2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Human-like Autonomous Vehicle Speed Control by Deep Reinforcement Learning with Double Q-Learning

Yi Zhang[1], Ping Sun[1*], Yuhan Yin[1], Lin Lin[1], Xuesong Wang[2]

***Abstract*—Autonomous driving has become a popular research project. How to control vehicle speed is a core problem in autonomous driving. Automatic decision-making approaches, such as reinforcement learning (RL), have been applied to control the vehicle speed. However, the popular Q-learning algorithm is unstable in some games in the Atari 2600 domain. In this paper, a reinforcement learning approach called Double Q-learning is used to control a vehicle's speed based on the environment constructed by naturalistic driving data. Depending on the concept of the direct perception approach, we propose a new method called integrated perception approach to construct the environment. The input of the model is made up of high dimensional data including road information processed from the video data and the low dimensional data processed from the sensors. During experiment, compared with deep Q-learning algorithm, double deep Q-learning has improvements both in terms of value accuracy and policy quality. Our model's score is 271.73% times that of deep Q-learning.**

Keywords—Autonomous Driving; Deep Reinforcement learning; Double Q-learning; Naturalistic Driving Data

I. Introduction

Nowadays, modern vehicles are already equipped with advanced driver assistance systems, thanks to the current research. Ever since that, we dream of developing a self-driving car by using a single camera mounted on the front of the vehicle. To achieve this, a system capable of learning how to drive autonomously under natural driving data sets.

It is essential for the vehicle to acknowledge its own circumstances and adapt to it in order to drive automatically. This is achieved by the agent-aware methods employed in most of today's autonomous driving systems. There are mainly three paradigms of autonomous driving: the mediation method, the behavior reflection method and the direct perception method.

Mediated perception approaches [1] involve multiple sub-components for recognizing driving-relevant objects, such as lanes, traffic signs, traffic lights, cars, pedestrians, etc. [2]. For example, there are components on the subject of the visual detection of lane markings [3, 4], cars [5, 6], pedestrians [7, 8], traffic signs [9, 10], traffic lights [11, 12] or combinations of the aforementioned [13, 14]. Other non-camera sensors, like LiDAR or radar, are also used in some systems for obstacle detection [15, 16]. The recognition results are then combined into a consistent world representation of the car's immediate surroundings. Since only a small portion of the detected objects are indeed relevant to driving decisions, this level of total scene understanding may add unnecessary complexity to an complex task. While other robotic tasks' final output space resides in a very low dimension, mediated perception computes a high-dimensional world representation, probably including redundant information. The complexity and the cost of a system is unnecessarily increased when facing those open challenges for general scene understanding in order to solve the simpler car-controlling problem.

Behavior reflex approaches construct a direct mapping from the sensory input to a driving action. This idea dates back to the late 1980s [17, 18] using a neural network to construct a direct mapping from an image to steering angles. To build the model, a human is needed to drive the car along the road while the system records the images and steering angles as the training data. Recently NVIDIA [19] was able to successfully train a ConvNet to map raw camera images to steering angles. In [31], an end-to-end FCN-LSTM(fully-convolutional network-long short-term memory) network is trained to predict multi-modal discrete and continuous driving behaviors.

Although this idea is very elegant, it may struggle to deal with traffic and complicated driving maneuvers for several reasons. Firstly, with other cars on the road, even when the input images are similar, different human drivers may make completely different decisions, which results in an ill-posed problem that is confusing when training a regressor. Secondly, the decision-making for behavior reflex is low-level. The direct mapping cannot see a bigger picture of the situation. Finally, because the input to the model is the whole image, the learning algorithm must determine which parts of the image are relevant. However, the level of supervision to train a behavior reflex model, i.e. the steering angle, may be too weak to force the algorithm to learn this critical information.

