Multi-objective Optimization Based Deep Reinforcement Learning for Autonomous Driving Policy

Hao Wang\textsuperscript{1}, Zhongli Wang\textsuperscript{1,*}, Xin Cui\textsuperscript{2}

\textsuperscript{1}School of electronic information engineering, Beijing Jiaotong University, Beijing 100044, China
\textsuperscript{2}China Railway Electrification Bureau Group Co, Ltd, Beijing 100036, China
\textsuperscript{*}Corresponding author’s email: zlwang@bjtu.edu.cn

Abstract: End-to-end autonomous driving approach seeks to solve the problems of perception, decision and control in an integrated way, which can better adapt to the new traffic scene. Due to the diversity of traffic scenes and the uncertainty of the interaction among surrounding vehicles, the design of autonomous driving policy is challenging. Many current methods manually design the corresponding driving policy for different traffic scene, resulting in suboptimal solutions and the maintaining is hard. Most of the existing deep reinforcement learning (DRL) methods can't work well in the complex urban traffic scenes because of the sensing and simple driving policy. In this paper, to extend the adaptability of the SAC-based method, we proposed to take the multiple sensor data as input, and a VAE network was used to enhance the quality of training data for SAC-based DRL method. A multi-constraints reward function for SAC-based driving policy training is designed, which account for the errors of transverse distance, longitudinal distance, heading, velocity and the possibility of collision. The multiple sensor data include the original RGB image captured by forward-view camera, a 3D LiDAR and the bird's-eye view map resulted from the perception processing, the mixed inputs could enrich the capability of scene perception. The proposed approach is validated with the multi-vehicle traffic simulation built with CARLA\textsuperscript{[1]}. The results showed that the simulated vehicle could adapt to more challenging traffic scenes, like passing intersections and turning in crowded urban scenes, etc with the driving policy generated by the proposed method, and its performances are obviously outperformed against the existing similar methods.

1. Introduction
The core technologies of autonomous driving include positioning, environment perception, prediction, planning or decision making, and control. Among them, planning or decision making uses inputs from sensors, prediction information and high-definition maps to realize driving decisions of vehicles in various traffic scenarios, such as turning, accelerating or decelerating, changing lanes or overtake, etc. A good driving policy can determine the corresponding driving behaviour based on the results of environmental perception and the status, ensuring safety, comfortable and smooth vehicle operation, and even responding to some unexpected situations. The design of robust and adaptable driving policy for complex urban traffic conditions is challenging, although there have been many research results \cite{2}, it is still a problem that has not been solved perfectly.

Recently, reinforcement learning algorithms have been used to solve the problem of driving policy or driving behaviour for unmanned vehicles. Lange \cite{3} used deep encoders to learn potential features from the input raw images and used reinforcement learning to successfully learn maneuvers such as
steering, acceleration and braking. Yu. A[4] used the DQN algorithm for automatic vehicle control with different reward functions to generate specific driving behaviours to implement a vehicle turning task in a simulated environment. Lillicrap [5] developed Deep Deterministic Policy Gradient (DDPG) to solve the continuous control problem and the method implemented a control policy for lane keeping in a simulated driving environment. Most of them use a single sensor data, the extremely complex high-dimensional visual features greatly increase the complexity of the learning samples and the training effect is not good.

Fewer existing deep reinforcement learning-based methods have been successfully applied in complex scenarios, and most research is still using basic deep learning algorithms such as DQN. The performance on training with high-dimensional data is poor. Some new deep reinforcement learning algorithms, such as DDPG and SAC presented in reference [6], which have strong exploration capabilities and high convergence efficiency, are less frequently used for end-to-end autonomous driving tasks.

In this paper, we make full use of the input information from different sensors, including front view RGB image, LiDAR data and BEV (bird's-eye view) map, which can provide more comprehensive environmental information for the reinforcement learning module and help to learn more reliable driving policy. However, due to the existences of some irrelevant information in the multi-source input information, such as lighting and texture information, latent features are extracted from the multiple complex raw data, which reduces the complexity of the input data without losing the valid information. In the driving policy learning module, a model-free reinforcement learning algorithm based on multi-objective optimization reward and SAC is elaborately designed to complete the learning of self-driving policy in complex urban traffic scenarios. Simulation experimental results show that the proposed method can not only automatically generate driving policy for a variety of typical traffic scenarios (lane keeping, passing intersections, turning, etc.), but also the driving policy generated by the method in this paper have more advantages in terms of performance metrics compared to learning methods that can only be implemented for a single scenario.

