



An Autonomous Vehicle With Deep Reinforcement Learning for Collision Avoidance

Hayder Salah and Hayder Salah

University of ALQadisiyah, College of computer science and Information Technology Diwaniyah, Iraq
Haydersalah235@gmail.com, ali.obied@qu.edu.iq

Abstract :

Keeping an autonomous car from colliding with other vehicles is a challenging challenge to do. Most traditional approaches in this sector depend on model-based solutions, which demand an understanding of vehicle dynamics and an accurate model of vehicle behavior in order to predict the route of the controlled vehicle and surrounding vehicles. It is difficult for these systems to predict and simulate the driving habits of others around them.

According to this study, an agent uses deep reinforcement learning to prevent collisions by calculating the distances to neighboring entities and outputting steering angles and accelerations. A variety of roads and cars are included in the Carla Simulator so that the Learning Agent may interact with them and gather data. Such training results in intelligent driving behavior; avoiding crowded areas and traffic situations; reducing speed to minimize rear or frontal accidents; and maneuvering as necessary to avoid side impacts.

Keywords: Autonomous driving, Deep Reinforcement Learning, Neural Networks, Collision Avoidance, Deep Deterministic Policy Gradients.

DOI Number:10.14704/nq.2022.20.8.NQ44590

NeuroQuantology 2022; 20(8): 5619-5635

1. Introduction

Controlling a high-dimensional sensory input is a major challenge for any reinforcement learning application. One approach to this challenge is to use handcrafted features and linear parameterization of the policy or value function. A high-quality product is essential to this strategy's success. An example of a workable solution is the capacity of neural networks to pick out key characteristics from raw sensory input. Mineh et al. [1] demonstrate a convolutional neural network (CNN) trained using a variant of Q-Learning that can learn

effective control rules. It was our goal to adapt this deep Q-Learning approach and Markov decision processor to the more harder reinforcement learning job of autonomously driving an automobile in 3D.

2. Background

2.1. Related Work

By using the reinforcement learning approach, Min et al. [3] were able to develop highways driving policy. Driver assistance systems might be improved even further with the addition of a supervisor agent trained using deep



distributional reinforcement learning. An end-to-end strategy is used to teach the supervisor agent how to use LIDAR data as well as camera images to create an action plan. However, they failed not demonstrate the method's effectiveness on a different simulation.

Recent efforts have been made to use RL algorithms to handle a range of issues, including the development of autonomous vehicles (AVs). To be safe, self-driving cars must have a degree of security that isn't always guaranteed in the real world. Here, we examine some of the most current research on RL algorithms for AVs, with a particular focus on publications that include restrictions or other ways to enhance the safety of the agents.

Self-driving car simulator CARLA was first published in 2017 by Dosovitskiy et al. [4] and is currently being actively developed and updated every couple of months. One of the fundamental driving tasks introduced in this study has since become a typical benchmark for the development of CARLA-based agents. For this objective, they have three standard agents that may be used. One of the earliest systems is a modular pipeline that includes independent subsystems for visual perception (the ability to see), planning (the ability to plan), and controlling the vehicle. End-to-end imitation learning from a human driver in the environment is used in the second technique. The last approach completes the job with a straightforward RL A3C RL agent.

While Liang et al. [5] provide an RL agent to fulfill the CARLA benchmark, it is not the only one. Before RL can be used to fine-tune the agent model, it must first be warmed up by learning via imitation of human demonstrations. When an agent takes the incorrect turn or turns too abruptly when the car should be travelling straight, they are penalized with a steady negative reward that they may use to improve their driving skills.

Toromanoff et al. [6] constructed an RL agent for CARLA without the use of imitation learning or pre-training, however. Desired speed, location, and rotation all play a role in their reward module. It's not only the agent's behaviors that determine how much they're worth; it's also the present status of the environment. According to the existing state of affairs, in order to maximize your reward, the vehicle should be able to meet the appropriate speed value. With each approaching slow car or red light, the optimal speed lowers, whereas with no barriers in front of the vehicle, the ideal speed rises. Additionally, depending on whether a vehicle is travelling straight or making a turn, the intended rotation and the corresponding reward value alter. The agent's performance in CARLA is significantly improved because to this dynamic incentive signal.

A convolutional neural network was utilized by S. Park et al. [7] to change vehicle steering parameters. The CNN inputs from black box photos were used to regulate the steering values of an autonomous driving car in this investigation. CNNs are typically composed of three layers, comprising convolutional and pooling layers, and an approach with two completely connected layers was presented. In this CNN, vanishing point coordinates from images were used to calculate steering angles as output values. The driving environment and lane disappearances make it challenging to discover stable vanishing points, yet the CNN was able to obtain steering angle prediction errors within an acceptable range and produce significant results.

2.2. Reinforcement Learning

RL is a machine learning technique that uses reinforcement learning (ML). As opposed to other ML approaches, its goal is to learn diverse behaviors depending on the surrounding environment.



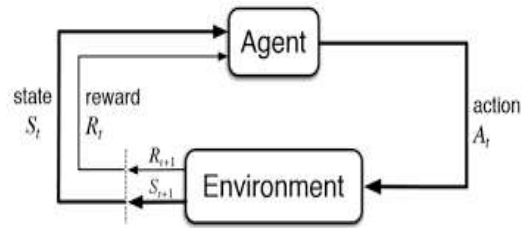


Figure 1: Reinforcement learning schema.

Figure 1 shows a basic schema of an RL setting. The primary character is named Agent. Actions are selected and carried out by the learning system after careful observation of the surrounding environment. Policy is the method used by the agent to choose the most effective Actions in order to maximize Reward.

In the context of acting, an action is a series of potential motions that the Actor may do. Following an Action, the Environment will either give you a reward or a punishment. It is the feedback that tells whether or not the Agent's Action was successful or unsuccessful. The Agent's environment refers to the physical world in which he or she is situated. The state (observation) of the Agent in the environment is the present and concrete situation of the Agent. For the most part, the Agent is in charge of monitoring the environment, making decisions based on that information, and taking action. Afterwards, in order to maximize Reward over time, the Policy discovers on its own which Actions the Agent should do while it is in a certain State.

