Model-Based Reinforcement Learning with Self-attention Mechanism for Autonomous Driving in Dense Traffic

Junjie Wen[1,2], Zuoquan Zhao[1], Jinqiang Cui[2](✉), and Ben M. Chen[1]

[1] Department of Mechanical and Automation Engineering, The Chinese Uinversity of Hong Kong, Hong Kong, People's Republic of China

[2] Department of Mathematics and Theories, Peng Cheng Laboratory, Shenzhen 518055, China

cuijq@pcl.ac.cn

Abstract. Reinforcement learning (RL) has recently been applied in autonomous driving for planning and decision-making. However, most of them use model-free RL techniques, requiring a large volume of interactions with the environment to achieve satisfactory performance. In this work, we for the first time introduce a solely self-attention environment model to model-based RL for autonomous driving in a dense traffic environment where intensive interactions among traffic participants may occur. Firstly, an environment model based solely on the self-attention mechanism is proposed to simulate the dynamic transition of the dense traffic. Two attention modules are introduced: the Horizon Attention (HA) module and the Frame Attention (FA) module. The proposed environment model shows superior predicting performance compared to other state-of-the-art (SOTA) prediction methods in dense traffic environment. Then the environment model is employed for developing various model-based RL algorithms. Experiments show the higher sample efficiency of our proposed model-based RL algorithms in contrast with model-free methods. Moreover, the intelligent agents trained with our algorithms all outperform their corresponding model-free methods in metrics of success rate and passing time.

Keywords: Reinforcement Learning · Attention Mechanism · Autonomous Driving

1 Introduction

Decision-making in a dense environment with dynamically moving obstacles has been a hot topic in the mobile robotics community. One typical problem is passing through dense traffic like unsignalized urban intersections for autonomous vehicles.

Although there are several algorithms that have already been successfully applied to autonomous vehicles [19,25], most of them still rely on hand-crafted rule-based methods which are only applicable in simple driving circumstances. Situations as complex as passing through intersections would dramatically increase the complexity of rule-based algorithm. In contrast, learning-based algorithms can better deal with complex driving situations without complexity increase. Some researchers applied imitation learning (IL) to autonomous driving by training the intelligent agent in a supervised manner [7,30]. However, IL suffers from tedious data labeling and poor generalizability.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

Unlike IL, RL-based methods can better generalize to unseen scenarios without data labeling by interacting with the environment. Decision-making in dense traffic environment with RL-based methods have already been studied [11,16]. However, most of them are based on model-free RL algorithms, which only train the intelligent agent by exhaustively interacting with the environment. Model-based RL algorithms, on the other hand, explicitly learn the environment transitions as well as interact with the environment, leading to an improvement in sample efficiency.

Researchers [9,31] have already applied model-based RL to autonomous driving. However, the environments they are dealing with are simple driving scenarios. In a complex environment where interactions across traffic participants are unavoidable, the environment model should be carefully designed. In such an environment, one participant's action not only determines its own future state but also others' behaviors. Compared to other model architectures, the self-attention mechanism is more capable of extracting the interactive dependencies across all inputs.

In this work, we for the first time apply a solely self-attention environment model to model-based RL for autonomous driving in a dense traffic environment. Our contributions are listed as follows:

- We propose an environment model solely on the self-attention mechanism to simulate the dynamic transitions in a dense traffic environment and it shows better performance in predicting the future states of traffic participants than other SOTA methods.
- We introduce two self-attention modules: the Horizon Attention module to encode the spatially interactive information and the Frame Attention module to encode sequential state features.
- The developed environment model has been introduced to model-based RL for autonomous driving in dense traffic. Testing results show the superiority of our proposed model-based RL algorithms over their model-free RL counterparts.

2 Related Work

2.1 Attention Mechanism

Attention Mechanism is first introduced in natural language processing (NLP) by Bahdanau et al. [3] as an improvement over the encoder-decoder neural machine translation system based on recurrent neural network (RNN) or long short-term memory (LSTM) [10]. However, due to the vanishing/exploding gradient problem, RNN suffers from long-range dependency. In addition, the sequential nature of RNN/LSTMs makes the parallelization of training examples extremely hard. By calculating the embeddings of all input words and the weighted sum of hidden states, the attention mechanism not only allows for the modeling of dependencies without regard to the distances but also accelerates the training process by data parallelization.