2. The Proposed Method
The block diagram of the proposed method in this paper is shown in Figure 1. The input of the algorithm includes the raw data of the LiDAR point cloud and the front view RGB image. It is difficult to train a good driving policy using only the raw data. The states of the surrounding vehicles, such as position, speed, and information about the current lane, the deviation distance of the vehicle's position from the target lane, the reference path, etc., are converted into a BEV map as supplementary information. To reduce the irrelevant information in the multiple input resource, latent features are extracted with a VAE network to improve the training efficiency. In the policy learning part, a model-free reinforcement learning algorithm based on multi-objective optimization reward and SAC is designed. A CARLA-based simulator is used to imitate the vehicle's behaviour and generate the raw sensor data.

2.1. Bird's-eye view and coding network
In CARLA simulation environment, the vehicle is equipped with front view RGB image and LiDAR sensors. The point cloud acquired by LiDAR is rendered into 2D LiDAR image. The results obtained by target detection and the road information is used to generate BEV map as in reference [7], which is an overhead view of the road situation and contains the features useful for training. All sensor data are updated in real time while the vehicle is in motion. BEV map consists of four parts, which include the driving map, reference path of the controlled vehicle and its attitude, other vehicles in the environment.

If the BEV map and raw sensor data are directly used as inputs for training and learning, the learning is costly and inefficient, since the raw data contains many irrelevant information for training, it may lead to overfit during model training.

To reduce the complexity of the input, the raw sensor information and the BEV map are encoded with a variational autoencoder (VAE) which is also used in the literature [8]. The input of the encoder is the observation data $x$ from the sensor, encoder network model $q_{\theta}(z \mid x)$ and the decoder network model $P_{\phi}(x \mid z)$, in order to make the and these two networks approximate gradually, from the method of literature. The network parameters $\theta, \phi$ are obtained by maximizing the divergence between $P_{\phi}(z \mid x)$ and $q_{\theta}(z \mid x)$, with the optimization objective of

$$L(\phi, \theta, x^{(i)}) = D_{KL}(q_{\theta}(z \mid x^{(i)}) \parallel P_{\phi}(z)) - E_{q_{\theta}(z \mid x^{(i)})}(\log P_{\phi}(z \mid x^{(i)}))$$

(1)

$D_{KL}$ is the Kullback-Leibler (KL) divergence. The encoded low-dimensional latent variables retain the key information of the original input data.

2.2. SAC-based modelling
SAC-based method relies on maximizing entropy for reinforcement learning, and the output policy considers both reward and entropy terms. The algorithm is chosen to balance stability, sample efficiency and exploration ability while still successfully completing the given task.

To handle the continuous action space, the $Q$ network and the policy network are each fitted by a parameter-controlled neural network, where the network is defined $Q_{\phi}(s, a)$ and the policy network $\pi_{\phi}(\cdot \mid s)$. The target policy under the maximized entropy is defined as:

$$\pi^* = \arg \max_{\pi} E_{(s, a) \sim \rho}[\sum_{t} R(s_t, a_t) + \alpha H(\log \pi_{\phi}(s_t | s)))]$$

(2)

The objective function of the $Q$ function minimizes the residuals using MSE as:

$$J_{Q}(\theta) = E_{(s, a, r, s') \sim \rho}[\frac{1}{2}(Q_{\theta}(s_t, a_t) - r_t(s_t, a_t) + \gamma Q_{\phi}(s_{t+1}, a_{t+1}) + \gamma \alpha \log \pi_{\phi}(a_{t+1} | s_{t+1}))^2]$$

(3)

The states $s$ are sampled from the replay buffer $D$ and the actions $a$ are computed from the current policy.

$$J_{\pi}(\phi) = E_{s \sim D, a \sim \pi_{\phi}}[\log \pi_{\phi}(a \mid s) - Q_{\phi}(s, a)]$$

(4)

According policy network, it outputs a continuous action space to control the vehicle motion in the simulation environment. It is necessary to train the policy network $\pi$ and the $Q$ network.
2.3. Multi-objective optimization reward function

For the design of a driving policy networks, different reward functions need to be considered for driving scenarios. The control actions for the vehicle include the steering and throttle. In the training process, a reward function controller is designed, which can adapt to a variety of driving scenarios. The corresponding action space is defined as follows:

\[ A = \{\delta, \tau\} \]  

