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

Automatically Generated Curriculum based Reinforcement Learning for Autonomous Vehicles in Urban Environment

Zhiqian Qiao[1], Katharina Muelling[2], John M. Dolan[2], Praveen Palanisamy[3], Priyantha Mudalige[3]

***Abstract*— We address the problem of learning autonomous driving behaviors in urban intersections using deep reinforcement learning (DRL). DRL has become a popular choice for creating autonomous agents due to its success in various tasks. However, as the problems tackled become more complex, the number of training iterations necessary increase drastically. Curriculum learning has been shown to reduce the required training time and improve the performance of the agent, but creating an optimal curriculum often requires human handcrafting. In this work, we learn a policy for urban intersection crossing using DRL and introduce a method to automatically generate the curriculum for the training process from a candidate set of tasks. We compare the performance of the automatically generated curriculum (AGC) training to those of randomly generated sequences and show that AGC can significantly reduce the training time while achieving similar or better performance.**

I. INTRODUCTION

How to approach and traverse an urban intersection can be a difficult problem, not only for autonomous vehicles (AV) but also for human drivers. The main reason for this high rate of accidents can be found in the abundance of external factors that the driver needs to pay attention to when making a decision. Especially at intersections that are not controlled by a traffic light, the driver has to monitor all vehicles continuously and to estimate their intentions and velocities. Designing a reliable planning algorithm that allows self-driving cars to make safe and efficient decisions is therefore a difficult task that is hard to manually construct.

Prior distance-based and time-to-collision-based (TTC-based) algorithms [1] for the intersection traversing problem always include some tuning parameters to deal with different scenarios. Tuning these parameters is laborious since the algorithms are not easily adapted to various environmental situations. They also require the design of a large number of distance-based rules to handle different situations.

As machine learning based approaches and especially deep reinforcement learning (DRL) based approaches have become very popular, the idea of apply DRL to autonomous driving scenarios has gained some attention. Recent work on using deep RL for learning to cross intersections was able to show that it is able to learn successful policies that are comparable or even can outperform rule-based systems in terms of successfully reaching the goal [17]. Unlike rule-based algorithms, RL can learn to deal with a continuously changing environments by trial and error. Unlike supervised learning, RL does not need a large amount of labeled data to train a data-based model. Rather than learning a mapping from input to label, RL enables an autonomous agent to learn a mapping from environment states to agent actions from its experience, which is similar to how human learn to drive.

[1]Zhiqian Qiao is a Ph.D. Student with Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, USA, 15213. zhiqianq@andrew.cmu.edu

[2] The Robotics Institute, Carnegie Mellon University

[3] Research & Development, General Motors, Warren, MI, USA 48093

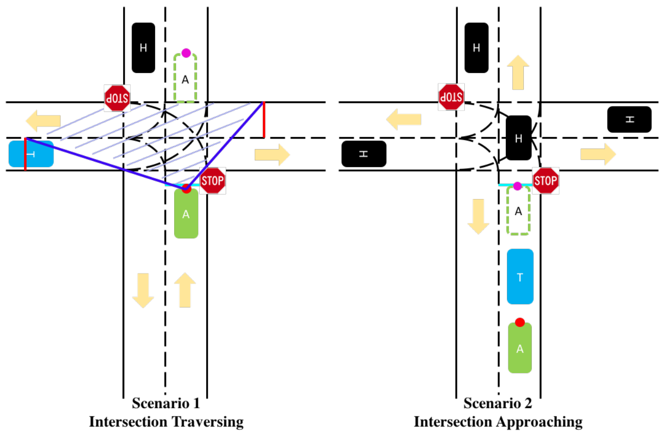

Fig. 1. Two proposed intersection scenarios. Scenario 1 is for intersection traversing problem and Scenario 2 is for intersection approaching problem. In the plots, the cyan lines are stop lines of the intersection corresponding to the red stop signs.