2.2.1. Challenges in modern RL

2.2.1.1. The issue of credit assignment

It is difficult to give merit to the agent's acts. Some actions may pay off long after they are completed. The reward gained after doing a series of actions cannot be linked to any one activity in particular. In contexts with delayed rewards, standard Deep Reinforcement Learning (DRL) algorithms are prone to errors (like Go, or chess). As a result, more sophisticated strategies are required to determine which activities to encourage.

2.2.1.2. The exploration-exploitation trade off

RL jobs aren't the only ones where this trade-off is relevant. Whenever a person has to choose

between collecting new knowledge or settling for the best-so-far choice, the exploration-exploitation trade-off develops [8, 9]. If you're at a restaurant and you're torn between the dish you've had the most success with and a new one you've never had before, the exploration-exploitation trade-off comes into play.

It is possible that an agent in real life would settle for suboptimal behaviors early on in the training process if it is simply motivated by maximizing environment rewards, since this may prevent it from exploring superior actions that are wrongly believed to be less lucrative. On the other side, encouraging it to go out and explore might lead to a lack of training convergence, since the actions it does may be too random to get relevant information from the environment. The state space is just too large for complicated settings. It is not computationally possible to explore the complete state and action space, the agent must confine itself to a very narrow range of spaces. The agent's training can only be successful if he or she knows how much to investigate.

2.3. Markov decision process

Our decision-making process may be streamlined using the Markov Decision Process (MDP). Agents interact with their surroundings in a time-ordered manner. While interacting with the surroundings, the agent, it provides a snapshot of its surroundings. The agent picks the action to do based on the representation of the state, as seen in the picture above figure 1.



The activity is completed and the agent is rewarded for their efforts. Taking action, altering states, and rewarding are all iterations of this cycle. The agent's ultimate objective is to maximize the overall benefits at all times.

Let's get a better sense of how things work:

First, The condition of the environment is S_t at a certain time t .

Secondly, the agent observes the current state S_t and chooses an action a_t

Finally, the environment shifts into the new state S_{t+1} and the agent is rewarded for their efforts. R_t

Observations from perceptual data are used by the agent in a partly observable Markov decision process (POMDP) to detect changes in the environment, which is followed by a reward.

POMDP $M := (I, S, A, R, P, \gamma)$, where $I, S, A, R, P,$ and γ are the six components of the POMDP.

* I : Observations

* S : Finite set of states

* A : Finite set of actions

* R : Reward function

* P : transition probability function

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{a \in A} Q(S_{t+1}, a) - Q(S_t, A_t)).$$

where:

$\alpha \in [0,1]$ is the pace at which new information is being absorbed. It determines how often Q values are recalculated at a particular time t .

* γ – discounting factor for future rewards.

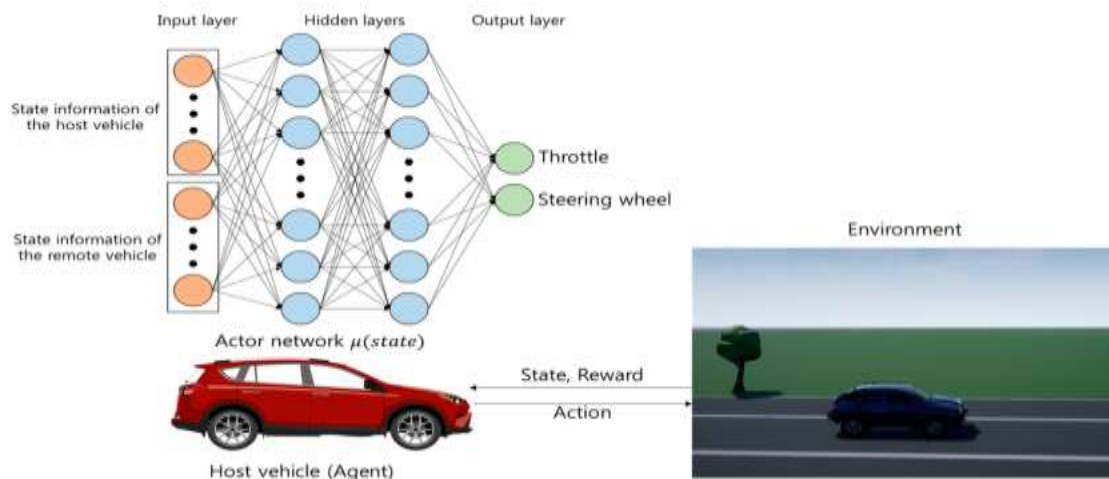
DRL's goal is to discover the ideal value-action function, or desired policy, that maximizes reward at each given time step (Q-function).

2.4. Q-Learning

Q-learning is often used in self-driving car DRL algorithms. This method does belong in the field of model-free education. To learn without a model, the agent will try to get as close as possible to the ideal state-action combination. Q-values and action-value pairs are still inspected and adjusted in accordance with policy (see the equation below). After making a mistake, the goal is to interact with the environment and make adjustments so that the best potential policy may be discovered.

Q-learning can find the optimum state-action value pairings if there are enough data samples or observations. With infinitely accessible actions, Q-learning has been demonstrated to converge to the best possible state-action values for an MDP with a chance of one. An example of Q-learning may be summarized using this equation.

5622



It's critical to keep in mind that the agent will learn from his or her mistakes in order to take the best possible action.

3. Training the self-driving with CNN

Deep learning algorithms in self-driving automobiles We will focus (CNN). To detect and categorize various portions of the road and to make suitable judgments through Reinforcement Learning, these systems mostly employ CNN.

It employed neural networks to recognize lines, partition the area, find its way around, and drive itself. However, because of its sluggish processing and sparse data, it was only partially effective.

Autonomous driving is becoming stronger all the time thanks to today's high-performance

3.1. Perception

Perception is essential for self-driving automobiles because it allows the vehicle to take in the environment around it and identify and categorize the objects it encounters. The automobile must be able to rapidly distinguish things if it is to make sound judgements.

Traffic lights, pedestrians and road signs, parking spaces, lanes as well as many other features must be recognized and classified by the automobile. It also needs to know the precise distance between itself and the

graphics cards and CPUs. If widely adopted, it has the potential to lessen traffic congestion while also improving safety.

Cars with self-driving features make their own decisions. Different sensors, such as cameras, LiDAR, RADAR, GPS and inertia sensors may all be used to generate streams of data for them. Using deep learning algorithms, this data is modeled, and these algorithms make judgments based on the environment the automobile is in.