The steering angle and throttle are normalized to the ranges of [-1,1] and [0,1], respectively, and the brake value is within [0,1] in CARLA simulator. The method in this paper mainly controls the speed and turning angle. The reward function designed by considering multiple objective optimization is defined as follows:

\[ r = k_1 \cdot r_{\text{col}} + k_2 \cdot r_s + k_4 \cdot r_f + k_5 \cdot r_{cy} + k_6 \cdot r_{\text{steer}} + k_7 \cdot r_b + c \]  

where

\[ r_{\text{col}} = \begin{cases} 
-1, & \text{if collision} \\
0, & \text{else} 
\end{cases} \]  

\[ r_s = -\text{steering}^2 \]  

\[ r_f = \begin{cases} 
-1, & \text{if speed} > 8 \\
0, & \text{else} 
\end{cases} \]  

\[ k_1, k_2, k_3, k_4, k_5, k_6 \] are set to 200, 1, 10, 1, 1, 1 empirically. As shown in Figure 2, the thick solid lines on both sides represent the boundary of the road. When the vehicle drives to the boundary or collides occurred in front and behind of the vehicle, the collision term \( r_{\text{col}} \) is defined as formula (7a). We found that if the penalty value is set too small, the trained policy will not be able to avoid collisions, if it is set too large, the algorithm will get negative returns and cannot learn an effective policy. Item \( r_s \) defined in formula (7b) is aimed to ensure the smoothness of the vehicle steering speed during cornering. \( r_f \) is a speed-related constraint term in formula (7c), it is assumed that the vehicle runs at a pre-set speed about 8 m/s, and the speed constraint penalty is to prevent the speed from being too fast. The constant value \( c \) in the multi-objective optimization reward function is set to -1, which guarantees to run continuously and get valid feedback. If this item is not included, the vehicle may stay stationary in simulation, because the errors of other items are equal to 0, and the penalty rewards of them do not work.

Item \( r_{ey} \) and \( r_{hr} \) are used to minimize the lateral distance error and the heading angle error. The desired heading angle is generated with vector field guidance (VFG) [9] and is described in Figure 2. As shown in Figure 2, the cyan arrow indicates the desired heading angle in current position, then the heading angle error \( \Delta \varphi \) is defined as the error between the desired and the sampled current heading angle. The objective item \( r_{hr} \) is defined as formula (8b). The lateral distance error \( e_y \) is shown in Figure 2, which is defined as the lateral distance between the reference path and current position, the correspondent objective item \( r_{ey} \) is defined as formula (8a). Both items are defined with exponential function, the path following performance can be achieved by controlling the lateral trajectory error and the heading angle error.

\[ r_{ey} = e^{-0.5 |e_y|} \]  

\[ r_{hr} = \begin{cases} 
e^{-|\Delta \varphi|} & \text{if } |\Delta \varphi| < 90^\circ \\
e^{180^\circ - |\Delta \varphi|} & \text{if } \Delta \varphi \geq 90^\circ \\
e^{-(180^\circ + |\Delta \varphi|)} & \text{if } \Delta \varphi \leq -90^\circ 
\end{cases} \]
2.4. CARLA Simulation setting
In order to verify the performance of the SAC learning method with multi-objective optimization reward functions, CARLA simulation software with different driving scenarios is used to train and validate the method. CARLA is an open source simulator that provides a high-fidelity dynamic world and different realistic physical vectors. Simulation were test on the map Town 3 provided by CARLA, the map area size is 400m*400m, where the road length is 6km, and the map integrates various urban traffic scenarios. Multiple vehicles were set up in random positions and drove freely in the virtual town to simulate the surrounding vehicle environment, at the beginning of training, the simulated vehicles were randomly set on a certain road.

2.5. Network structure settings
(1) VAE model processing layer is composed of four 3*3 convolutional layers of 32, 64, 128, 256 and finally a latent space layer of size 64 is fully connected to the final convolutional layer. The training data for the whole algorithm does not require any labels, all algorithm networks use the same convolutional layer and VAE network model. After processing the data, the $Q$ network and the policy $\pi$ network of SAC use same network structure.