However, most DRL methods still have difficulty with high-dimensional state space problems such as autonomous vehicle decision-making in urban environments. The problem with DRL algorithms such as Deep Q-learning [4] (DQN) and Deep Deterministic Policy Gradient [5] (DDPG) is that they need a long training period to get an acceptable result. As a result, the basic objective of our work is to apply deep reinforcement learning (DRL) methods to train an agent that can autonomously learn how to approach and traverse an urban stop-sign intersection by collecting information on surrounding vehicles and the road. In this paper, we propose an algorithm which can generate a curriculum sequence to accelerate the training process for high-dimensional reinforcement learning problems. The paper's main contributions: 1). A novel algorithm, Automatically Generating Curriculum (AGC), which can help to solve tasks with high-dimensional state spaces within fewer iterations of training by using DRL. 2). Applications of AGC-based DRL that significantly reduces the training time and training iterations for a complex autonomous driving scenario by only using the information of other simulated vehicles within the ego vehicle's visibility range instead of a god-view which can get full information near the scenario.

The remainder of this paper is organized as follows: first, we give an overview of the related work in Section II. In Section III, we give a formal introduction to the problem addressed here and in Section IV, we describe the proposed algorithm. The experimental results for learning autonomous driving behavior and discussions about the results can be found in Section V. Finally, we make conclusions of the work in Section VI.

II. RELATED WORK

The previous work related to the paper is divided into two categories: First, papers address the problem of generating autonomous driving behavior in various environments [11][6]; second, previous work that focuses on DRL and algorithms that speed-up the training [8] [9].

A. Algorithms of autonomous driving behavior

1) Rule-based algorithm: Rule-based algorithms [10] [7] have been a popular choice for generating autonomous driving behaviors in urban environments. Time-to-collision (TTC) [12] is a rule-based algorithm which has been normally applied in intersection scenarios as a baseline algorithm. However, the rule-based algorithm needs much work from human beings to design various rules in order to deal with different scenarios in urban environments.

2) Learning-based algorithm: One of the main problems of rule-based systems is that they have problems anticipating potentially complex human driving behaviors. Instead of designing various rules to or set up different time-to-collision parameters, learning-based methods allows transferring multiple rules into a mapping function or one neural network. Brechtel et al. [14] formulated the decision-making problem under uncertain environments as a POMDP. They trained out a Bayesian Network representation to deal with a T-shape intersection merging problem. Sadigh et al. [15] modeled the interaction between autonomous vehicles and human drivers in a simulation. They simulated autonomous vehicles to motivate human drivers' reactions and acquired reward functions through Inverse Reinforcement Learning (IRL) [16]. [17] dealt with the traversing problem via Deep Q-Networks combined with a long-term memory component. They trained a state-action function Q to allow an autonomous vehicle to traverse intersections with moving traffic. However, they applied the Krauss stochastic driving model which allow the simulated vehicles to yield to ego vehicle when the situation has possible to crash. They also assumed full knowledge of other vehicles with a god view state space.

B. Methods of reinforcement learning and speed up technologies

1) Reinforcement learning approaches: Recent rapid advances in function approximation with neural networks have shown advantages in handling high-dimensional sensory inputs. Deep Q-Networks [4] (DQN) and its variants have been successfully applied to various fields including Atari games [18] and Go [19]. However, DQN is restricted to the discrete action domain, which makes it hard to apply to autonomous driving. DDPG [5], which was proposed by Lillicrap et al., adapted the ideas underlying the success of DQN to the continuous action domain. The algorithm has shown to produce the great performance for finding optimal policies and is competitive with other planning algorithms which need full access to the dynamics of the system.

2) Algorithms for Training Acceleration : For most learning-based algorithms, a major challenge is the long training time needed for the learning process. Curriculum learning [21] was proposed to speed up the learning process by first training the system on easy tasks and then gradually increase the complexity of the problems presented to the learning agent. Most of the work on Curriculum learning focused on simple scenarios which can be solved by hand-designed curricula[22]. For more complex tasks, however, hand-designing curricula is a difficult problem. Florensa et al. [23] proposed a curriculum generation method for reinforcement learning which trains the robot in reverse sequence: starting the learning process from an initial position close to the destination, it gradually increases the distance towards a random start configuration. However, the results are mainly based on static scenarios and not on changeable environments. In contrast, the complexity of autonomous vehicle problems is based on the different scenarios resulting from other vehicles' interactions. Therefore, the reverse curriculum cannot be applied directly in our work. [8] introduced an approach which can generate a curriculum as a directed acyclic graph instead and did experiments on the agents using RL. [9] designed a Teacher-Student Curriculum which learns a curriculum by supervised learning or reinforcement learning in order to complete some tasks which cannot be finished if trained directly without a particular curriculum.