Based on the attention-mechanism, many breakthroughs have been made in NLP and computer vision tasks. The Transformer [26] differs from previous neural translation models in that it is based on self-attention mechanisms to draw global dependencies between input and output. It gets SOTA performance in translation with less training time and significantly more parallelization. BERT [5] also achieves SOTA performances in a wide range of NLP tasks by pretraining deep bidirectional representations from unlabeled text based on Transformers and finetuning one additional task-specific output layer. ViT [6] directly applied a pure Transformer architecture [26] to sequences of image patches and showed excellent results on image classification tasks. Swin-Transformer [17] surpassed previous SOTA networks on a broad range of vision tasks by a large margin with a hierarchical Transformer whose representation is computed with shifted windows.

In this work, we equip the environment model of dense traffic with self-attention mechanisms to extract the dependencies across both different traffic participants and different state histories.

2.2 Traffic Trajectory Prediction Methods

Predicting the future trajectories of other traffic participants in dense traffic is a complex problem for the autonomous driving community. Early researchers [21,25] mostly focused on rule-based Finite/Hybrid State Machines to encode the desired behavior of the vehicle in the encountered urban scenarios. However, the complexity of these methods could dramatically increase as the driving scenarios become more general. Besides, the constant velocity assumption that is usually employed in those systems ignores the surrounding vehicles' reactions, leading to potential risks in dense urban traffic scenario [8].

The learning-based prediction method, on the other hand, does not show a complexity increase even in complicated situations and is more capable of modeling the participants' intentions. Altche et al. [2] introduced an LSTM-based network that predicted the future longitudinal and lateral trajectories for vehicles on the highway. Ma et al. [18] also proposed an LSTM-based network that contains instance and category layers to predict the trajectories of heterogeneous traffic participants. Tang et al. [24] introduced a probabilistic framework to efficiently model the future motions of agents with a dynamic attention-based state encoder.

However, most previous learning-based prediction methods are based upon RNN or combine RNN with the attention mechanism, which could be time-consuming during inference due to RNN's poor parallelization capability. In this work, we propose a prediction network that is solely based on the self-attention mechanism for forecasting the future motions of all the participants involved in dense traffic.

2.3 Reinforcement Learning

RL involves interactions between the intelligent agent and the environment. It is usually formulated as a Markov decision process $< S, A, P, R, \gamma >$, with the measurable space S, action space A, state transition dynamics P, reward function R and discount factor γ. The training purpose for solving RL task is just to find a policy π that maximizes the expected γ-discounted cumulative reward.

In the RL paradigm, based on whether the intelligent agents predict the environment responses, there are two kinds of RL methods: model-free and model-based RL. Model-free RL indicates that the agent learns policies by directly interacting with the

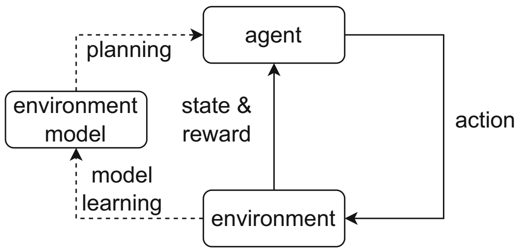

Fig. 1. Illustration of model-free and model-based RL. The loop formed by the solid line denotes the model-free RL while the loop of dotted line represents the model-based RL.

environment, while model-based RL constructs a predictive model of the environment dynamics and integrates both learning and planning, as illustrated in Fig. 1.

Among various model-free algorithms, Q-learning [28] is a typical value optimization method by learning a near optimal state-action value function with a sufficient number of learning samples. Deep Q-Networks (DQN) [20] achieved SOTA performances in many Atari games by introducing deep neural network (DNN) as a non-linear value approximator and experience replay technique. The Q value updating rule is:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)), \tag{1}$$

where $Q(S_t, A_t)$ is the estimated action value of state S when choosing action A at time t, $\alpha \in [0, 1]$ is the learning rate.

Instead of evaluating a value function before getting the optimal policy, policy optimization methods aim to obtain an optimal policy directly. In policy optimizations, the policy function is usually parameterized as a learnable neural network π_θ. The REINFORCE [29] algorithm is a simple straightforward policy optimization algorithm, whose policy parameters are updated by:

$$\theta \leftarrow \theta + \alpha\gamma^t G \nabla_\theta \ln\pi(A_t|S_t, \theta), \tag{2}$$

where θ denotes the parameter of policy network, α is the learning rate and $\gamma^t G$ is the γ-discounted cumulative reward. Actor-Critic [14] is a policy optimization method which integrates the benefits of value optimization. Deep Deterministic Policy Gradient (DDPG) [22] is an actor-critic algorithm that can learn policies for continuous action space with DNN approximation function.