(2) Using frame skipping technique. After output action at a certain moment in training, the output action is kept for $k=4$ consecutive frames with the same control commands for the vehicle. Reducing the action output data indirectly reduces the amount of training data and improves the training speed.

The several typical methods used in the comparison experiments are as follows: (1) DQN method. This method is a deep reinforcement learning algorithm proposed by Deepmind, which uses a combination of deep network and $Q$ learning; (2) DDPG combines Actor-Critic and policy gradient with each other, which can handle a continuous action space and finally output a deterministic policy; (3) TD3 [10] method uses the method of policy regularization and delayed update policy to improve the overestimated.

3. Simulations and analysis
From the simulation results, combining the BEV map with the sensors data can not only achieve a multi-constrained control task with stable cumulative returns, but also obtain higher cumulative returns.
In the experiment, the average return value is output after every 5000 training iterations and is used to evaluate the algorithm. During the training phase, if a vehicle collides or drifts out of its lane, all vehicles will be reset at random position throughout the map. As show in the Figure 3, the SAC algorithm of ours using multi-constrained reward function is significantly better than the baseline method, which can reach higher reward return values faster than other methods, ours can obtain higher reward returns at 20,000 steps and obtain the highest return values at 70,000 steps, with relatively flat and stable curves and the final trend tends to increase. In contrast, the curve of the benchmark DDPG method is very volatile, and there are four curves with the lowest return values.

The average return value of DQN is low and the curve is unstable and fluctuates widely, the vehicle rushes out of the lane during the turn (Figure 4) and collides with the middle barrier, no effective turning policy is learned. As shown in Figure 5, the TD3 algorithm continuously kept the vehicle stationary during the training process and could not complete normal driving, the training result shows that it cannot learn a valid driving policy.

The simulation results of the presented method and three input images of the presented method are shown in Figure 7 and Figure 8 respectively.
The DDPG algorithm has a minimum value and large fluctuations and despite the tracking error considered in the constraint, the vehicle just keeps turning left (Figure 6), colliding with the lane and almost unable to go straight and unable to explore the learning policy.

The results of the simulation of the method in this paper are shown in Figure 7 and 8. The vehicle trained by our algorithm can pass the aforementioned traffic scenario without colliding with the road, can pass the corner smoothly, and maintain a uniform speed. The trained output driving policy can already perform the driving tasks of various traffic scenarios relatively stable.

4. Conclusion
Based on the SAC-based DRL method, the original data from LiDAR and camera, combined with BEV map as inputs, are first processed with VAE for information condensing, and then the learned latent representations are fed into the SAC-based DRL module which included a designed multi-constraints-based reward function. The simulation results show that with the mixed inputs from multi-sensor data and bird-eye view map, the multi-constraints reward function-based SAC algorithm can make the vehicle adaptive to some challenging traffic scenes, such as intersection, turning, and straight-ahead situations.

Further work includes improving the efficiency of the algorithm and optimizing the training time, as well as conducting more simulation experiments with more challenging traffic scenarios.

References
[1] Dosovitskiy A, Ros G, Codevilla F, et al. CARLA: An open urban driving simulator [J]. arXiv preprint arXiv:1711.03938, 2017.
[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.
[3] Lange S, Riedmiller M, Voigtländer A. Autonomous reinforcement learning on raw visual input data in a real world application[C]//The 2012 international joint conference on neural networks (IJCNN). IEEE, 2012: 1-8.
[4] Yu A, Palefsky-Smith R, Bedi R. Deep reinforcement learning for simulated autonomous vehicle control[J]. Course Project Reports: Winter, 2016: 1-7.
[5] Lillicrap T P, Hunt J J, Pritzel A, et al. Continuous control with deep reinforcement learning[J]. arXiv preprint arXiv:1509.02971, 2015.
[6] Haarnoja T, Zhou A, Abbeel P, et al. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor[J]. arXiv preprint arXiv:1801.01290, 2018.
[7] Bansal M, Krizhevsky A, Ogale A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst[J]. arXiv preprint arXiv:1812.03079, 2018.
[8] Kingma D P, Welling M. Auto-encoding variational bayes [J]. arXiv preprint arXiv:1312.6114, 2013.
[9] Woo J, Yu C, Kim N. Deep reinforcement learning-based controller for path following of an unmanned surface vehicle [J]. Ocean Engineering, 2019, 183: 155-166.
[10] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods[C]//International Conference on Machine Learning. PMLR, 2018: 1587-1596.