In this work, we take the hardest situation that even though the simulated vehicles have seen the ego vehicle when approaching the intersection, they do not slow down to yield to the ego vehicle. This kind of scenario is more similar to a normal stop-sign intersection in the real world and is much more difficult to deal with for ego vehicle. We utilize a neural network to approximate the policy function and apply an automatically generated curriculum based reinforcement learning method to learn the optimal behavior policy and decrease the training time and training iterations.

III. PROBLEM DEFINITION

The setup of scenario considered in this work is a four-way intersection with two-way stop signs (see Fig. 1). In each scenario, the AV (green solid rectangle with the letter "A") has to reach a pre-defined destination (green hollow rectangle with the letter "A"). Vehicles shown as blue rectangles with the letter "T" (front vehicle or approaching vehicles according to different scenarios) are target vehicles within ego vehicle's visibility range whose information will be included in the state space. All the simulated vehicles in the scenario stay in their lanes with constant velocities. Furthermore, all traffic participants except the AV will not change their trajectory in response to the AV.

We consider two scenarios in the work. In scenario 1, which is referred as Intersection Traversing, the AV has successfully stopped close to the stop line and has to yield to the target vehicles (blue rectangle with the letter "T") on the main road and then traverse the intersection to the other side. In scenario 2, the AV is approaching the intersection and has to first stop at the stop line while also avoiding running into the target vehicle in front of it (blue rectangle with the letter "T"). We refer to this scenario as Intersection Approaching.

A. Preliminaries

Formally, we model the autonomous intersection crossing problem as a Markov Decision Process (MDP) which is defined as a tuple $\{S, A, R, T, \gamma\}$. Here, S denotes a set of states, A defines the set of available vehicle actions, and $T(s_{t+1}, a_t, s_t)$ is a transition function that maps a state-action pair (s_t, a_t) to a new state s_{t+1}. Specifically, we define the state space to contain the velocity of the ego vehicle, the target vehicle velocity (if exists within visibility range of ego vehicle), the time to collision with the ego vehicle and road geometry information including the distance to the stop line d_{sl}, the lower boundary d_{lb}, the mid-point d_{mp} and the upper boundary d_{ub} of the intersection. Vehicle T in scenario 1 and 2 (see Fig. 1) is the target vehicles for each task. The action space for the first scenario is formulated as a discrete decision of $Observe \in [0,1], Wait \in [0,1]$ or $Go \in [0,1]$. For the second scenario, the action is a continuous value which describes the acceleration or deceleration of the ego vehicle. The reward function R defines the immediate rewards for each state-action pair and γ is a discount factor for the long-term reward.

B. Reward function

For the proposed problem, the reward function is designed as follows:

- For the ego vehicle, a positive reward is calculated according to the percentage of the trip that has been finished plus a negative constant reward for the penalty of time in each step: $r_{ego} = \sigma_1 \frac{\|p_{des} - p_{ego}\|^2}{\|p_{des} - p_{init}\|^2} - \sigma_2$ where σ_1 and σ_2 are constants and p_{init}, p_{des} and p_{ego} represent the initial position, destination and current position of ego vehicle, respectively.
- For a target vehicle, a negative reward will be added if the vehicle is too close to the ego vehicle, which may result in a potential crash in the future step: $r_{target} = \sum_i \sigma_3 \max\left[d_{safe} - (\|p^i_{target} - p_{ego}\|^2), 0\right]$.
- A constant penalty σ_4 is imposed if a crash happens ($\|p^i_{target} - p_{ego}\|^2 = 0$) and a constant penalty σ_5 for unfinished of a task in 2000 epochs which is 200 seconds in real world for both Intersection Approaching and Traverse scenarios. For approaching scenario, an extra penalty σ_6 is imposed for not stopping at the stop line. Moreover, a positive reward σ_7 is added if the ego vehicle reaches the goal ($\|p_{des} - p_{ego}\|^2 = 0$). Here, σ_7 is a constant and p^i_{target} is the position of the i^{th} target vehicle.