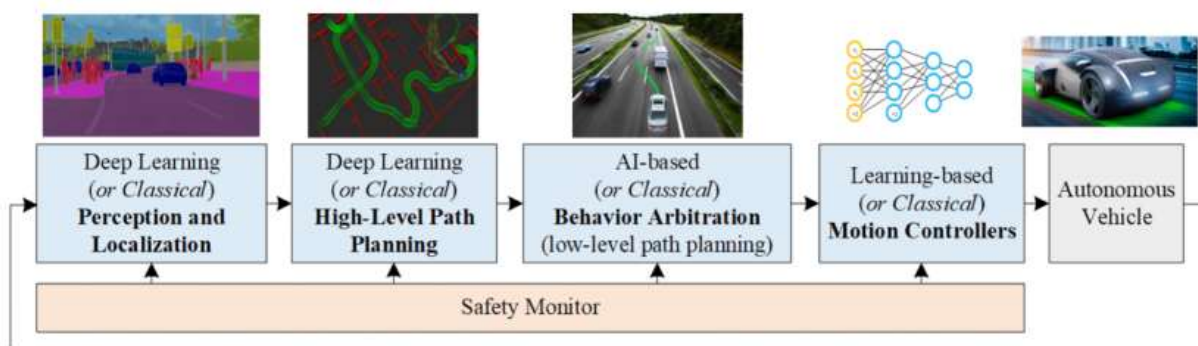
There is a driving decision-making pipeline shown in the graphic above. The many sensors that collect data from the surrounding environment are critical components of this strategy.

A self-driving automobile needs three types of sensors to be able to see objects at this degree of detail:

1. Camera
2. LiDAR
3. RADAR

3.1.1. Camera

Camera vision enables the automobile to do numerous tasks, including classification, segmentation, and localization, all of which are supported by the camera. High-resolution



surrounding objects.. More than merely seeing and categorizing, perception gives the system the ability to gauge distance and determine whether or not to slow down or brake as a result of that information being available.

cameras that correctly depict the surrounding environment are required.

fuse together many cameras provide a 360-degree image of the surroundings in order to ensure that the automobile gets visual input



from all four sides: front, rear, left, and right. There is a long-range and a short-range view for

better perception with these cameras, with the long-range being up to 200 meters.



5624

As an added benefit, while parking, the camera gives a bird's eye perspective to aid in making smarter decisions.

It's useless in poor visibility, such as dense fog or heavy rain, and more worse at night, when the cameras handle all the perception-related work. A camera's ability to record images under harsh settings is limited by the amount of noise and inconsistencies it can pick up.

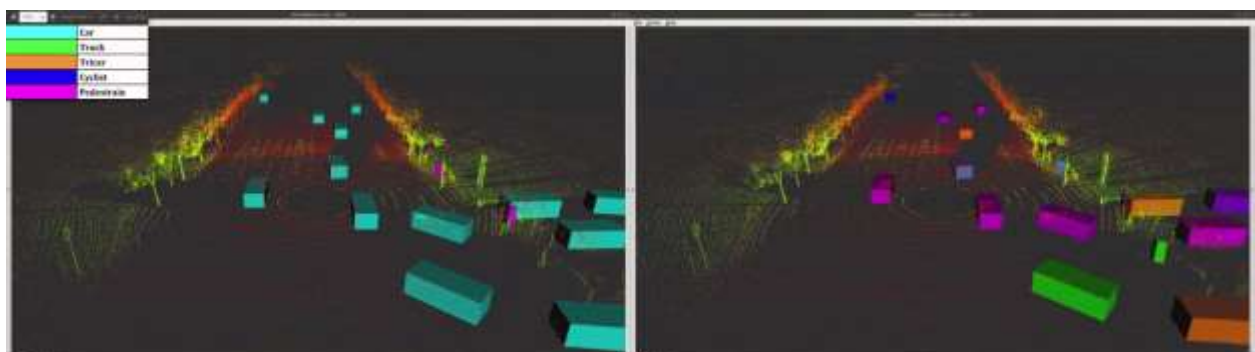
It is necessary to have sensors that can work without light and also sense distance in order to get beyond these constraints.

3.1.2. LiDAR

By measuring how long a laser beam takes to reflect off an item after being fired, a technique known as LiDAR may be used to determine an object's distance.

A camera in the automobile can only capture photographs of what's happening on in front of and behind it. To get a 3D view of what's happening around the automobile, you need to integrate the camera with the lidar sensor.

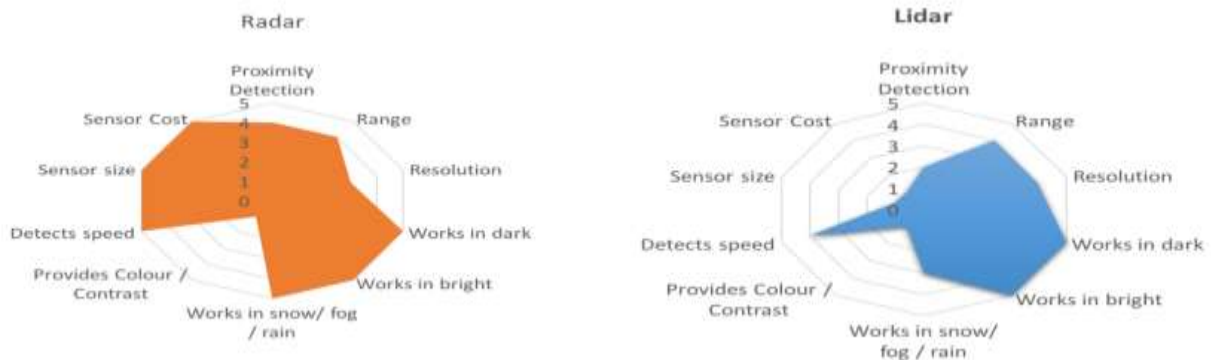
So, LiDAR is able to gather spatial data. Deep neural networks may then be used to forecast what other cars and objects will do next. An junction with several exits, where the automobile can assess all other vehicles and determine the safest course of action, is an excellent example of when this kind of technology comes in handy.



3.1.3. RADAR

When it comes to military and civilian uses alike, RADAR technology is indispensable. It was initially employed by the military to locate items. Radio wave transmissions are used to determine distance. Today, it's found in a wide range of cars and is a key component of the self-driving automobile. In contrast to lasers, RADARs use radio waves, which means they may operate in any environment.