In [20] a direct perception approach is proposed. Instead of using detections of lane markings and other objects to derive requirements for the vehicle's behavior indirectly, a set of affordances for driving actions is defined. Those affordances include the heading of the vehicle, the distances to surrounding lane markings and preceding cars. A ConvNet is trained to extract the defined affordances directly from the visual sensor input. Steering commands and the decision to overtake preceding cars are issued by a simple controller based on the

[*]Research supported by the Chinese National Science Foundation (No. 51522810), the Science and Technology Commission of Shanghai Municipality (No. 15DZ1204800), the 111 Project (No. B17032)

[1] Yi Zhang, Ping Sun, Yuhan Yin, Lin Lin are with college of Software Engineering, Tongji University (corresponding author to provide; e-mail: pingsun@tongji.edu.cn).

[2] Xuesong Wang is with Key Laboratory of Road and Traffic Engineering, Ministry of Education, National Engineering Laboratory for Integrated Optimization of Road Traffic and Safety Analysis Technologies, Tongji University (wangxs@tongji.edu.cn)

affordances' values. While direct perception reduces the complexity of the environment model, the affordances and the vehicle's controller are hand-crafted.

Based on this approach, we propose a new approach called integrated perception approach. High dimensional video data is processed into meaningful information. These information mainly includes the distances to surrounding lane markings. But they are fused together with the low dimensional data from the sensors as the input of the model. It effectively reduces the complexity of the environment model,

However, these methods mostly fail to produce human-like behaviors. Modeling the human driving behavior becomes even more complicated when considering scenarios such as driving in large cities with many intersections. Therefore, applying machine learning methods to extract the models directly from the expert demonstrations appears more promising.

Still, they might fail when facing scenarios that are very different from the ones in the training data. A more promising formulation for this problem is based on Markov Decision Processes (MDPs). Recent breakthroughs of deep reinforcement learning (DRL) in AlphaGo and playing Atari set a good example in handling large state space of complicated control problems and could be potentially utilized to solve the auto-driving problem.

Driving solely is based on the visual input in the open racing simulation TORCS, in [21] and [22] Koutn´ık et al. An evolutionary reinforcement learning approach is employed during training recurrent neural networks and ConvNets. The networks evolved based on a fitness function are used to perceive the car's environment and control the car. Mnih et al. let their system learn to play a wide range of Atari 2600 games on or above human level with their proposed deep Q-network (DQN) [23, 24]. The DQN has powerful generalizability and various extensions have been published since the inception, ranging from a continuous action space [25]. Asynchronous agents [26] and prioritizing experience replay [27] to extended network architectures and to separate state values and action advantages [28]. While most of these DQN approaches are evaluated on an Atari 2600 benchmark [29], [25] and [26] also successfully train the DQN to drive a vehicle in TORCS.

Although the popular Q-learning algorithm is known to estimate action values under certain conditions, [30] finds that the DQN algorithm suffers from substantial overestimations in some games in the Atari 2600 domain. Based on Double Q-learning algorithm, which can be generalized to work with large-scale function approximation, they propose a specific adaptation to the DQN algorithm. Comparing with deep Q-learning, this resulting algorithm not only yields more accurate value estimates, but leads to much higher scores on several games.

In reference to both Double Q-learning and DQN, we refer to the resulting learning algorithm as Double DQN. In this paper, we use double DQN to build the vehicle speed model. By approximating this function rather than directly learning the state-action pairs in a supervised fashion, one can handle new scenarios better.

Furthermore, naturalistic driving data is used to construct the model's environment. Naturalistic driving data records the process of driving via data acquisition techniques in natural state of normal driving condition without interference and experimenters. In this paper, by using integrated method to process natural driving data into environmental input

In this paper, by using natural driving data as environmental input, as well as machine learning model algorithm to learn human's decision-making and judging ability, we are able to achieve a human-like learning effect. Human-like behavior appears more promising when dealing with changeable real environment.

The remainder of this paper is structured as follows: In Section II we give an overview about our system's framework and approaches to construct the environment. Section III an overview about deep reinforcement learning and double Q-learning algorithm and modelling of our system. It includes a detailed description of our reward concept, action space and neural networks. Our results is shown in Section IV. Section V concludes the whole paper.

