waymo announced that it has begun testing driver-less cars without a safety driver at the driver position [10]. However, there is an employee in the car. And the list still continues with Uber, Tata, Intel and other automobile industries.

Driving is a complex problem. The evolution of autonomous car involves complex modules, which uses traditional control algorithms [11], SLAM based navigation systems [12] and Advanced Driver Assistant Systems [13] in recent days. But all the previous methods fail to replace human driver completely. ~~But~~ increasing computing power of GPUs, accurate + sensors ~~and~~ + deep neural networks has made complete autonomy is realisable. ~~But~~ recent failure of Uber and Tesla, and the listed disengagements in [14] show that the autonomous car research needs more attention in the upcoming years to achieve 'better than human-level driving'.

In the context of 'human-level control', [15] presented a deep reinforcement learning approach which redesigned Q-Learning using a deep neural network. It introduces 4 steps to unstable Q-learning: Experience Replay, Target Network, Clipping Rewards and Skipping Frames. It achieved human-level performance (sometimes even better than human) tested in different complex game environment. The breakthrough achievement of DQN has motivated us to implement it in car autonomy for human-level performance.

In this paper, we present a DRL-based autonomous driving strategy for urban environment using Deep Q Network. We use sensor fusion on two types of input sensor data: vision based camera sensor ~~and~~ a laser sensor. The approach is tested in simulation environment, which is designed using Unity Game Engine. ~~We also design a car prototype capable of running our approach in real-time. The prototype uses Hokuyo Lidar Sensor and camera sensors embedded with Nvidia Jetson TX2 embedded GPU.~~ The paper structure is as follows: section I introduces autonomous car and the neural network approaches in car autonomy, section II presents the background and mathematical representation of Deep Q Network. Section III describes proposed methodology with the network architecture and configurations, action definitions and simulation environment. It also presents the learning curve during network training. Section IV discusses our car prototype capable of running DQN in real-time.

This research was supported by Unmanned Vehicles Advanced Core Technology Research and Development Program through the National Research Foundation of Korea (NRF), Unmanned Vehicle Advanced Research Center (UVARC) funded by the Ministry of Science, ICT & Future Planning, the Republic of Korea (No. 2016M1B3A1A01937245) and by the Ministry of Trade, Industry & Energy (MOTIE) under the R&D program (Educating Future-Car R&D Expert). (N0002428). It was also supported by Development Program through the National Research Foundation of Korea (NRF) (No. 2016R1D1A1B03935238).

*Center for AI & Autonomous Systems, Dept. of Mechanical Engineering, Kunsan National University, Republic of Korea

**Algorithm 1** DQN with experience replay by DeepMind

```
Initialise replay memory $D$ to capacity $N$
Initialise action-value function $Q$ with random weights $\theta$
Initialise target action-value function $\hat{Q}$ with weights $\bar{\theta}$ = $\theta$
For episode = 1, $M$ do
    Initialise sequence $s_1$=$x_1$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
    For $t = 1$, $T$ do
        With probability $\epsilon$ select a random action $a_t$
        Otherwise select $a_t = \arg\max_a Q * (\phi(s_t), a; \theta)$
        Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
        Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
        Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
        Sample random mini-batch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
        Set $y_j = \begin{cases} r_j, & \text{if episode terminates at step } j+1, \\ r_j + \gamma \max_a' \hat{Q}(\phi_{j+1}, a'; \bar{\theta}), & \text{otherwise.} \end{cases}$
        Perform a gradient descent step on $(y_j Q(\phi_j, a_j; \theta))^2$ w.r.t. network parameters $\theta$
        Every $C$ steps, reset $\hat{Q} = Q$
    end for
end for
```

Section V concludes the paper with a summary of the work.