However, model-free methods usually suffer from low sample efficiency. For intelligent agents which need to be applied to practical situations, interacting with real-world environments would be limited due to cost and safety concerns. Model-based RL [12,23] is advantageous in reducing the required number of interactions by simultaneously learning the environment transitions. In this work, we develop various model-based RL algorithms with our proposed environment model for autonomous driving in dense traffic.

3 Methodology

3.1 Problem Definition

The problem of developing model-based RL algorithms for passing through dense traffic could be decomposed into two tasks: one is constructing an environment model for dense traffic, and the other is solving the passing problem with RL algorithm with the environment model.

We first assume the states of agents involved in the traffic are preprocessed, including their spatial coordinates, velocities, and headings. At time t, the feature vector of any agent A_i^t is denoted as $f_i^t = (p_x, p_y, v_x, v_y, \cos\phi, \sin\phi)_i^t$, where p_x/p_y is the coordinate in x-/y- axis, v_x/v_y is the velocity in x-/y- axis, and ϕ is the agent heading. In the following paragraphs, we denote 'ego-agent' as the traffic participant controlled by RL algorithms and 'other-agent' as the participants in the environment except 'ego-agent'. The task of formulating an environment model is just to observe the states of all the agents during the time interval $[1, T_{obs}]$ and predict their states and rewards from the environment at time step $t = T_{obs} + 1$.

With the environment model, the problem of autonomous driving in dense traffic could be viewed as the ego-agent navigating to a target position without collision with other-agents. It is an optimization problem whose objective is to maximize the expected cumulative reward. For an environment with one ego-agent and N other-agents, the optimization objective should be:

$$\begin{aligned} \underset{\pi_\theta}{\text{argmax}} \quad & \mathbb{E}[R|\pi_\theta, S_e, S_{o,1:N}], \\ s.t. \quad & T_e \in [0, T_{\max}], P_e \in P_f, V_e \in V_f \end{aligned} \tag{3}$$

where π_θ is the policy taken by the intelligent ego agent, R is the cumulative reward, S_e is the ego agent state, $s_{o,1:N}$ is the states for other agents. T_e is the passing time of ego agent, with the maximum allowable time as $T_{\max}$. P_e and V_e are the ego agent position and velocity, which should be limited in the feasible position region P_f and velocity region V_f, respectively. The collision restriction is not shown in Eq.(3) because it is integrated in the cumulative reward R.

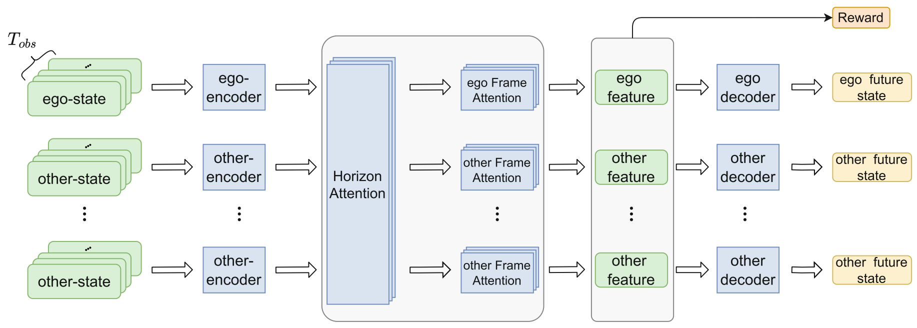

Fig. 2. The overall network structure for traffic environment modeling with HA precedes FA (HA-FA model).

3.2 Environment Model

The transition dynamics of a dense traffic environment is complex due to the interactions between different agents. The action taken by one agent results in both its own state transition and others' decision makings. Hence, the environment model for a dense traffic should consider not only the state histories of each agent, but also the interaction dependencies on others. We propose two self-attention modules to deal with both the state histories and the interactions: the Frame Attention (FA) module to encode the sequential state features for each agent and the Horizon Attention (HA) module to encode the interactive information with others.