II. FRAMEWORK AND ENVIRONMENT

A. Framework

In this paper, our system's framework is an agent-environment interaction system. As shown in Fig.1, the general agent-environment interaction modeling (of both traditional RL and the emerging DRL) consists of an agent, an environment, a finite state space S, a set of available actions A, and a reward function: S×A → R. The decision maker is called the agent, and should be trained as the interaction system runs. The agent needs to interact with the outside, which is called the environment. The interaction between the agent and the environment is a continual process. At each decision epoch k, the agent will make decision a_k based on the current state s_k of the environment. Once the decision is being made, the environment would receive the decision and make corresponding changes, and the updated new state s_{k+1} of the environment would be presented to the agent for making future decisions. The environment also provides reward r_k to the agent depending on the decision a_k, and the agent tries to maximize some notion of the cumulative rewards over time. This simple reward feedback mechanism is required for the agent to learn its optimal behavior and policy.

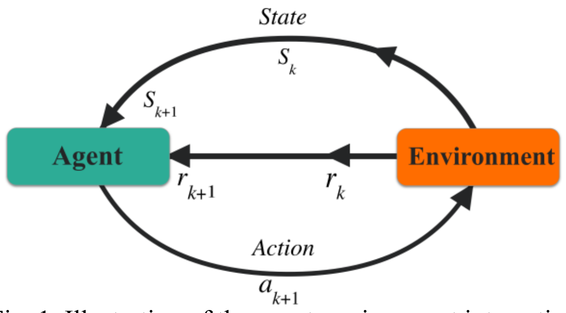

Fig. 1. Illustration of the agent-environment interaction system.

Fig.2 show the architecture of our system. According to the concept of direct perception approach, high dimensional video data are processed into meaningful data about road. But they are fused with low dimensional data from the sensors to construct the environment. Given integrated data processed from naturalistic driving dataset, the Double DQN calculates Q-value estimates for speed control actions. Actions contain acceleration, deceleration and maintaining. The speed control action with the highest Q-value is chosen and executed by the model.

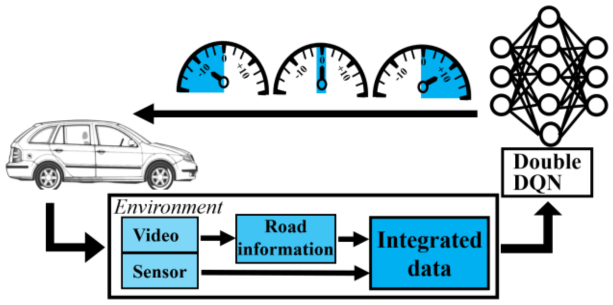

Fig. 2: Architecture of our system

B. Environment

When constructing the environment, it is based on data from the Shanghai Naturalistic Driving Study cooperated between Tongji University and an automotive company. It is the first natural driving study in China. During data collection, drivers involved in the experiment should meet the following conditions: (1) Hold at least 2 years of driving experience. (2) Aged 25 to 60. (3) Drive vehicles every day. (4) Not a professional driver or a taxi driver.

One major characteristic of naturalistic driving data is that it is collected in natural state of normal driving condition without interference and experimenters. Under environment constructed by naturalistic driving data, reinforcement learning algorithm could learn human's decision-making and judging ability. Human-like behavior appears more promising when dealing with changeable real environment.

The data acquisition system includes a data interface of vehicle bus, an accelerometer with three axes, a LIDAR system which can track 8 targets, GPS system which can fix position and a camera with four directions. Figure.3 shows the sensors installed in the vehicle and data's process flow.

Basing the concept of integrated perception approach, the environment is constructed by meaningful road information data processed from the high dimensional video data. But it also includes the low dimensional sensor data. Figure.3 also shows the variables of the road information data and senor data. Among the road information variables, the judgment of the road width is the distance from the inner edge of the innermost lane of the road to the outer edge of the lane mark on the outermost side of the road. As for the lane offset value, it is the distance from the centerline of the car to the centerline of the driveway. In addition, lane line type will be classified into three categories which are solid line and dotted line and unsure as well. Sensor data could be mainly classified into position and acceleration data. Lidar system records the tracked targets' relative position and velocity in two vertical directions.