## II. Deep Q Network

Chris Watkins introduced Q-Learning in his Ph.D. thesis [16]. It is based on Markov Decision Process (MDP). In 2005, Riedmiller used a neural network to approximate Q-function. It is further researched and presented in 2015 with a deeper neural network by Mnih et. al. [15] This approach uses a deep fully connected convolutional neural network to approximate Q-function based on image input. It takes images of the environment as input to provide the game states. Experienced Replay based Reinforcement Learning is applied using a reward function to train the agent. The agent explores the environment and acts. Every action is provided with a given reward value (either positive or negative). The goal of the agent is to maximise the positive rewards and minimise the negative rewards. This approach has been verified with the implementation to play various Atari 2600 games. It resulted in human-level performance, even better in some cases.

Previously known Q-Learning has its limitations. It is useful in a small state-space but fails in a larger state-space. Minh et. al. brings Q-Learning to a larger state-space by adding experience replay memory and a separate target network. It solves the correlation between consecutive updates and correlation between target and predicted Q-values. The method represents an action-value function, which generates expected utility when an action is taken from an allowed action set. It stores the state-action transitions ($s \rightarrow a \rightarrow s'$) provided a reward value ($r$) to attain the next state ($s'$) and updates the Q- Network. At each training step, the reward is estimated by the Q function for a particular action from state $s$ to the next state $s'$ and maximises the reward. The discount factor is used for future rewards and relative error, in particular, is calculated with the knowledge (comes from exploration) of observed reward $r$ and next state $s'$. In each training steps, for an action-state transition ($s \rightarrow a \rightarrow s'$), the maximised positive reward $r$ is estimated with the Q-function. A discount factor is used for the future rewards and relative errors. It is calculated based on the agent's knowledge (comes from exploration) on observed reward $r$ and next state $s'$. The stored action-state transitions $(s, a, r, s')$ are divided in mini-batches and called at random for training. It provides efficiency in the use of previous experience, by learning with it multiple times. The Q-learning updates are incremental and do not converge quickly. Multiple passes with same data provide better outcome especially when there is low variance in immediate outcomes (reward, next state) given the same state, action pair. It also provides better convergence behaviour in Q-function approximation during training. Also in Q-Network, current Q value is based on the previous one. So previous $Q(s, a)$ is closer to next $Q(s, a)$. Due to this closer nature of both the Q values, the target is likely to shift with each update. Thus, it leads the network to divergence. So, a target network, which is a copy of training network (or previous network) but frozen on time (also Frozen Network) is added to copy the weights on specific training interval. To be effective the interval must be large enough to leave enough time for the original network to converge. Algorithm 1 presents the DeepMind's DQN algorithm with experience replay.

## III. Proposed Methodology

### A. Network Architecture

We design a deep neural network based on fully-connected CNN to approximate Q-function. The network consists of 3 convolutional layers and 4 dense layers. Two types of data input: front camera image (RGB image of size 80 x 80) and lidar data (grid-map of size 80 x 80) are fed into the neural network, processed using the convolutional layers for each