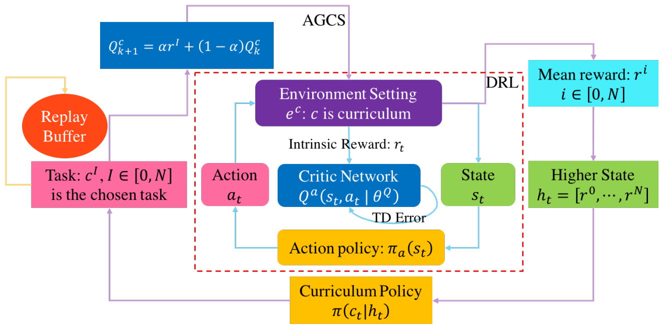
AGCS
$Q^c_{k+1} = \alpha r^I + (1-\alpha)Q^c_k$
DRL
Mean reward: r^i $i \in [0, N]$
Replay Buffer
Environment Setting e^c: c is curriculum
Intrinsic Reward: r_t
Task: $c^I, I \in [0, N]$ is the chosen task
Action a_t
Critic Network $Q^a(s_t, a_t \mid \theta^Q)$
State s_t
Higher State $h_t = [r^0, \cdots, r^N]$
TD Error
Action policy: $\pi_a(s_t)$
Curriculum Policy $\pi(c_t|h_t)$

Fig. 2. Flow chart of Deep Reinforcement Learning with Automatically Generated Curriculum Sequence

IV. METHODOLOGY

We model both the Intersection Traversing and Intersection Approaching scenario as an MDP. Our goal is to learn a policy π that maximizes the expected accumulated reward $\mathbb{E}[R(s_t, a_t), \pi]$. To learn the optimal policy we use two established deep reinforcement learning algorithms: Deep Q-learning [4] and Deep Deterministic Policy Gradient [5].

While both methods have been successfully applied to various tasks, training directly on a complex task as considered in this paper can lead to long training times. Instead of training an agent from scratch, here we suggest using curriculum learning (CL)[21] by first choosing a relatively easy task to train on and then gradually increasing the difficulty of the problem presented to the agent during the learning process. The main problem with CL is the design of the curriculum, i.e., generating the sequence of problems that will decrease the number of training iterations needed to converge. As a result, we address this problem by proposing an automatic curriculum generation method for Reinforcement Learning. A general flowchart showing the training process is shown in Fig. 2.

A. Automated Curriculum Generation

AGC-based reinforcement learning is curriculum reinforcement learning method which involves two levels of learning: The higher level is responsible for automatically generating a curriculum according to total rewards of test samples after the current training iteration with respect to each task. In the lower-level structure, we apply a traditional DRL algorithm such as DQN or DDPG to train a policy. The actions considered can be either discrete or continuous.

Fig. 2 shows the flow chart for AGC-based DQN. The inner rectangle with a red dashed outline is the DRL (DQN or DDPG) process and the outer part is the process for automatically generating the curriculum used for training the DRL algorithm. In the outer part, the policy of the curriculum generation is formulated as a k-armed bandit problem [2], where k is chosen based on the number of tasks. In this paper, we use an action-value based incremental method for the AGC-based RL algorithm. We update the V function of the curriculum generation through Equation 1 according to [2], in which $n \in [1, N]$ is the task id and k specifies training iteration.