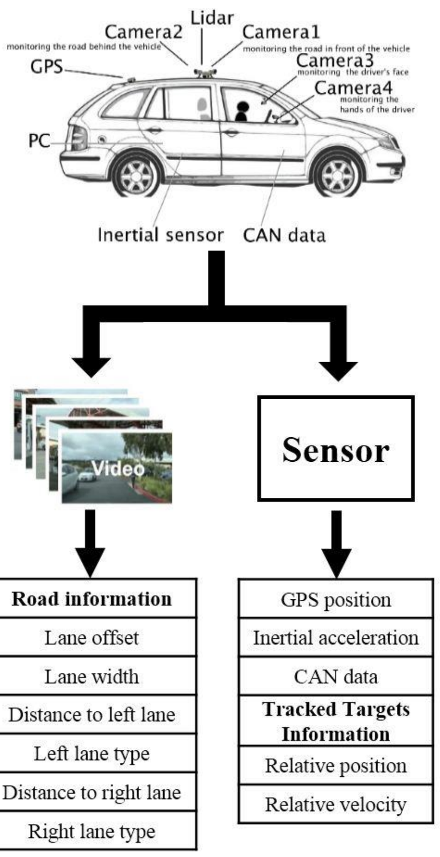

Fig. 3: Illustration of integrated perception approach.

III. DOUBLE DEEP Q LEARNING -BASED CONTROLLER

A. Overview of Deep Reinforcement Learning

A generalized form of deep reinforcement learning (DRL) technique was presented in this part. The deep reinforcement learning technique is comprised of an offline DNN construction phase and an online deep Q-learning phase. The offline phase adopts DNN to derive the correlation between each state-action pair (s, a) of the system under control and its value function Q(s, a). The offline DNN construction phase needs to accumulate enough samples of Q(s, a) value estimates and corresponding (s, a) for constructing an enough accurate DNN, which can be a model-based procedure or from actual measurement data.

Arbitrary policy and gradually refined policy can be applied in this procedure. The transition profiles are stored in an experience memory D with capacity N_D . The use of experience memory can smooth out learning and avoid oscillations or divergence in the parameters. Basing on the stored state transition profiles and Q(s, a) value estimates, the DNN is constructed with weight set θ trained using standard training algorithms. The key steps in the offline procedure was summarized in Algorithm 1.

During the algorithm 1, if the model have achieve an ideal quality during accuracy evaluation, it doesn't need iterations for procedure one and two anymore. Otherwise, it will continue iterations.

Algorithm 1 Illustration of the General DRL Framework

1. Extract real data profiles using certain control policies and obtain the corresponding state transition profiles and Q(s, a) value estimates;
2. Store the state transition profiles and Q(s, a) value estimates in experience memory D with capacity N_D;
3. Iterations may be needed in the above procedure;
4. Pre-train a DNN with features (s, a) and outcome Q(s, a);

B. *Overview of Double deep Q-learning algorithm*

The Double DQN algorithm is adopted for the online control based on the offline-trained DNN. It is investigated in [30], which uses the existing architecture and deep neural network of the DQN algorithm without requiring additional networks or parameters. The idea of Double Q-learning is to decompose the max operation in the target into action selection and action evaluation. Different from the DQN algorithm, there are two networks with the same structure.

Although not fully decoupled, the target network in the DQN architecture provides a natural candidate for the second value function, without having to introduce additional networks. Double DQN evaluates the greedy policy according to the online network, but using the target network to estimate its value. The update used by Double DQN is the same as for DQN, but replacing the target value with

$$Y_k = r_{k+1} + \gamma Q(S_{k+1}, argmaxQ(S_{k+1}, a; \theta_k), \theta_k^-) \quad (1)$$

The weights of the second network θ_k are replaced with the weights of the target network θ_k^- for the evaluation of the current greedy policy. The update to the target network stays unchanged from DQN, and remains a periodic copy of the online network. And the exact same hyper-parameters as for DQN is used in Double DQN.