The overall structure of our environment model can be seen in Fig. 2. The sequential states of each agent within the time interval T_{obs} are first embedded by encoders. The parameters of the ego-encoder differ from the other-encoders because it is used for embedding ego-agent's state. The embedded features are then fed into the HA module to extract the interactive information across agents at each time step $t \in [1, T_{obs}]$. After that, the FA module draws the time dependencies of sequential features with length T_{obs} for each agent, with ego-FA for ego-agent and other-FA for other-agents. Decoders are finally used to get the state of each agent in the next time step. The reward from the environment is regressed by a feed-forward layer after concatenating all the FA features.

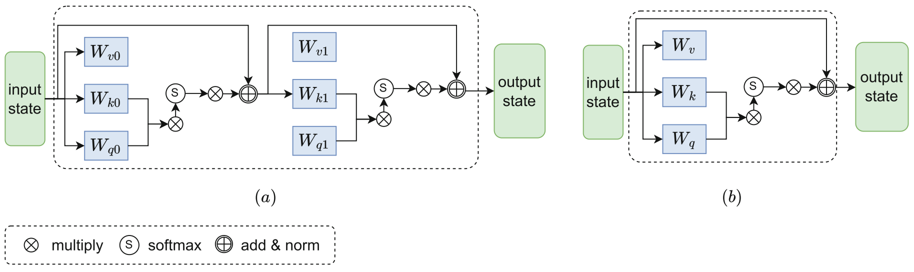

Fig. 3. (a) The detailed architecture of single-head HA-module. (b) The detailed architecture of single-head FA-module.

The detailed architectures of HA and FA modules can be seen in Fig. 3. Both modules adopt the self-attention mechanism used in [26]. HA-module recursively uses two self-attention modules while FA-module uses one. The main difference between HA and FA modules lies not in their architectures but in their functionalities.

Horizon Attention (HA): The HA module is based on the multi-head self-attention mechanism to extract the dependencies between different agents that have interactions. In this module, the relationship between each agent can be extracted by a weight matrix, in which a higher value indicates a more important dependency. The HA module outputs a high-dimensional feature for each agent that not only encodes the ego-agent state but also other-agents' state and their corresponding relationship.

Frame Attention (FA): The FA module is proposed to encode the sequential features of each agent in the scene. Compared to RNN/LSTM architecture, the FA module is advantageous in data parallelization and dependency extraction. Besides, taking an agent's full history of states gives it a better understanding of the dynamic environment.

Algorithm 1. Procedure for model-based value optimization in dense traffic

```
1: Initialize value function $Q(s, a)$, environment model $Model(s, a)$ and an empty experience
   buffer $B$
2: While not done:
3:     Observe $S$ from real environment
4:     Get action $A \leftarrow \varepsilon$-greedy$(S, Q)$
5:     Execute action $A$, observe reward $R$ and state $S'$
6:     $B \leftarrow B \cup (S, A, R, S')$
7:     Update $Q(S, A)$ with Eq.(1) and $Model(s, a)$ with $(S, A, R, S')$
8:     Repeat $n$ times:
9:         $S \leftarrow$ random previously observed state
10:        $A \leftarrow$ random previous action taken in $S$
11:        $R, S' \leftarrow Model(S, A)$
12:        Update $Q(S, A)$ with Eq.(1)
```

3.3 Model-Based RL for Dense Traffic

Model-based RL is more advantageous than model-free RL in its higher sample efficiency. With an environment model, the intelligent agent could learn how the environment transfers and then predict the action that would lead to desirable outcomes. In this work, we use our proposed environment model to generate imagined future rollouts and add them to the experience buffer from which the intelligent agent could learn a passing policy.

The procedure for model-based value optimization RL that combines direct RL, model learning, and planning is presented in Algorithm 1. The environment model is trained online with the real experience from the environment, and the value function is updated by both the real environment experience and the imagined rollouts from the environment model.

Algorithm 2. Procedure for model-based policy optimization in dense traffic

```
1: Initialize parameters $\theta$ of policy $\pi$, parameters $\alpha$ of environment model $M$ and an empty
   experience buffer $B$
2: While not done:
3:     Observe $S$ from real environment
4:     Get action $A \leftarrow \varepsilon$-greedy$(\pi_\theta)$
5:     Execute action $A$, observe reward $R$ and state $S'$
6:     $B \leftarrow B \cup (S, A, R, S')$
7:     Update environment model $M_\alpha$ with $(S, A, R, S')$
8:     Update policy $\pi_\theta$ with $M_\alpha$ and $B$
```

The general algorithm for model-based policy optimization is shown in Algorithm 2. The policy is also updated with both real and imagined experiences. The policy updating method is general such that it could be implemented by any policy optimization method such as TD3 [4], DDPG [22], etc.