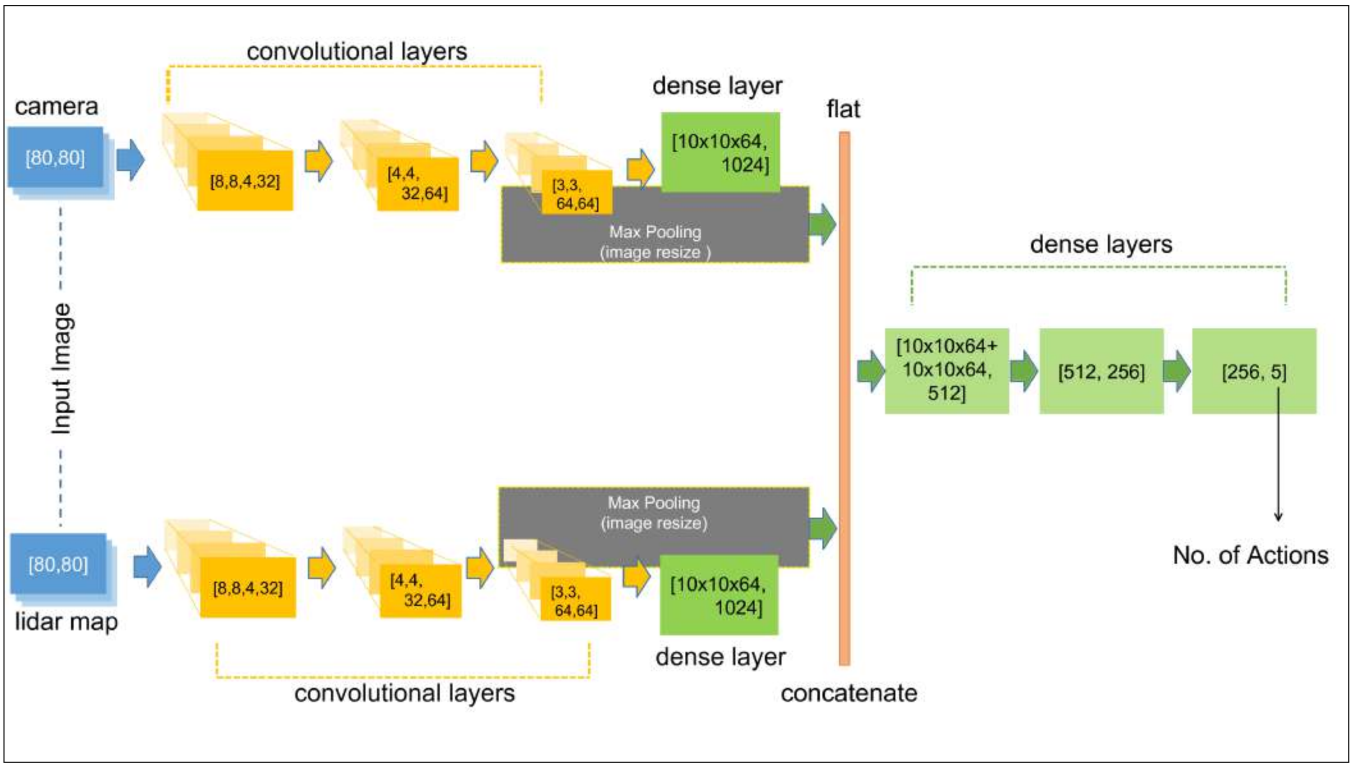
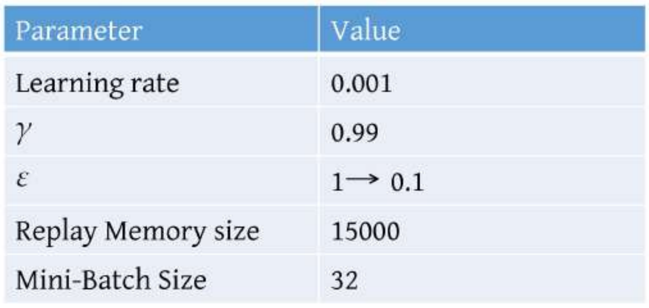


Fig. 1. Our DQN architecture: the deep neural network is based on fully-connected convolutional neural network.

data type. Two data types are fused using a flatten layer, preceded by a dense layer and a pooling layer (used for data resize) for each data type. Our target network follows the same architecture, which is updated every 6000 steps. The neural network is trained to accept 5 actions with defined reward. The network uses mini-batch of size 32, trained with learning rate 0.001. Figure 1 illustrates the discussed network and table I presents the network configurations for the training.