Double Q-learning can be used at scale to successfully reduce this DQN's unstable problem, resulting in more stable and reliable learning. Double DQN has improvements both in terms of value accuracy and in terms of policy quality.

C. *Double deep Q-learning-based Vehicle speed controller*

In the Double DQN-based speed control model, the speed is controlled by the Double DQN agent. Integrated data processed from naturalist driving dataset constructs the driving environment. The Double DQN-based speed control framework is discrete time based and event driven, in that the agent selects an action at the time of each environment data arrival.

In detail, at each decision epoch t_k of action controlling sequence, the whole system is under control in a state s_k. The agent performs inference using the DNN to derive the $Q(s_k, a)$ estimation of each state-action pair (s_k, a), and uses ϵ −greedy policy to derive the action with the highest $Q(s_k, a)$ with probability $1 - \epsilon$ and choose the other actions randomly with total probability ϵ. The chosen action is denoted by a_k. At the next decision epoch t_{k+1}, the DRL agent performs Q-value updating based on the total reward r_k observed during this time period $[t_k, t_{k+1})$.

Then the weights of the second network θ_k are replaced with the weights of the target network θ_k^- for the evaluation of the current greedy policy. At the end of the execution sequence, The DRL agent also updates the weights of target network θ_k^- using the newly observed Q-value estimates. And these updated networks will be utilized in the next execution sequence. The whole procedures are shown in Algorithm 2.

Algorithm 2 Illustration of the Double deep Q-learning-based Vehicle speed controller

```
1:Initialize replay memory D to capacity N
2:Initialize action-value function Q with random weights
3: for each execution sequence do
4:   for at each decision epoch $t_k$ do
5:   With probability $\epsilon$ select a random action, otherwise $a_k = argmax_a Q(s_k, a)$, in which $Q(s_k, a)$ is estimated from DNN;
6:   Perform vehicle speed control $a_k$;
7:   Observe state transition at next decision epoch $t_{k+1}$ with new state $s_{k+1}$, receive reward $r_k$ during time period $[t_k, t_{k+1})$;
8:   Store transition ($s_k$, $a_k$, $r_k$, $s_{k+1}$) in D;
9:   Replace the second network's weights $\theta_k$ with the weights of the target network $\theta_k^-$
10:  If terminal $s_{k+1}$: $Y_k = r_k$
     else $Y_k = r_k + \gamma Q(S_{k+1}, argmaxQ(S_{k+1}, a; \theta_k), \theta_k^-)$
11:  Perform a gradient descent step on
     $\sqrt{1 + (y_k - Q(S_k, argmax_{a'} Q(S_k, a; \theta_k), \theta_k^-))^2} - 1$
12:  end for
13: end for
```

The action space, reward function and DNN constructions of the Double DQN based speed control model are defined as follows:

Action space: The action of the Double DQN agent for speed is defined as speed selections. These selections contains acceleration, deceleration, maintaining. The action space for speed control is defined as follows.

$$A = \{a | a \in \{\text{Acceleration}, \text{Deceleration}, \text{Maintaining}\}\} \quad (2)$$

In the formula, the numbers in the set correspond to acceleration, deceleration, maintaining. It can be observed that the action space is significantly reduced by using an event-driven and DRL-based decision framework. The DRL framework requires a relatively low-dimensional action space because in each decision epoch the DRL agent needs to enumerate all possible actions at current state and perform inference using DNN to derive the optimal Q(s, a) value estimate, which implies that the action space needs to be reduced.

Reward network: Reward is necessary in almost all reinforcement learning algorithms offering the goal of the reinforcement learning agent. The reward estimates how good the agent performs an action in a given state (or what are the good or bad things for the agent). In this paper, we design a reward network to map each state to a scalar, a reward, expressing the intrinsic expectation of the state.