4 Experiments

4.1 Environment Setup

The simulated environment is modified upon the intersection environment presented in [15]. The environment range is set to $[-40\,\mathrm{m}, 40\,\mathrm{m}]$. The ego-agent is rewarded by 5 if it reaches its destination, -5 if it collides with other-agents or violates traffic rules, and a positive reward of 0.5 is used to encourage driving in a desired speed range. There are 3 levels of difficulty for the environment based on the average number of other-agents, i.e. easy, medium, and hard.

The agents in the simulated environment are controlled in a hierarchical manner, including a low-level lateral controller and a high-level longitudinal controller. The lateral action is implemented by a low-level lateral tracker for both ego- and other- agent. The longitudinal acceleration is realized by our learned policies for the ego-agent while implemented by a rule-based intelligent driver model (IDM) [13] for other-agents.

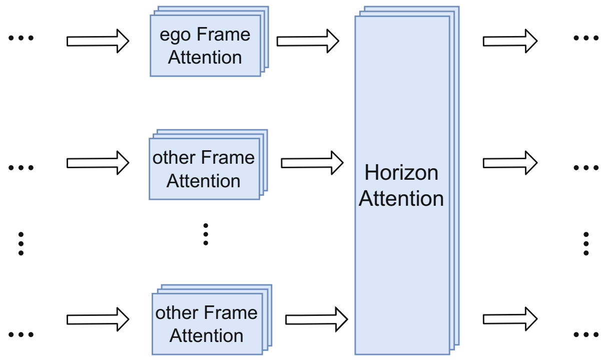

Fig. 4. FA-HA environment model. (Dots indicate the same with HA-FA model.)

4.2 Environment Model Evaluation

In order to evaluate the effectiveness of our environment model, we first collect experience data from the simulated environment through random policy and then train environment models in a supervised manner. The training set contains 100k data while the testing set contains 10 k. The models are trained on a computer with the configurations: NVIDIA GTX1080Ti graphics card, Intel i7 6800k, 48 GB RAM. During training, the batch size is 32, the learning rate is set to 0.0005, and maximum training epoch is set to 50.

Besides the proposed HA-FA model, we also implement an FA-HA model as shown in Fig. 4, where the only difference is the order of HA and FA modules. The HA/FA module is also replaced with multi-layer perceptron (MLP) to get MLP-FA/HA-MLP models. Other implemented environment models include: Social-LSTM (SL) [1], Social-Attention (SA) [27], GMM [31], WM [9] and a naive MLP model.

Table 1. Testing Displacement Error of Different Models

Models	MLP	SA	SL	GMM	WM	MLP-FA	HA-MLP	FA-HA	HA-FA
Average Error (m)	0.650	0.325	0.329	0.401	0.457	0.359	0.190	0.436	**0.167**
Straight Road Error (m)	0.276	0.151	**0.141**	0.191	0.188	0.183	0.192	0.204	0.143
Intersection Error (m)	0.782	0.388	0.397	0.477	0.555	0.425	0.187	0.520	**0.172**

The positional displacement error of these methods with respect to the testing data is shown in Table 1. It is noted that our proposed HA-FA model achieves the best average prediction performance than other methods. All these methods achieve good results in straight road prediction, indicating the effectiveness of common prediction methods in simple driving scenarios. However, in complex situations like intersections, the performances of these methods differ a lot. Our proposed HA-FA model performs best in such situation, mainly contributed to the adoption of HA module as compared with MLP-FA model. It should be noted that FA-HA model shows a dramatic performance drop even though it contains both HA and FA modules. The reasons are discussed in Sect. 5.

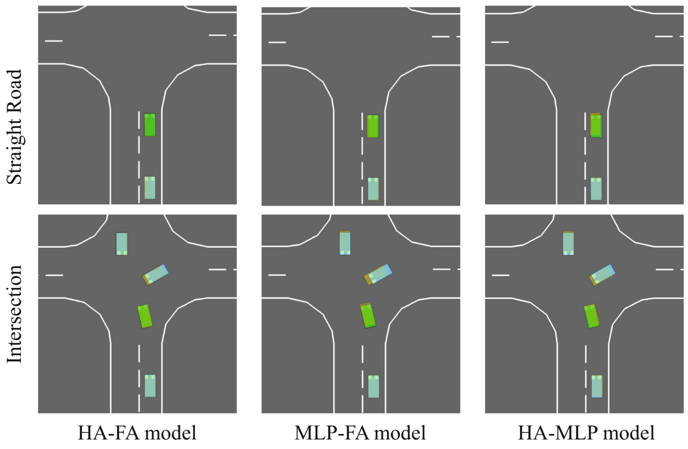

Fig. 5. Visualization of environment transition predictions. Green rectangle: ego-agent, blue rectangle: other-agents, transparent yellow rectangle: predicted states. (Color figure online)