TABLE I
PARAMETER CONFIGURATION FOR TRAINING



| Parameter | Value |
|---|---|
| Learning rate | 0.001 |
| $\gamma$ | 0.99 |
| $\varepsilon$ | $1 \rightarrow 0.1$ |
| Replay Memory size | 15000 |
| Mini-Batch Size | 32 |

### B. *Action and Reward Definition*

Action definition is one of the key points in Reinforcement based learning strategy. We choose 5 actions for the agent to be trained with DQN: Keep Going (reward= speed/5), Left (reward = -0.6), Right (reward = -0.2), Accelerate (reward = +1) and Brake (reward = -0.4). The basic action for our agent is to keep going (or in other terms do nothing) in the

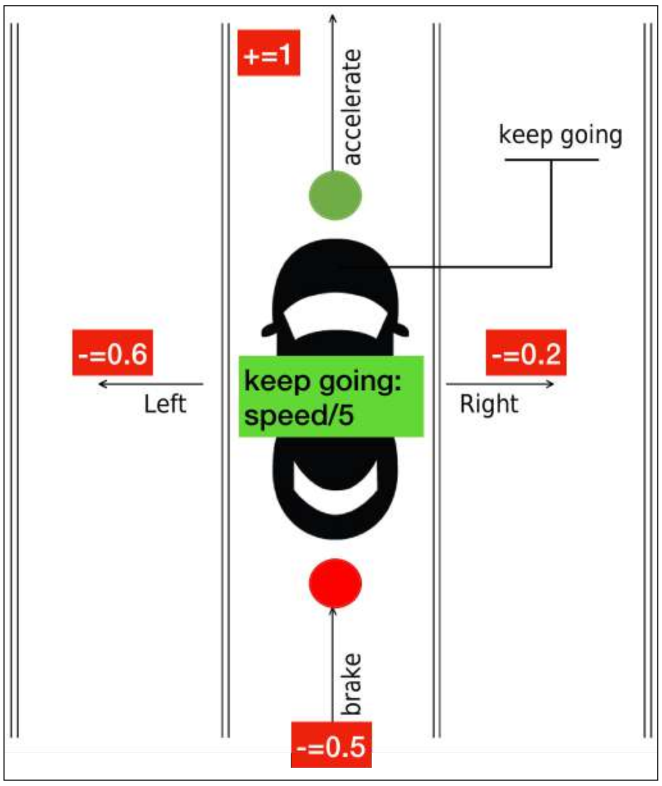


Fig. 2. Action definition for the car

road. For a hit, a bad reward of value - 6 is assigned. It needs to learn how to accelerate when the front is empty and decelerates (brake) when there are other cars or agents in front. Our action definition also defines left and right motions for the car to teach the agent to change lane. Figure 2 illustrates the actions and their corresponding rewards for the car to be trained using DQN.

### C. Simulation Environment

We design a simulated urban environment in Unity Game Engine. The environment is based on Geneve environment and consists of 5 lanes highways. It is a city-like structure (see fig. 3) with buildings, trees, and street lights. The environment is cooperative; other cars run on the highways (see fig. 4). The basic goal in our design is to mimic the real-time urban environment as close as possible in the simulation. It allows us to train the DQN to drive the car autonomously, with almost real-time driving behaviour.