Radars are noisy sensors, and that must be taken into consideration. In other words, even if the camera doesn't identify any impediments, the radar will.



Self-driving vehicle (in green) utilizing LiDAR to identify and estimate the size and form of objects in its immediate vicinity. If you look at the image below, which was acquired with a RADAR sensor, you'll notice a lot of additional commotion .

5625



RADAR data must be sanitized in order to make accurate forecasts and judgments. Thresholding is the process of separating weak signals from strong ones. Filtering and interpretation of the signal are done using Fast Fourier Transforms (FFTs).

3.2. Localization

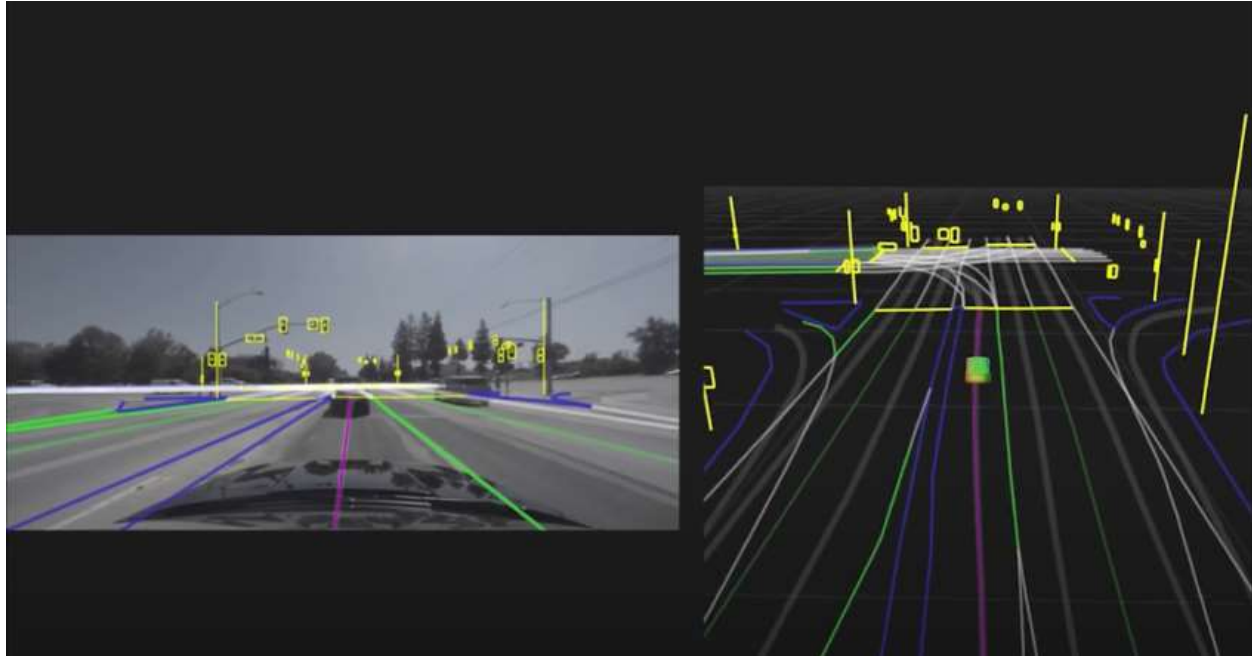
It is a technology known as Visual Odometry that helps self-driving automobiles keep track of

their location and orientation while on the road (VO).

Video overlay (VO) operates by synchronizing crucial moments in successive video frames. Each frame's critical points are sent into a mapping algorithm and utilized as input. It is possible to categorize adjacent items such as roads, people, vehicles, and more using a

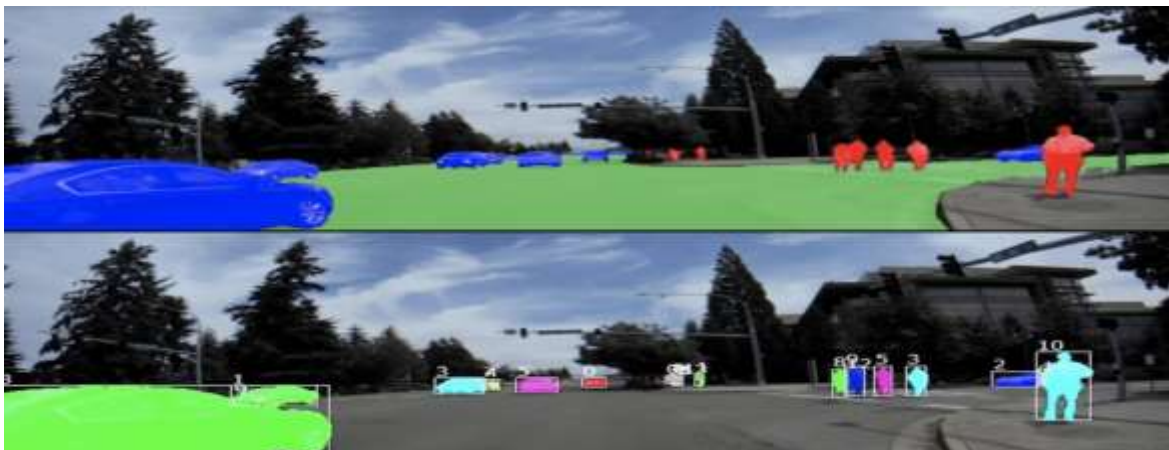


mapping technique such as “Simultaneous localization and mapping (SLAM)”.



To increase the performance of VO and to identify diverse things, deep learning is often utilized. Using point data, “neural networks, such as PoseNet and VLocNet++”, can predict the 3D position and orientation. It is possible to obtain scene semantics from these predicted 3D coordinates and orientations, as shown below.

5626



3.3. Prediction

It's quite difficult to determine the motivations of human drivers. Instead, it's driven by feelings rather than rationality. Because it's impossible to foresee what other drivers or pedestrians will do next, a system

that can anticipate their behaviors is critical for maintaining a high level of road safety.