In order to better understand the effectiveness of HA/FA modules, the visualization results of the predicted future positions for different agents of HA-FA/MLP-FA/HA-MLP models are shown in Fig. 5. It is seen that our HA-FA model has superior prediction

Table 2. Average Inference Time of Different Models

Models	MLP	SA	SL	GMM	WM	MLP-FA	HA-MLP	FA-HA	HA-FA
Average Inference Time (s)	**0.042**	0.102	0.097	0.059	0.115	0.047	0.051	0.052	0.055

performance for agents in both straight roads and intersections. The MLP-FA model, however, has similar performance in straight road scenario while shows bad predictions when agents are in the intersection region. In contrast, the HA-MLP model shows better results than MLP-FA in intersections but slightly worse when agents are at straight road.

The average inference time of each model is computed in Table 2. Although the inference time of our proposed HA-FA model is not the shortest, it's still much less than the time used by other RNN/LSTM- based models like SA/SL/WM, validating the effectiveness of parallelization for the self-attention mechanism.

4.3 Model-Based and Model-Free RL

We implement the training of different RL algorithms in both model-based and model-free manners. Model-free RL methods have been tried with both value optimization like DQN and policy optimization methods like DDPG and TD3. For DQN, the ego-agent output three discrete actions "slow-down","idle", and "speed-up". For policy optimization algorithms, the ego-agent produces continuous acceleration ranging in $[-5, 5]$. The ego-agents for all the implemented algorithms have the same structure as the Ego-Attention network [16], and our proposed HA-FA model is used as the environment model for the model-based RL methods.

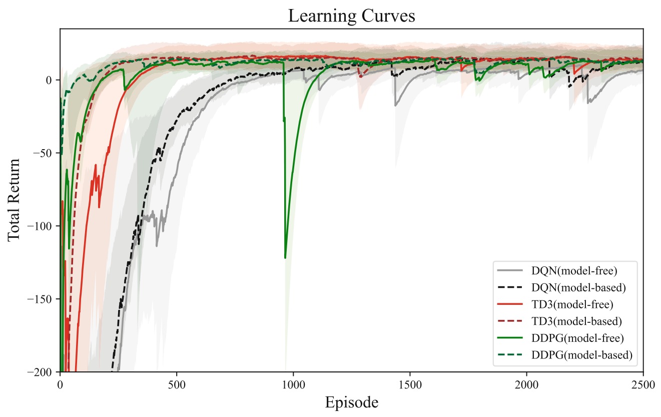

Fig. 6. Learning curves of model-based and model-free methods

The learning curves are shown in Fig. 6 and the average cumulative reward for each methods is shown in Table 3. If we define "Rising Time" as the used episodes when

Table 3. Cumulative Reward and Rising Time of Different Methods (MF: model-free, MB: model-based)

Models	DQN (MF)	DQN (MB)	TD3 (MF)	TD3 (MB)	DDPG (MF)	DDPG(MB)
Cumulative Reward	15.3	20.9	20.8	**22.3**	21.1	21.3
Rising Time	1215	1038	381	219	238	**137**

Table 4. Evaluated result of success rate and passing time (MF: model-free, MB: model-based)

Success Rate	Easy	Medium	Hard	Passing Time	Easy	Medium	Hard
IDM (baseline)	97.02%	95.05%	88.12%	IDM (baseline)	132.46	147.19	173.84
DQN (MF)	95.34%	92.23%	86.78%	DQN (MF)	128.36	140.34	175.53
DQN (MB)	96.81%	92.13%	87.09%	DQN (MB)	127.53	139.28	175.42
TD3 (MF)	96.32%	95.31%	88.19%	TD3 (MF)	125.55	138.54	172.33
TD3 (MB)	**97.57%**	**96.87%**	**89.77%**	TD3 (MB)	**120.21**	**136.91**	**170.85**
DDPG (MF)	95.44%	94.98%	88.09%	DDPG (MF)	125.38	139.33	173.95
DDPG (MB)	96.87%	95.87%	88.59%	DDPG (MB)	123.29	137.65	171.32

the cumulative reward reaches 90% of the final average reward, then the method with less rising time has better sample efficiency, as shown in Table 3. It can be seen that all the model-based methods equipped with our proposed environment model show higher cumulative reward and better sample efficiency than their corresponding model-free methods. The model-based TD3 method achieves the highest cumulative reward and the model-based DDPG method has the best sample efficiency.