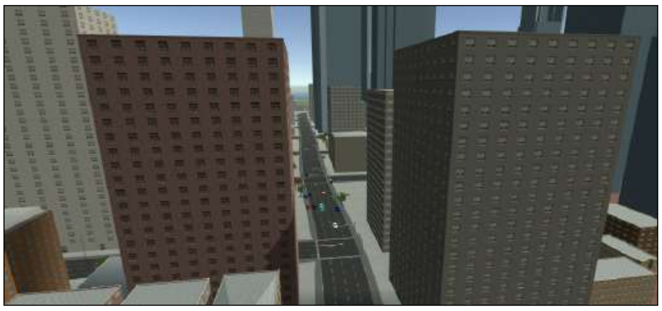


Fig. 3. Simulated urban environment

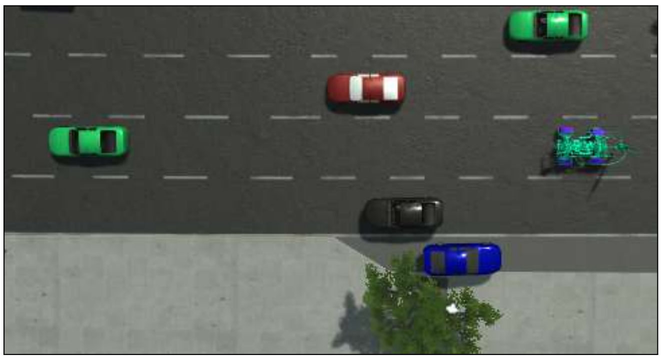


Fig. 4. Car in simulation environment

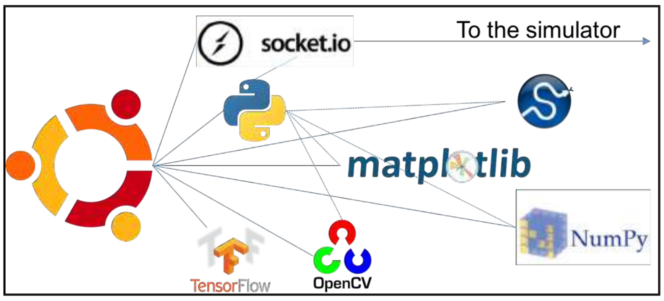


Fig. 5. Software architecture for simulation environment

### D. Software Architecture

The simulation test is done in a Ubuntu platform (v. 16.04LTS) installed with python 3 (v. 3.6.2). The Deep Q Network is designed using tensorflow API. OpenCV (v. 3.3.0) is used for image related tasks. It uses numpy, scipy and matplotlib. The simulator is connected to the DQN algorithm using Socket connection (python-socket-IO). Figure 5 illustrates the software used in simulation.

### E. DQN Training

We feed the neural network with 4 consecutive frames of the camera (front, size 80x80) images. The lidar data are plotted as map frames, and 4 consecutive frames are fed into the network. Both the data types are fused using a flatten layer preceded by a pooling layer pooling from last convolution and first dense layer. Figure 6 shows the learning curve with cumulative reward for 100000+ steps (plotted total reward per 3000steps). It is gradually increasing in nature which represents agent's successful learning over time. Figure 7 and 8 present the camera output and the lidar output of the running car at time $t$.

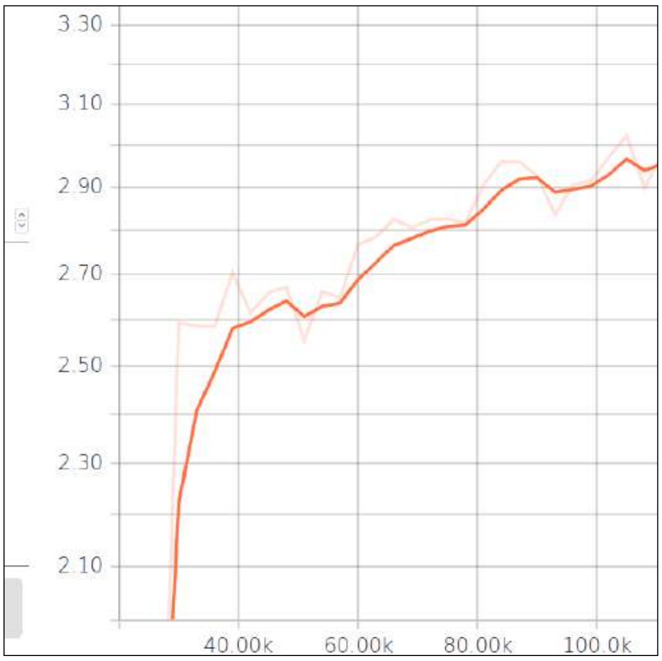


Fig. 6. The learning curve during training: cumulative reward/3000 steps, 100000+ steps