In this paper, the Double DQN algorithm was designed by judging the action of series of the vehicle. In the auto-driving task, drivers need to drive correctly. If the agent gives actions of opposite direction compared with the ideal state-action, the action is called A^{-}. For instance, if agent is supposed to accelerate, however, it decelerates instead. The agent will receive a negative reward. Additionally, the opposite driving behavior can lead to more serious driving accidents and this reward is defined as the most negative,

If the agent don't give an action and keep still, it will be accord with the ideal action as well, which is called B^{-}. It will receive a fairly small negative reward. Although it is not the most optimal for vehicles to remain stationary, the opposite acceleration leads to more serious consequences. Thus its reward is fairly small compared with action A^{-}. If the agent's decision is accord with the ideal action perfectly, this action is called C^{-}. It will receive the greatest positive reward so as to reinforce the state-action pair and learn human behavior.

Furthermore, we introduce a modifying behavior. During real driving process, modifying behavior is one of important method for drivers. Generally, drivers will make decisions according to their experiences. They will also make modifying behavior when making mistakes. For example, when drivers make acceleration decision, unfortunately, they find there are possibilities to accidents. They will make decelerate motion to compensate their mistakes. In same way, if the agent makes mistakes at first, but modify its behaviors into the opposite direction in the following episodes, it will receive a positive reward as well. Its reward is still smaller than action C^{-} and this action is called D^{-}.

The reward network is designed through the above reward work in this paper. When:

$$f(x) = \begin{cases} +2 & (x,a) \in \{C^{-}\} \\ +1 & (x,a) \in \{D^{-}\} \\ 0 & else \\ -1 & (x,a) \in \{B^{-}\} \\ -2 & (x,a) \in \{A^{-}\} \end{cases} \quad (3)$$

DNN Construction: In this paper, due to Double DQN algorithm's estimation theory, two feed-forward neural network with the same network structure are constructed to output Q value estimates. This works well with problems with relatively small state and action spaces (e.g., in [23], the number of actions ranges from 2 to 18).

When design this neural network's structure, experiment method is mainly used. Different activation and hidden layers is tested. Finally, a three layers fully connected neural network is built. Its input is the environment information and the Environment Predictor's outputs data. The input and hidden layers' activation function is RELU. The outputs layers' activation function is Linear. Loss function is defined as follows:

$$Loss = \sqrt{1 + (y_k - Q(S_k, argmax_{a'} Q(S_k, a; \theta_k), \theta_k^{-}))^2} - 1 \quad (4)$$

IV. EXPERIMENT

A. Experiment Setups

We trained the model by letting the vehicle agent drive on the data recorded in the real environment. A Quadro M4000 GPU is used to assist accelerating training phase. The parameters are fixed as following: discount rate=0.95, initial =0.5, final=0.1, replay size = 128. The maximum memory is set as 2000. The behavior policy during training was ε-greedy with annealed linearly from 1 to 0.01 and fixed at 0.01 thereafter. We trained our model using three different gradient-based optimization methods to examine impacts on the training of our model: a stochastic gradient descent (SGD), Adam and RMSProp algorithm.

Fig.4 presents the track of average reward along the training process. There are reward drop the early stage of training process. This is due to the negative reward when the agent make incorrect actions. These drops disappeared after a certain number of training iteration since the agent learned from these penalties and avoids such actions in the later stage of training. This is also a convincing evidence of performance improvement from learning from the historical driving records.