The testing result of each method with respect to the success rate and passing time is shown in Table 4. The baseline metrics are implemented upon the IDM method. It is seen that the model-based methods with our proposed environment model achieve a relatively higher success rate and less passing time than their corresponding model-free methods. Model-based TD3 and DDPG algorithms could even outperform the rule-based IDM method in our evaluated metrics.

5 Discussion

In this section, we qualitatively discuss the performance drop by the order exchange of HA and FA modules. As shown in Fig. 7(a), the HA-FA model first extracts the interactive dependencies across agents in each frame by HA, and then the decoded features of all frames are fed into FA to get the time dependencies. In contrast, as shown in Fig. 7(b), the FA-HA model first gets the time dependencies of each agent and then extracts the interactive dependencies with the encoded features, which has fewer connections than the HA-FA model.

Hence, the performance drop of the FA-HA model might be caused by two factors: (1) The environmental information is not fully utilized by the FA-HA model. As shown in Fig. 7(a)(b), the HA-FA model fully utilizes the interactive information of all frames,

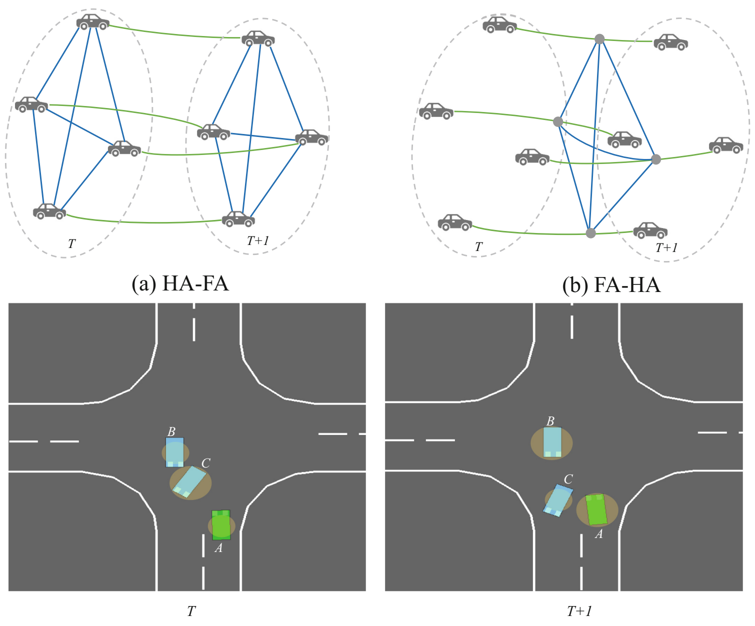

Fig. 7. (a) Network connections for HA-FA model, (b) Network connections for FA-HA model, Blue Lines: Horizon Attention, Green Lines: Frame Attention. (c) Illustration about the different attention importance for different agents at different time steps. (Color figure online)

while the FA-HA model only uses the encoded features from the FA module with time dependency while ignoring the interactive information at most time frames; (2) The time dependencies extracted by the FA module for each agent could vary for different agents such that the HA module could not get useful interactive information. As shown in Fig. 7(c), the encoded feature for agent C might have higher importance at frame T while agents A and B higher at frame $T + 1$.

6 Conclusion

In this work, an environment model for dense traffic based solely on the self-attention mechanism has been proposed and model-based RL algorithms upon the proposed environment model have been developed. The FA and HA modules are presented: the FA module is used to encode temporal features of one agent while the HA module is to understand the interactive dependencies among all agents spatially at each time frame. The HA-FA model shows superior performance over other SOTA methods in predicting future states of traffic participants. The performance drop of the FA-HA model could be caused by less information utilization and a mismatch of temporal dependencies for different agents. Model-based RL algorithms for both value and policy optimizations have been developed for dense traffic environments. With our proposed HA-FA model, we successfully trained intelligent agents in a model-based manner for various RL algorithms like DQN, DDPG, and TD3 with better sample efficiency and higher cumulative reward. The agents trained with the proposed model-based algorithms also surpass the corresponding agents trained in a model-free manner with a higher success rate and less passing time.