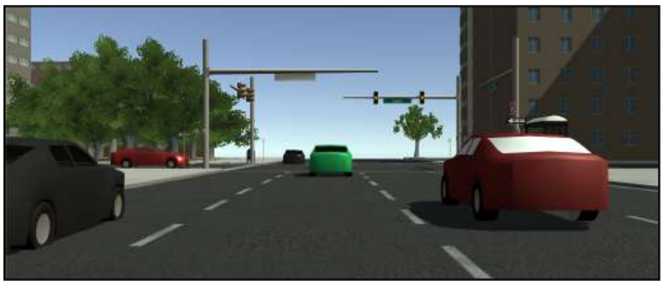


Fig. 7. front camera output

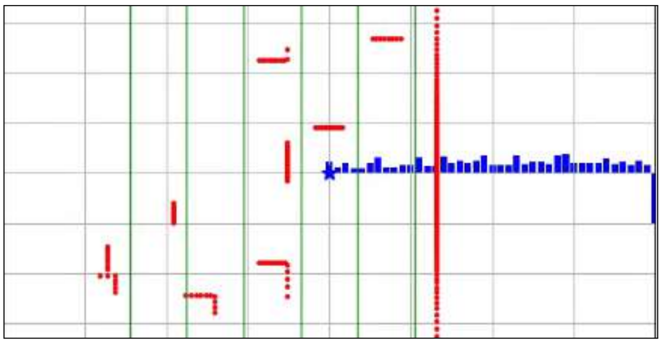


Fig. 8. lidar data generating map

## IV. Our Real-Time Car Prototype

Based on the simulation set-up, we designed a car capable of running Deep Neural Networks in real-time. Deep Neural Networks are data-centric. The performance of such methods depends on the computing power of the system. We consider Traxxas Remote Control Racing car and redesign it with Nvidia-Jetson TX2; the fastest embedded GPU. We add a Hokuyo lidar sensor (model: Hokuyo UTM-30LX Scanning Laser Rangefinder) and a camera. Ubiquity wifi access point provides the car with required communication to the ground station. Figure presents our car prototype capable of running DQN algorithm in real-time with sensor data fusion. Figure 9 presents our designed car prototype capable of running DQN in real-time.

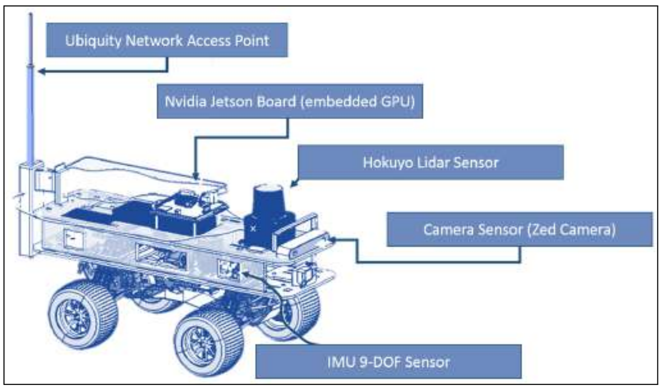

Ubiquity Network Access Point
Nvidia Jetson Board (embedded GPU)
Hokuyo Lidar Sensor
Camera Sensor (Zed Camera)
IMU 9-DOF Sensor

Fig. 9. The car prototype for the Real-Time DQN implementation

## V. Conclusion

In this paper, we presented a reinforcement-learning based approach with Deep Q Network implemented in autonomous driving. We defined the reward function and agent (car) actions and trained the neural network accordingly to maximise the positive reward. We also discussed our sensor fusion technique: data concatenation of Lidar and Camera. We presented the simulated urban environment that was designed using Unity Game Engine. We tested the car for autonomous driving with the trained network in simulation. The simulation tests showed that the trained neural network was capable of driving the car autonomously by taking proper decisions (choose an action from action set) in the environment. Finally, we designed a car prototype based on our simulation experiments which is capable of running a DQN in real-time.