Algorithm 1 Automatically Generating Curriculum for DQN

```
1:  procedure AGC-RL
2:      Construct an empty replay buffer B
3:      for $n \leftarrow 1$ to $N$ candidate tasks do
4:          Randomly initialize critic network $NN^n_{Q^a}$ with weights $\theta^n_Q$ and the target critic network $NN^n_{Q^{a'}}$ with weights $\theta^n_{Q'}$.
5:          for $e \leftarrow 1$ to $E$ epochs do
6:              $r^n_e$, $NN^n_{Q^a}$, $NN^n_{Q^{a'}}$, $TB_n$ = Train($task^n$, $NN^n_{Q^a}$, $NN^n_{Q^{a'}}$, B) and add $TB_n$ into B
7:          $V^0_n = \frac{1}{E}\sum_e r^n_e$
8:      for $k \leftarrow 0$ to $K$ training iterations do
9:          $P(n) = \pi(c_n|V^k_n) = \frac{\exp(V^k_n)}{\sum^N_n \exp(V^k_n)}$
10:         $I = \text{sample}([1,\cdots,N], \text{prob}=[P(1),\cdots,P(N)])$
11:         if $k \geq 1$ and sample([0,1], prob=[$P(n), 1-P(n)$]) is 1 then:
12:             Add $TB_n$ into B according to $P(n)$
13:         for $e \leftarrow 1$ to $E$ epochs do
14:             $r^I_e$, $NN^n_Q$, $TB_n$ = Train($task^I$, $NN^I_{Q^a}$, B)
15:         for $n \leftarrow 1$ to $N$ candidate tasks do
16:             $V^{k+1}_n = \alpha\frac{1}{E}\sum_e r^n_e + (1-\alpha)V^k_n$
17:             for $e \leftarrow 1$ to $E$ epochs do
18:                 Get initial states $s_0$ of $task^n$
19:                 for $t \leftarrow 0$ to $T$ do
20:                     Select $a_t = \arg\max_{a_t} Q^a(s_t, a_t)$ and execute $a_t$
21:                     $s_{t+1} = T(a_t, s_t)$, $r^n_e = r_{ego} + r_{target} - \sigma_4\mathbb{I}_{\|p^i_{target}-p_{ego}\|^2=0} + \sigma_5\mathbb{I}_{\|p_{des}-p_{ego}\|^2=0}$
22: procedure TRAIN($task^n$, $NN^n_{Q^a}$, $NN^n_{Q^{a'}}$, B)
23:     Empty $TB_n$ and get initial states $s_0$ of $task^n$, $r^n = 0$
24:     for $t \leftarrow 0$ to $T$ do
25:         Select $a_t = \pi(s_t)$ according to $\varepsilon$ exploration and execute $a_t$ to get new state $s_{t+1} = T(s_t, a_t)$.
26:         Get reward $r_t$ and $r^n += r_t$
27:         Add $(s_t, a_t, r_t, s_{t+1})$ to the temporary Replay Buffer $TB_n$ and sample random mini-batch of $M$ transitions $(s_i, a_i, r_i, s_{i+1})$ from the replay buffer B.
28:         Minimize critic loss: $L = \sum_i (y_i - Q^a(s_i, a_i|\theta^Q))^2$ where $y_i = r_i + \gamma Q^{a'}(s_{i+1}, a_{i+1}|\theta^{Q'})$.
29:         Update weights: $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
```

$$V^{k+1}_n = \alpha r^n + (1-\alpha)V^k_n \tag{1}$$

We use the value function V_i to evaluate the difficulty of each task i. Higher values indicate that this kind of task is relatively easier to get a better performance than other tasks. AGC-based RL chooses the task to be trained in the next iteration via the Boltzmann distribution exploration method: $\pi(c_n|V_n) = \frac{\exp(V_n)}{\sum^N_n \exp(V_n)}$. An easier task with a higher V_n will result in a higher probability to be chosen as the curriculum for the next training iteration.

B. Replay Buffer

DQN [4] and DDPG [5] usually store the last M experiences as tuples of $s_t, a_t, r_t, s_{t+1}, a_{t+1}$ in a replay buffer **B**. The original algorithm uses uniform sampling, which gives equal importance to all the transitions in the replay memory and may result in an unbalanced memory with more failure cases than success epochs. Here, the experiences during a training iteration are added to the Replay Buffer **B** such that it represents the probability of the current chosen task I to be chosen in next training iteration, which is $\pi(c_I|V^{k+1}_I)$. This means that after training on one task in one training iteration, if the training result improves the policy and gets better performance, the experiences during the training iteration have a higher probability of being added to the replay buffer. Meanwhile, if the transition is not chosen, it will be abandoned without storing it in the buffer. As a result, the buffer only contains transitions which may include more buffers that can help to train out a better result. The complete algorithm is shown in Algorithm 1.

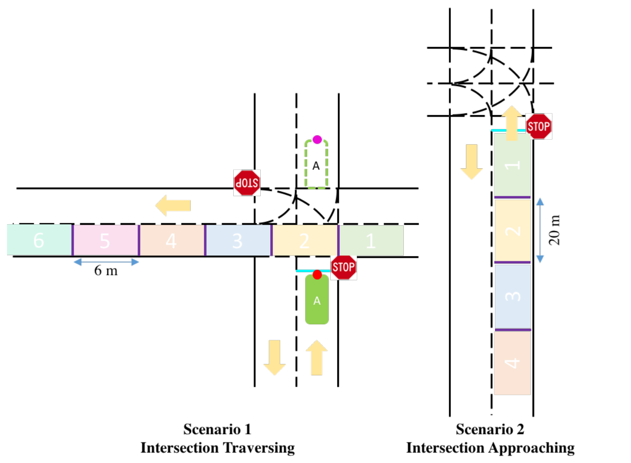

Fig. 3. Intersection traversing problem is divided into six tasks according to initial position of approaching human vehicles.