With a full 360-degree vision of its surroundings, the automobile can take in and interpret all of the available data. After being input into the deep learning system, it is able to predict what other road users will do. As in a

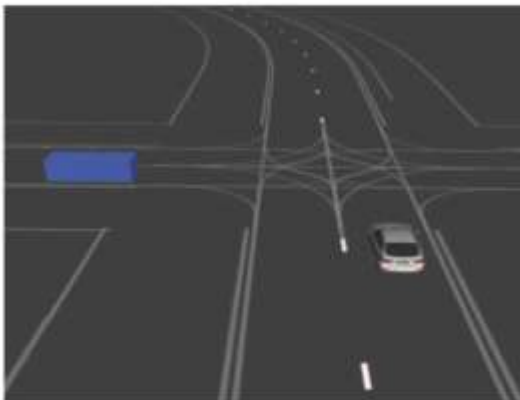


game, the player has a limited amount of moves and must choose the best one in order to win the game.

Object identification, segmentation, and localization are all made possible by the sensors in self-driving automobiles. The automobile can make predictions about what's around it using several types of data representation.

It is possible to train a deep learning system to model pictures and cloud data points from LiDAR and RADARs. It is conceivable for an automobile to prepare for every potential braking, stopping, slowing, changing lanes, and so on by using the same model throughout inference.

Using deep learning, self-driving vehicles are able to comprehend complicated visual tasks, locate themselves in the surroundings, increase



A self-driving automobile may be seen approaching an intersection in the picture to the right. The junction is being approached by a second vehicle, this one colored blue. Self-driving cars must determine if the other vehicle will travel straight, left, or right in this situation. To avoid a collision, the automobile must select which move to make in each situation.

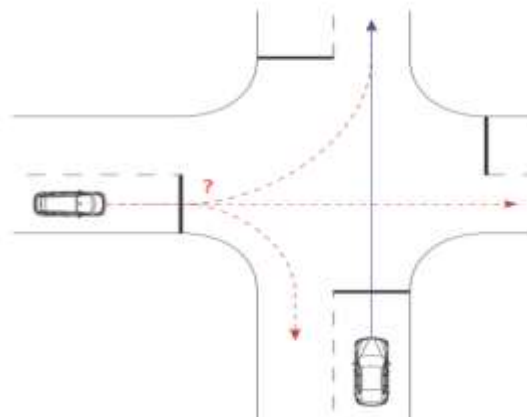
Choosing the appropriate course of action requires that the vehicle be equipped with sufficient data. We discovered that the sensors aid the vehicle in gathering data, and that deep learning techniques may be used for localization and prediction.

perception, and perform kinematic motions. This promotes both public safety on the road and convenience for drivers.

But the most difficult aspect is deciding which of a limited number of options is the best course of action.

3.4. Decision-making

In self-driving automobiles, decision-making is essential. Dynamic and accurate systems are needed in an unpredictable environment. Sensor readings aren't always accurate, and drivers often make erratic decisions while behind the wheel. There is no way to directly quantify these things. Even if we could accurately measure them, we would still be unable to accurately forecast them.



There are a n different actions or maneuvers that an automobile may do dependent on its location and the surrounding environment, which is known as localization. The issue remains: which of the numerous possible actions is best?

We employ deep reinforcement learning for making choices (DRL). Additionally, the Markov decision process (MDP) is what drives DRL's decision-making (It was previously explained where we talked about reinforcement learning).

In most cases, an MDP is utilized to make predictions about how other drivers will act in the future. The more items, particularly

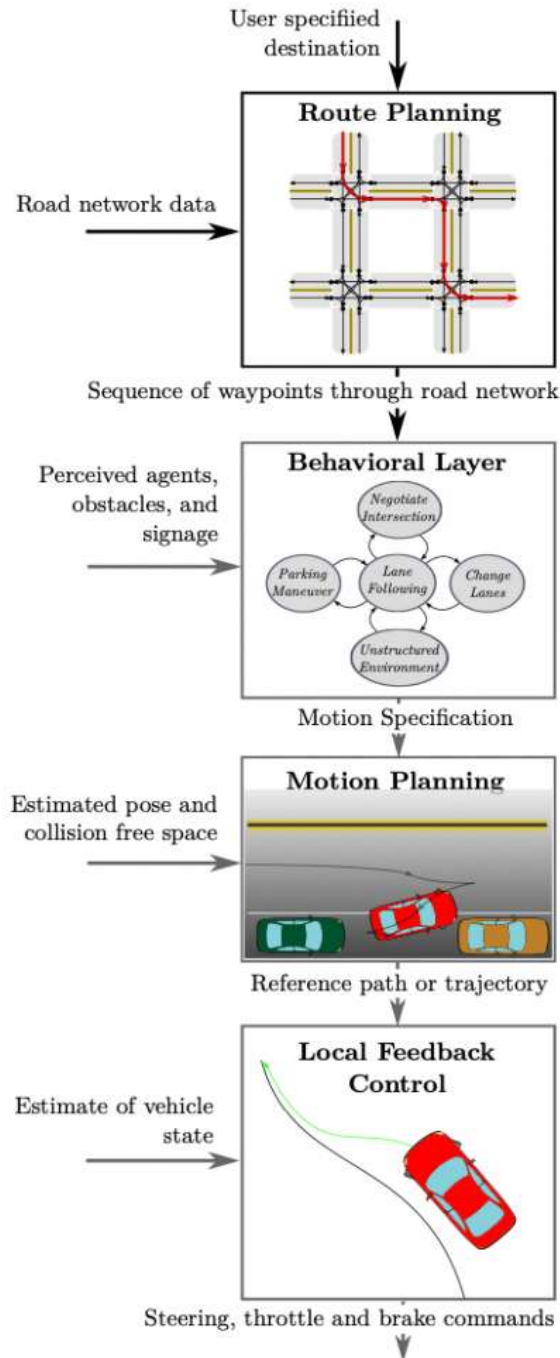


those in motion, the more difficult the situation might get. The self-driving vehicle will be able to do more maneuvers as a result of this.