## Acknowledgment

This research was supported by Unmanned Vehicles Advanced Core Technology Research and Development Program through the National Research Foundation of Korea (NRF), Unmanned Vehicle Advanced Research Center (UVARC) funded by the Ministry of Science, ICT & Future Planning, the Republic of Korea (No. 2016M1B3A1A01937245) and by the Ministry of Trade, Industry & Energy (MOTIE) under the R&D program (Educating Future-Car R&D Expert). (N0002428). It was also supported by Development Program through the National Research Foundation of Korea (NRF) (No. 2016R1D1A1B03935238).

## References

[1] TU-Automotive, “Driverless vehicles will continue to dominate auto headlines tu automotive [online],” April, 2016, available: http://analysis.tu-auto.com/autonomous-car/driverless-vehicles-will-continue-dominate-auto-headlines. [Accessed: 10-April-2018].

[2] L. Fridman, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, J. Kindelsberger, L. Ding, S. Seaman, H. Abraham, A. Mehler, A. Sipperley, A. Pettinato, B. Seppelt, L. Angell, B. Mehler, and B. Reimer, “Mit autonomous vehicle technology study: Large-scale deep learning based analysis of driver behavior and interaction with automation,” Nov 2017, available: https://arxiv.org/abs/1711.06976.

[3] WHO, “Global status report on road safety 2015. world health organization,” 2015.

[4] Sean O’Kane (The Verge), “History of autonomous car [online],” available: http://research.omicsgroup.org/index.php/History_of_autonomous_car [Accessed: 21-April-2018].

[5] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” in *Advances in neural information processing systems*, 1989, pp. 305–313.

[6] Q. Chen, U. Ozguner, and K. Redmill, “Ohio state university at the 2004 darpa grand challenge: developing a completely autonomous vehicle,” *IEEE Intelligent Systems*, vol. 19, no. 5, pp. 8–11, 2004.

[7] K. Mahelona, J. Glickman, A. Epstein, Z. Brock, M. Siripong, and K. Moyer, “Darpa grand challenge,” 2007.

[8] N. Spinrad, “Google car takes the test,” *Nature*, vol. 514, no. 7523, p. 528, 2014.

[9] Michael McAleer (The Irish Times), “Audi’s self-driving a8: drivers can watch youtube or check emails at 60km/h [online],” July 11, 2017, available: https://www.irishtimes.com/life-and-style/motors/audi-s-self-driving-a8-drivers-can-watch-youtube-or-check-emails-at-60km-h-1.3150496 [Accessed: 19-April-2018].

[10] Andrew J. Hawkins (The Verge), “Waymo is first to put fully self-driving cars on us roads without a safety driver [online],” Nov 7, 2017, available: https://www.theverge.com/2017/11/7/16615290/waymo-self-driving-safety-driver-chandler-autonomous [Accessed: 19-April-2018].

[11] P. G. Trepagnier, J. E. Nagel, M. T. Dooner, M. T. Dewenter, N. M. Traft, S. Drakunov, P. Kinney, and A. Lee, “Control and systems for autonomously driven vehicles,” Feb. 28 2012, uS Patent 8,126,642.

[12] J. Guivant, E. Nebot, and S. Baiker, “Autonomous navigation and map building using laser range sensors in outdoor applications,” *Journal of robotic systems*, vol. 17, no. 10, pp. 565–583, 2000.

[13] A. Paul, R. Chauhan, R. Srivastava, and M. Baruah, “Advanced driver assistance systems,” SAE Technical Paper, Tech. Rep., 2016.

[14] V. V. Dixit, S. Chand, and D. J. Nair, “Autonomous vehicles: disengagements, accidents and reaction times,” *PLoS one*, vol. 11, no. 12, p. e0168054, 2016.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.

[16] C. J. C. H. Watkins, “Learning from delayed rewards,” Ph.D. dissertation, King’s College, Cambridge, 1989.