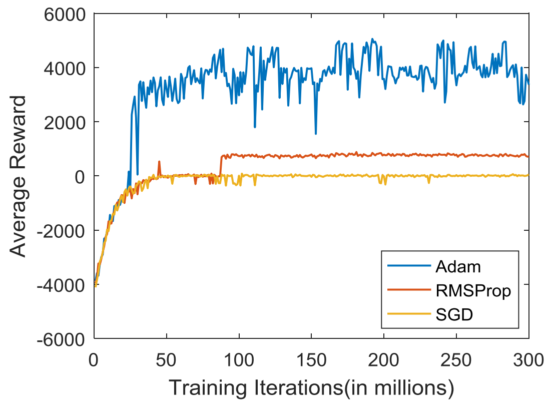

Fig.4. Track of reward under different optimization methods

As shown in Fig.4, it is quite clear that the reward increasing along the training process. During it, the Adam solver shows the greatest performance. The results demonstrates that the Adam solver achieve the highest average reward and become more stable with continued training. We conclude that the evaluated approaches are suitable for our Double DQN and the performance differences are not significant over a longer training period.

B. Performance Comparison

For evaluation purposes, we compared Double DQN algorithm with DQN algorithm. The performance difference is also presented in Fig.5. The Fig.5 shows the detrimental effect of this on the score achieved by the agent as it is evaluated during training. Learning with Double DQN is much more stable. It can be seen that Double DQN algorithm's performance is much better than the DQN algorithm. After calculation, the total score of Double DQN is 271.73% of

DQN. Double DQN has improvements both in terms of value accuracy and policy quality.

Double DQN appears more robust to this more challenging evaluation, suggesting that appropriate generalizations occur and that the found solutions do not exploit the determinism of the environments. This is appealing, as it indicates progress towards finding general solutions rather than a deterministic sequence of steps that would be less robust.

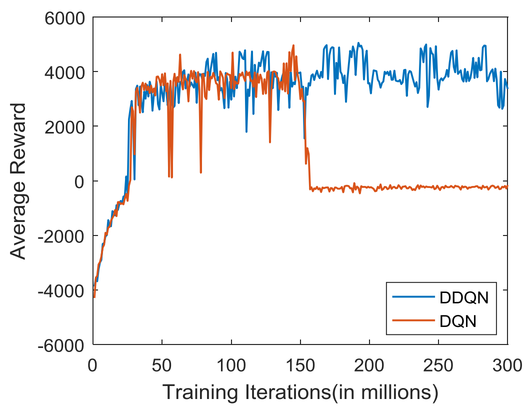

Fig.5. Track of reward under different algorithm

V. CONCLUSIONS

In this paper, a deep reinforcement learning based vehicle speed control system is designed and trained with natural driving data from a real-world commute trip in Shanghai. The proposed model combines a Double Q-learning and deep neural networks to form a deep Q-network which is capable of learning and providing the optimal control decisions in continuous environment and actions states. The evaluation with real-world trip data shows that Double Q-learning can be used at scale to successfully reduce this DQN's unstable problem, resulting in more stable and reliable learning. Double DQN has improvements both in terms of value accuracy and in terms of policy quality. Our model's score is 271.73% times that of DQN. The future work would be focused on the further testing on vehicle platforms with more real-world driving data.

REFERENCES

[1] S. Ullman. "Against direct perception". Behavioral and Brain Sciences, vol.3,no.03,pp. 373–381, 1980.

[2] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "Vision meets robotics: The kitti dataset". The International Journal of Robotics Research, 2013.

[3] Y.-W. Seo and R. R. Rajkumar, "Utilizing instantaneous driving direction for enhancing lane-marking detection," in Intelligent Vehicles Symposium Proceedings, IEEE, pp. 170–175, 2014

[4] J. Huang, H. Liang, Z. Wang, "Robust lane marking detection under different road conditions," in Robotics and Biomimetics (ROBIO), IEEE, pp. 1753–1758, 2013

[5] L. C. León and R. H. Jr., "Car detection in sequences of images of urban environments using mixture of deformable part models," Pattern Recognition Letters, vol. 39, pp. 39–51, 2014.

[6] C. M. Bautista, C. A. Dy, M. I. Ma` yalac, "Convolutional neural network for vehicle detection in low resolution traffic videos," in Region 10 Symposium(TENSYMP), IEEE, pp. 277–281, 2016.

[7] S. Zhang, R. Benenson, M. Omran, "How far are we from solving pedestrian detection?" arXiv preprint arXiv: 1602.01237, 2016.