The deep learning model is tuned utilizing Bayesian optimization in order to solve the challenge of determining the optimum move for itself. In certain cases, a combination of a hidden Markov model and Bayesian optimization is utilized to help make decisions. In self-driving automobiles, decision-making is often hierarchical. There are four parts to this procedure:

- A. **“Path or Route planning”**: The first of four decisions a car must make is deciding on a route. After entering the environment, the car should construct the best possible route to the provided destination. The goal is to come up with the best solution out of all the others.
- B. **Arbitration of behavior**: Once the route has been planned, the automobile must find its own way around it. Static components, such as roads, junctions, typical traffic congestion, and more, are well-known to the automobile, but it has no idea what the other road users will be doing at any given time throughout the trip. MDPs, a probabilistic planning method, are effective in mitigating this uncertainty about other road users' behavior.
- C. **“Motion Planning”**: Once the behavior layer has decided the optimum course of action for a certain segment of the route, the motion planning system takes over and controls the car's movements.. The driver and passengers must be able to man oeuvre the vehicle in a comfortable and safe manner. Everything from how fast a car goes to how it changes lanes is dependent on the environment in which it is functioning.
- D. **“Vehicle Control”**: Vehicle control is used to implement the reference route generated by the motion planning system .





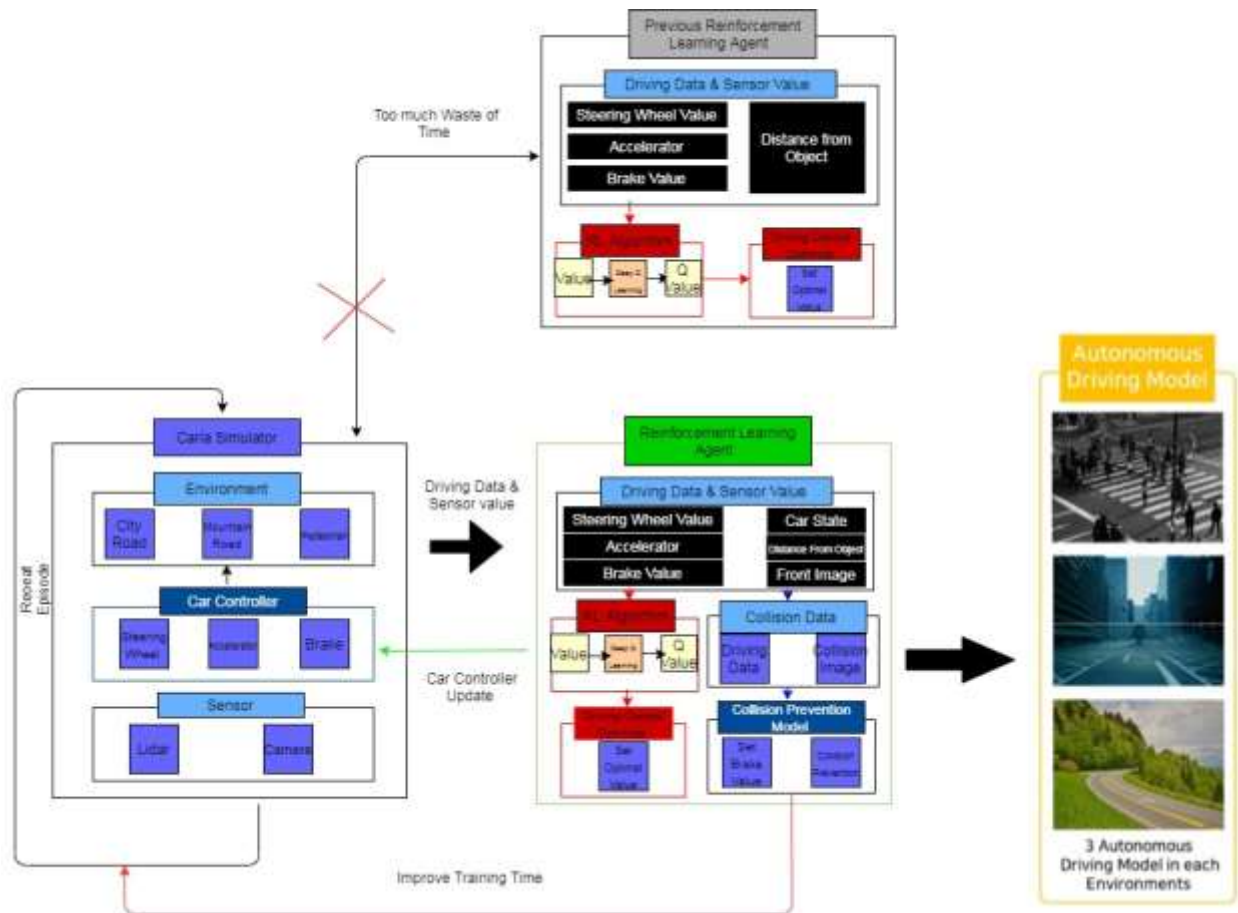
4. Simulator Design for Improving the Performance of Autonomous Driving

4.1. The Proposed Simulation's General Architecture

Using the Carla simulator, this research creates an autonomous vehicle driving and training facility. Carla was used to create a training environment based on roadways for reinforcement learning. Steering wheel,

accelerator pedals and braking pedals were all utilized to control driving in a simulator built using the Python Library in Carla. Input data for the reinforcement learning agent comprised driving data, obstacle distance measured using LiDAR sensors, and films of the driver in action captured by the front camera of the car. The suggested simulation's general flow chart is shown in Figure 2.





5630

Figure 2. Flow chart for the planned simulation in its entirety .

The Carla simulator is used to train the autonomous driving model using a reinforcement learning agent. For reinforcement learning, driving and sensor data are input as "behavior" and "state" parameters, respectively. During this procedure, a Deep-Q-Learning algorithm is used to find optimal driving control settings and to execute optimization.

Crash photographs and driving data are used to train the vehicle's collision avoidance model. To prevent collisions, a model was constructed and updated to boost the brake levels when driving data resembled crash situations . By doing so, the reinforcement

learning process is spared any pointless collisions. It is one of the benefits of this strategy to have less duplicate states in the training process, which saves both time and money in the process of reinforcement learning