References

1. Alahi, A., Goel, K., Ramanathan, V., et al.: Social LSTM: human trajectory prediction in crowded spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 961–971 (2016)
2. Altché, F., de La Fortelle, A.: An LSTM network for highway trajectory prediction. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), pp. 353–359. IEEE (2017)
3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
4. Dankwa, S., Zheng, W.: Twin-delayed ddpg: a deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. In: Proceedings of the 3rd International Conference on Vision, Image and Signal Processing, pp. 1–5 (2019)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. Dosovitskiy, A., Beyer, L., Kolesnikov, S., et al.: An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
7. Eraqi, H.M., Moustafa, M.N., Honer, J.: End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. arXiv preprint arXiv:1710.03804 (2017)
8. Fletcher, L., Teller, S., Olson, E., et al.: The MIT-cornell collision and why it happened. J. Field Robot. **25**(10), 775–807 (2008)
9. Ha, D., Schmidhuber, J.: World models. arXiv preprint arXiv:1803.10122 (2018)
10. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
11. Isele, D., Nakhaei, A., Fujimura, K.: Safe reinforcement learning on autonomous vehicles. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1–6. IEEE (2018)
12. Kaiser, L., Babaeizadeh, M., Milos, P., et al.: Model-based reinforcement learning for atari. arXiv preprint arXiv:1903.00374 (2019)
13. Kesting, A., Treiber, M., Helbing, D.: Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. Philos. Trans. R. Soc. A: Math. Phys. Eng. Sci. **368**(1928), 4585–4605 (2010)
14. Konda, V., Tsitsiklis, J.: Actor-critic algorithms. Adv. Neural Inf. Process. Syst. **12**, 1008–1014 (1999)
15. Leurent, E.: An environment for autonomous driving decision-making. GitHub (2018)
16. Leurent, E., Mercat, J.: Social attention for autonomous decision-making in dense traffic. arXiv preprint arXiv:1911.12250 (2019)
17. Liu, Z., Lin, Y., Cao, Y., et al.: Swin transformer: hierarchical vision transformer using shifted windows. arXiv preprint arXiv:2103.14030 (2021)
18. Ma, Y., Zhu, X., Zhang, S., et al.: Trafficpredict: trajectory prediction for heterogeneous traffic-agents. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, pp. 6120–6127 (2019)
19. Minderhoud, M.M., Bovy, P.H.: Extended time-to-collision measures for road traffic safety assessment. Accid. Anal. Prev. **33**(1), 89–97 (2001)

20. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
21. Montemerlo, M., Becker, J., Bhat, S., et al.: Junior: the stanford entry in the urban challenge. J. Field Robot. **25**(9), 569–597 (2008)
22. Silver, D., Lever, G., Heess, N., et al.: Deterministic policy gradient algorithms. In: International Conference on Machine Learning, pp. 387–395. PMLR (2014)
23. Sutton, R.S.: Dyna, an integrated architecture for learning, planning, and reacting. ACM Sigart Bull. **2**(4), 160–163 (1991)
24. Tang, C., Salakhutdinov, R.R.: Multiple futures prediction. Adv. Neural Inf. Process. Syst. **32**, 15424–15434 (2019)
25. Urmson, C., Anhalt, J., Bagnell, D., et al.: Autonomous driving in urban environments: boss and the urban challenge. J. Field Robot. **25**(8), 425–466 (2008)
26. Vaswani, A., Shazeer, N., Parmar, N., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
27. Vemula, A., Muelling, K., Oh, J.: Social attention: modeling attention in human crowds. In: 2018 IEEE international Conference on Robotics and Automation (ICRA), pp. 4601–4607. IEEE (2018)
28. Watkins, C.J., Dayan, P.: Q-learning. Mach. Learn. **8**(3–4), 279–292 (1992)
29. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3), 229–256 (1992)
30. Xu, H., Gao, Y., Yu, F., Darrell, T.: End-to-end learning of driving models from large-scale video datasets. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2174–2182 (2017)
31. Zhuo, X., Jianyu, C., Masayoshi, T.: Guided policy search model-based reinforcement learning for urban autonomous driving. arXiv preprint arXiv:2005.03076 (2020)