V. EXPERIMENTS

We applied the proposed methods to the two scenarios described in Section III. We modeled both scenarios as MDP and trained the Q-network with DQN and DDPG respectively for scenario 1 and scenario 2. For both scenarios, we trained an agent using a random curriculum and compared the performance and the training time with an agent trained with the AGCS-generated curriculum.

A. Experimental Setup

By modeling the problem as an MDP, we solved the Intersection Traversing and Intersection Approaching problems using Deep Q-learning [4] and Deep Deterministic Policy Gradient [5], respectively. All the critic network and actor networks for DQN and DDPG are constituted by two hidden layers with 600 and 300 nodes. "ReLU" activation function is used for all hidden layers and "tanh" activation function is used for the actor output in DDPG. We set up a replay buffer with a size of 500000.

For every training epoch, a successful case means that there is no collision between the ego vehicle and other vehicles and the whole process can be finished within 2000 steps (200 seconds). We use collision rate and steps to finish as metric to evaluate the tasks.

B. Random Curriculum

We applied the DQN or DDPG algorithm by generating curricula randomly, which means the probability of each task to be chosen obeys a discrete uniform distribution. After training for 250000 iterations, we can get an 80% of success rate for Traversing scenario and for Approaching scenario, it takes 7000 to get to a success rate of 60%.

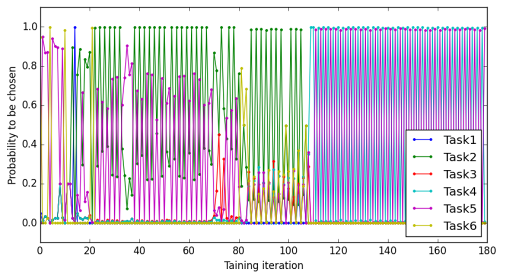

Fig. 4. Probability of being chosen for next iteration of training for the intersection traversing case

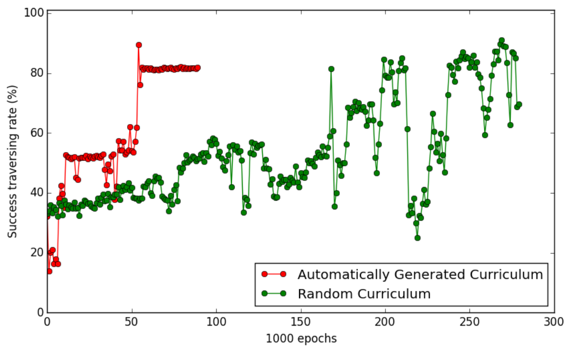

Fig. 5. Success rate of Random Curricula and AGC-based Model for the intersection traversing case

C. *Manually Designed Curriculum*

In order to manually design the curriculum, we firstly train DQN on each task separately. According to different performance for each task, we manually design the curriculum, to begin with the task which achieves higher success rate more quickly.

D. *Automatically Generated Curriculum*

For intersection traversing problem, we created a set of six tasks according to different initial positions of the first approaching vehicles and for intersection approaching problem, four tasks are generated according to the initial positions of the ego vehicle. The initial positions of approaching simulated vehicles or ego vehicle are randomly generated for initialization at first epoch and different numbers in the Fig. 3 is corresponding to different task IDs.

1) Intersection traversing problem: Applying the AGC-based approach, the probability of each task to be chosen for the next training iteration varies according to the mean rewards the task can get during the current training iteration. Fig. 4 shows the probability density function of each task to be chosen for the next training iteration for different tasks according to the AGC-based model for the intersection traversing case. We see that in the first period (roughly before the 100th training iteration), the system prefers Task 2 because training Task 1 can reach an acceptable result easily and Task 2 is relatively simple compared to other tasks. However, in the later training period, the most difficult tasks, Task 4 and 5, are preferred over other tasks because they get the lowest scores when the other scenarios perform well.