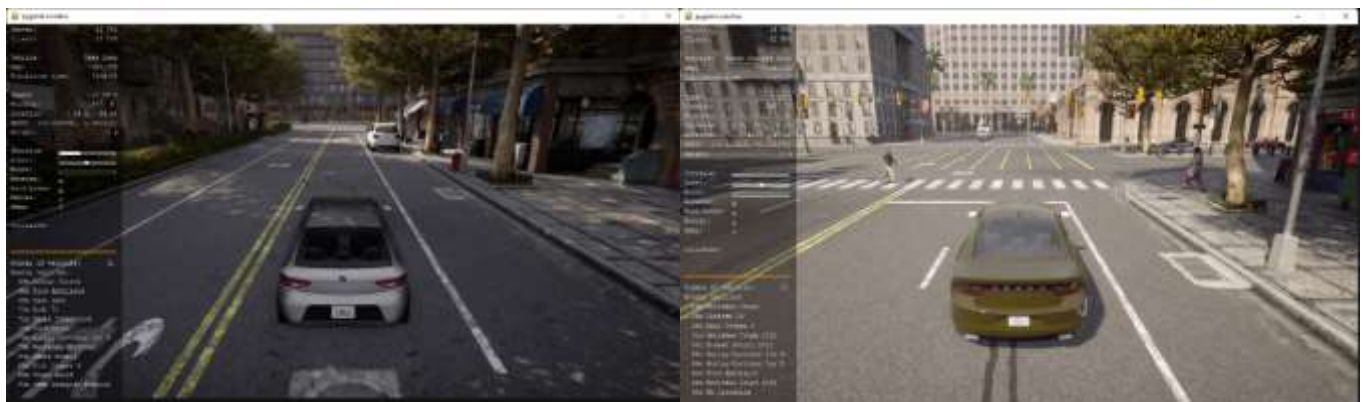
The vehicle control approach is optimized when training for these two models is completed using a reinforcement learning agent. Normally, a car uses a reinforcement learning-based control system to navigate the roads. Reinforcement learning training increases the brake value when the front camera receives visuals that are very comparable to a collision.



4.2. Simulation Configuration

The Carla simulator was used to provide virtual training and testing for the automobile driving simulator. The Unreal Engine's interoperability with the simulator allowed for the creation of scenarios suitable for autonomous driving. It was necessary to create two autonomous driving settings in order to evaluate autonomous driving performance in different contexts. There were two types of

one without pedestrians. In order to save hardware resources and reduce input data volume, only picture camera footage from the front of the vehicle was retrieved utilized as input data. Car collisions, distance from lane markers, and speed were all gathered using the Carla simulator throughout the training phase for a reinforcement learning agent that needed these information. Figure 3 depicts the Carla-based training environment.



5631

urban environments, one with pedestrians and

4.3. Collection of Driving Data

This paper's suggested approach of autonomous driving only employs the pictures captured by the vehicle's data from the vehicle's onboard cameras and the driver's perspective while driving. The Carla simulator's API is used to acquire first-person photos for the front shots. The preprocessed front pictures and driving data were sent into the convolutional neural network employed in the learning model.

Front driving photos were preprocessed using images with 128 by 128 pixel size, which were cropped from 144 x 256 pixel front photographs. Examples are shown in Figure 4. Due to a lack of factors impacting vehicle driving outcomes above the vehicle, forward driving photos were not utilized in their original collecting states, but were instead filtered out during preprocessing.

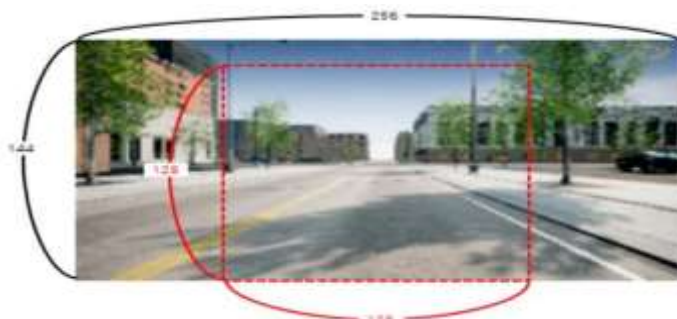


Figure 4. Carla is being used to create a training environment.

Occasionally, a flying or falling item from a structure may impair driving and add to the danger, although these occurrences are very

rare. In a training procedure for reinforcement learning that is repeated several times, it is not necessary to teach unique circumstances. Thus,



our research removed these cases from consideration. Remove superfluous information from the photos, such as objects above a vehicle, and offer track angles to decrease the amount of computation needed for training. Reinforcement learning uses CNN outputs based on the qualities of the input pictures as the state. The agent acts and learns on the basis of their current condition and the rewards they get. Driving straight, making a 45-degree left turn, turning right at 45-degrees and halting are all examples of eight discontinuous actions that may be performed inside an agent's action area. It was important to take into account road conditions and vehicle driving directions. Replay memory was used to capture driving data, such as the speed of the car, the values of the accelerator, brake, steering, and the results of the driving.

4.4. Replay Memory Buffer

Replay memory was used to implement the algorithm's experience replay. Data correlations may be minimized by using DQN's replay memory to save autonomous driving vehicle actions. New values cannot be preserved in a replay memory update when the identical actions were taken in response to a picture from this research's actor network.

Each episode of the reinforcement learning training process, referred to as S , A , R , and s' , updated the memory with new values for the S , A , R , and next state parameters. Results of the driver's exam were also entered into the database. It was necessary to add an index to the driving dataset so that it could be used as a file name for driving outcome data. When fresh data could not be stored due to extensive training, the earliest driving experience data in the memory buffer was kept in the file. The file's driving data was deleted from the memory buffer in order to free up space for new data. As inputs to the CNN neural network, driving images and data were recovered from the replay memory and used to generate reward functions for driving. CNN updates avoided a local minimum by using random multiples of replayed data in the replay memory to limit the

continuous link between the input data. It may be possible to utilize previously recorded driving data that was securely retained throughout the learning process while training under similar or comparable settings. DQN-based reinforcement learning cannot be employed in deep learning applications because of its dependence on episode order. As a consequence of this local minimum, successful learning cannot be guaranteed if experience replay is not used for input data.

4.6. Design of the Collision Prevention Algorithm

There must be a lot of repetition in reinforcement learning. As a consequence of the excessive repeating of crash scenarios, it is impossible to anticipate efficient training. To increase the effectiveness of the training, an algorithm for avoiding front collisions was devised and included into the training stages of reinforcement learning episodes. Using pictures from the front cameras and data from prior reinforcement learning experiences, a collision avoidance system makes predictions about when and where a collision will occur. Crash avoidance and braking are addressed in Algorithm 1. During the reinforcement learning process, collision data is gathered. This information is gleaned from replay memories of the images and driving data captured just before the incident occurred. During an episode, the previously recorded front photos and driving data are utilized to determine the timing of the incident.