[8] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "Pedestrian detection with spatially pooled features and structured ensemble learning," IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 6, pp. 1243–1257, 2016.

[9] S. Houben, J. Stallkamp, J. Salmen, "Detection of traffic signs in real-world images: the german traffic sign detection benchmark," in Neural Networks (IJCNN), The 2013 International Joint Conference on, IEEE, pp. 1–8,2013.

[10] S. K. Berkaya, H. Gunduz, O. Ozsen, "On circular traffic sign detection and recognition," Expert Systems with Applications, vol. 48, pp. 67–75, 2016.

[11] N. Fairfield and C. Urmson, "Traffic light mapping and detection," in Robotics and Automation (ICRA), IEEE,, pp. 5421–5426,2011.

[12] M. Weber, P. Wolf, and J. M. Zöllner, "Deeptlr: a single deep convolutional network for detection and classification of traffic lights," in Intelligent Vehicles Symposium (IV), IEEE, pp. 342–348, 2016.

[13] B. Huval, T. Wang, S. Tandon, "An empirical evaluation of deep learning on highway driving," arXiv preprint arXiv: 1504.01716, 2015.

[14] H. Yu, Y. Yuan, Y. Guo, "Vision-based lane marking detection and moving vehicle detection," in Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2016 8th International Conference on, IEEE, pp. 574–577,2016.

[15] A. Asvadi, C. Premebida, P. Peixoto, "3d lidar-based static and moving obstacle detection in driving environments: an approach based on voxels and multiregion ground planes," Robotics and Autonomous Systems, vol. 83, pp. 299–311, 2016.

[16] P. Amaradi, N. Sriramoju, L. Dang, "Lane following and obstacle detection techniques in autonomous driving vehicles," in Electro Information Technology (EIT), IEEE, pp. 674–679, 2016.

[17] Pomerleau, Dean A. ALVINN,"An autonomous land vehicle in a neural network". Advances in neural information processing systems 1. Morgan Kaufmann Publishers Inc, pp. 305-313 ,1989.

[18] Pomerleau, Dean A. "Neural Network Perception for Mobile Robot Guidance." Springer International 239,1992.

[19] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316, 2016

[20] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. "Deepdriving: Learning affordance for direct perception in autonomous driving". In Proceedings of the IEEE International Conference on Computer Vision, pp. 2722–2730, 2015.

[21] J. Koutnik, G. Cuccu, J. Schmidhuber, et al., "Evolving large-scale neural networks for vision-based reinforcement learning." Machine Learning, pp.541-548. 2013.

[22] J. Koutn´ık, J. Schmidhuber, and F. Gomez, "Evolving deep unsupervised convolutional networks for visionbased reinforcement learning," Conference on Genetic and Evolutionary computation - GECCO '14, pp. 541– 548, 2014.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Humanlevel control through deep reinforcement learning," Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al., "Continuous control with deep reinforcement learning," arXiv preprint arXiv:1509.02971, pp. 1–14, 2015.

[26] V. Mnih, A. P. Badia, M. Mirza, et al., "Asynchronous Methods for Deep Reinforcement Learning," arXiv, pp. 1–28, 2016.

[27] T. Schaul, J. Quan, I. Antonoglou, et al., "Prioritized Experience Replay," arXiv, 2015.

[28] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling Network Architectures for Deep Reinforcement Learning," arXiv, 2016.

[29] M. G. Bellemare, Y. Naddaf, J. Veness, et al., "The arcade learning environment: an evaluation platform for general agents," Journal of Artificial Intelligence Research, vol. 47, pp. 253–279, Jun. 2013.

[30] Hasselt H V, Guez A, Silver D., "Deep Reinforcement Learning with Double Q-learning". arXiv:1509.06461 [cs], 2015.

[31] Xu, Huazhe, et al. "End-to-End Learning of Driving Models from Large-Scale Video Datasets."in Computer Vision and Pattern Recognition IEEE,pp. 3530-3538,2017.