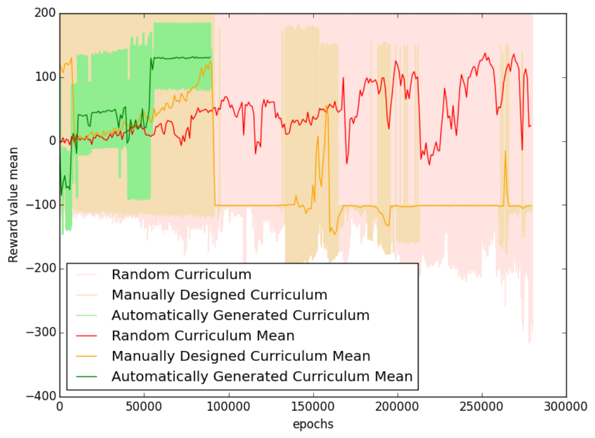

Fig. 6. Rewards Value of three methods for the intersection traversing case

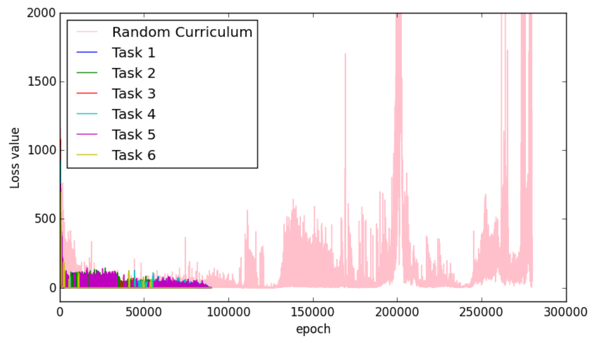

Fig. 7. Loss Function of Random Curricula and AGC-based Model for the intersection traversing case

As a result, we compared the success rate and mean reward values by using random curriculum and AGC-based DQN in Fig. 5 and Fig. 6. With the help of the AGC structure, the system can reduce training time by a factor of six to reach a similar and more stable performance compared to the vanilla DQN algorithm. In Fig. 6, we compared AGC with Random Curriculum and Manually Designed Curriculum (MDC). Especially for MDC, the system performs well for some easy-to-handle tasks; however, when the difficulty increases and the curriculum is not designed well, it may not achieve the expected result. Random Curriculum can achieve an acceptable result after a long time of training.

Fig. 7 shows the critic loss of lower-level DQN. It is shown that if it is trained directly via random curriculum, the system is not stable, with the training time increase, the system may have a divergent critic loss. However, the use of AGC helps the critic loss to gradually decrease and although the chosen task keeps switching, the critic loss is bounded all the time and has the tendency to decrease with the increase of training time.

2) Intersection approaching problem: For the intersection approaching problem, we created a set of four tasks. Each task corresponds to a different range of initial positions for the ego vehicle. The initial distance between the front vehicle and the AV is randomly generated and is greater than 10 meters. The front vehicle always stops at the stop

TABLE I

COMPARISON OF THE RESULTS BETWEEN DIFFERENT METHODS WITH THE SAME TRAINING ITERATIONS

		Steps	Collision	Unfinished	Not stop	Success	Mean Reward
Intersection Traverse	TTC	365	2.2%	10.5%	N/A	87.3%	22.1
	Random Curricula	294	25.6%	16.2%	N/A	58.2%	67.3
	AGC-based Curricula	206	13.5%	4.4%	N/A	82.1%	**132.1**
Intersection Approaching	Random Curricula	183	22.5%	20.7%	15.6 %	41.2%	150.23
	AGC-based Curricula	130	0.21%	0.10%	1.0%	98.69%	**480.39**

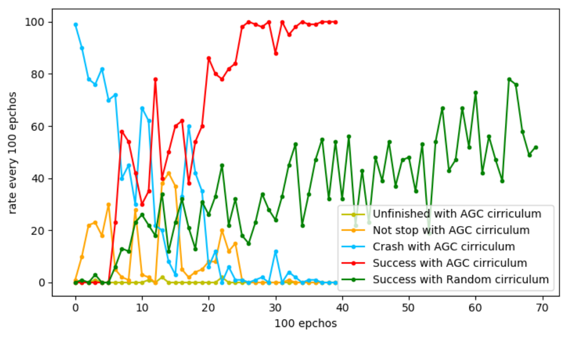

Fig. 8. Success rate of Random Curricula and AGC-based Model for the intersection approaching case

sign first and then speeds up to traverse the intersection. The initial velocity of the ego vehicle is randomly generated and is between 8 m/s and 12 m/s and the destination of the ego vehicle is to stop throughly at the stop line. Fig. 8 compares the results between Random Curricula and AGC-based model. It is shown that with the AGC model, the system may take fewer iterations to achieve the 99.2% success rate.