A series of images is kept for the time step t_{coll} , which includes 10 frames prior to the collision. An picture collection of accident circumstances called Img_{coll} comprises 10 images and a driving duration of 8 to 9 seconds just before to the crash. At the same time the picture collection is being built, driving data D_{coll} is being acquired. There are three types of driving data in D_{coll} : acceleration, braking, and steering. In order to begin the data filtering process, at least 50 accident images and driving



data must be gathered. Data for the collision prevention model are not appropriate for use if Action brake for braking just before a collision is greater than the threshold. The filtered data is used to construct *Imgcoll*, which excludes any data that has been filtered out. Image tagging is done for *Imgcoll* after the completion of training. Images taken at 5t before the moment of contact bear a high degree of resemblance to the collision and are tagged "collision". A "caution for collision" warning appears on any photographs in which a collision cannot be averted. To develop the classification model, a multiple classification neural network is fed with the labeled images from *Imgcoll*, a collection of 50 collision instances. In order to fine-tune the braking levels based on the collision list outcomes, *ImgRL*, the driving pictures collected during a reinforcement learning event, is categorized using the classification model. As soon as the classification results are [0, 1], it's time to increase the braking force. When the numbers were [1, 0], a warning that a collision was imminent, the brake values were progressively raised to give the player more time to react and prevent a collision. No adjustment is made to the brake levels if the results are [0, 0], which indicates that there is no danger.

To categorize collisions, the CNN-based learning model uses labeled pictures as input data. The Tensorflow library is used to train the CNN model, while Softmax is used as the activation function for multiple classification. Cross Entropy Error is used as the loss function, while Adam is used as the optimizer. Files are used to

store the weights and bias values of a learning model that uses labeled pictures in its training process. An episode of reinforcement learning is used to apply the collision judgment model developed. During episode training, the front cameras continually gather driving pictures, which are then utilized as input data in the collision prevention model to avoid collisions. 'Collision' and 'warning for collision' are the outputs of the input pictures. Braking values are raised when photos are classified as belonging to the "caution for collision" class. To continue collecting images and making collision predictions even when slowing down, a little amount of inertial force keeps the vehicle moving ahead. Even while traveling at a reduced pace, the agent may adjust the vehicle's steering values, which can cause the pictures acquired while driving to change and, as a result, alter the crash prediction values. With a drop in these numbers, the vehicle's braking power is lowered while its accelerator power is boosted, allowing it to accelerate faster than before. Although no real collision occurred, reward values are given based on a judgment that a "collision"-like scenario exists, and the episode concludes and another one begins when using the "collision" class after "caution for collision". The algorithm for preventing collisions is also being trained as part of the reinforcement learning process. A collision avoidance algorithm is updated after 50 collisions by extracting driving data and saving collision photos. Figure 6 depicts the procedure for avoiding a collision and using the brakes.

5633



Algorithm 1. Collision Prevention Model training

Input: $I_{m_{coll}}$, D_{coll} = (Actionaccel, Actionbrake, Actionsteering)
Output: Collision List

Collision time step t_{coll} in $I_{m_{coll}}$
for $i = 1$ to 50 **do**
 $I_{m_{coll}} = I_{m_{coll}} - I_{m_{collision}}(\text{Actionbrake } 0.3)$
 $I_{m_{coll}}(8 - t_{coll} - 5)$ collision label
 $I_{m_{coll}}(8 - t_{coll} - 5)$ collision Warning label
end for
 Multiple Classification Neural Networks model p
 Train classifier: p on $I_{m_{RL}}$
Return Collision List

$I_{m_{coll}}$: 10 timestep images before collision
 D_{coll} : Driving data at $I_{m_{coll}}$

Figure 6: Flow diagram of training for reinforcement learning sessions.



5634

5. Conclusions

A multitude of sensors and camera pictures are used by an autonomous driving system to swiftly analyze road conditions in real time. For example, it should be able to notice rapidly changing road conditions and make immediate judgements about actions like as accelerating, decelerating and stopping. Continuous values of action outcomes are addressed using continuous values of input photos or driving data in autonomous driving. To create an autonomous driving system for automobiles, a reinforcement learning

algorithm was used. Based on trials in the driving simulator Carla, an enhanced autonomous driving model based on reinforcement learning was also presented.

This research proposes an autonomous driving model based on DQN-applied driving data from self-driving cars. It was decided to use replay memory buffers in order to maximize training efficiency and enhance the incentive functions. Images from the cameras of self-driving vehicles were utilized as input material for CNN, a popular tool for analyzing pictures and image data. Reinforcement learning was



shown to take a lengthy time due of the repetition of collision occurrences during training, resulting in a lower production level. A collision avoidance algorithm was used in conjunction with a reinforcement learning training method in the current work to try to find a solution to this problem. In order to speed up the previous reinforcement learning training process and boost autonomous driving capabilities, this algorithm was developed.

Reference

- [1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. "Playing Atari with Deep Reinforcement Learning". *Nature* 518 (2015): 529-533
- [2] S. W. P.J. Antsaklis, K.M. Passino. An introduction to autonomous control systems. 1991.
- [3] Min, K., Kim, H., Huh, K., 2019. Deep distributional reinforcement learning based high level driving policy determination. *IEEE Transactions on Intelligent Vehicles* 4(3), 416 – 424
- [4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.
- [5] X. Liang, T. Wang, L. Yang, and E. P. Xing. CIRL: controllable imitative reinforcement learning for vision-based self-driving. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference*, Munich, Germany, September 8-14, 2018, *Proceedings*, Part VII, volume 11211 of *Lecture Notes in Computer Science*, pages 604–620. Springer, 2018.
- [6] M. Toromanoff, E. Wirbel, and F. Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. *CoRR*, abs/1911.10868, 2019.
- [7] Park, S.; Hwang, K.; Park, H.; Choi, Y.; Park, J. Application of CNN for steering control of autonomous vehicle. In *Proceeding of the Spring Conference of the Korea Institute of information and communication Sciences*, Yeosu, Korea, 20–22 May 2018; pp. 468–469.
- [8] W. T. V. N. H. L. P. M. e. a. Stafford T, Thirkettle M. A novel task for the investigation of action acquisition. 2012.
- [9] M. E. S. D. Berger-Tal O, Nathan J. The exploration-exploitation dilemma: A multidisciplinary framework. 2014.