E. Discussion

For testing the performance of the different method, we train the model with same numbers of iterations and then use the result to test for 1000 episodes with the task chosen at random in each episode. The result of the test with different methods is shown in Table I. For the intersection traverse scenario, we train the model for 100000 iterations. The TTC method was found to be too conservative and generated Observer or Wait actions for longer period of time than necessary to traverse the intersection. This resulted in several unfinished episodes were the TTC method failed to traverse through the intersection. For the intersection approaching scenario, we train the model for 40000 iterations and the AGC-based model can get a much better performance on the test tasks.

VI. CONCLUSIONS

Deep reinforcement learning is promising for autonomous vehicle behavior planning problems in which rule-based algorithms may have difficulty. However, DRL always needs a long training period for a good result and sometimes cannot obtain acceptable results. This paper proposes an AGC-based DRL method which significantly reduces training time compared to plain DRL for the autonomous vehicle behavior planning problem at intersections. Our initial results have focused on simple scenarios. In future work, we aim to extend the AGC model to more complicated scenarios.

REFERENCES

[1] Dresner, Kurt M., and Peter Stone. "Sharing the Road: Autonomous Vehicles Meet Human Drivers." IJCAI. Vol. 7. 2007.
[2] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.
[3] Silver, David, et al. "Mastering the game of go without human knowledge." Nature 550.7676 (2017): 354-359.
[4] Mnih, Volodymyr, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).
[5] Lillicrap, Timothy P., et al. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015).
[6] Dong, Chiyu, John M. Dolan, and Bakhtiar Litkouhi. "Interactive ramp merging planning in autonomous driving: Multi-Merging leading PGM (MML-PGM),." 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). 2017.
[7] Ramyar, S., et al. "Fuzzy modeling of drivers' actions at intersections." World Automation Congress (WAC), 2016. IEEE, 2016.
[8] Svetlik, Maxwell, et al. "Automatic Curriculum Graph Generation for Reinforcement Learning Agents." AAAI. 2017.
[9] Matiisen, Tambet, et al. Teacher-Student Curriculum Learning. arXiv preprint arXiv:1707.00183 (2017).
[10] Baker, Christopher R., and John M. Dolan. "Traffic interaction in the urban challenge: Putting boss on its best behavior." Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on. IEEE, 2008.
[11] Dong, Chiyu, John M. Dolan, and Bakhtiar Litkouhi. "Intention estimation for ramp merging control in autonomous driving." Intelligent Vehicles Symposium (IV), 2017 IEEE.
[12] Lee, David N. "A theory of visual control of braking based on information about time-to-collision." Perception 5.4 (1976): 437-459.
[13] NGSIM dataset, available from: http://www.ngsim.fhwa.dot.gov/.
[14] Brechtel, Sebastian, Tobias Gindele, and Rdiger Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs. Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on. IEEE, 2014.
[15] Sadigh, Dorsa, et al. Planning for Autonomous Cars that Leverage Effects on Human Actions. Robotics: Science and Systems. 2016.
[16] Ng, Andrew Y., and Stuart J. Russell. Algorithms for inverse reinforcement learning. Icml. 2000.
[17] Isele, David, Akansel Cosgun, and Kikuo Fujimura. "Analyzing Knowledge Transfer in Deep Q-Networks for Autonomously Handling Multiple Intersections." arXiv preprint arXiv:1705.01197 (2017).
[18] Mnih, Volodymyr, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).
[19] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, et al. Mastering the game of go with deep neural networks and tree search. Nature, 529(7587):484489, 2016.
[20] Kulkarni, Tejas D., et al. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. Advances in Neural Information Processing Systems. 2016.
[21] Bengio, Yoshua, et al. Curriculum learning. Proceedings of the 26th annual international conference on machine learning. ACM, 2009.
[22] Bengio, Samy, et al. "Scheduled sampling for sequence prediction with recurrent neural networks." Advances in Neural Information Processing Systems. 2015.
[23] Florensa, Carlos, et al. Reverse curriculum generation for reinforcement learning. arXiv preprint arXiv:1707.05300 (2017).
[24] Moore, Andrew W., and Christopher G. Atkeson. "Prioritized sweeping: Reinforcement learning with less data and less time." Machine learning 13.1 (1993): 103-130.