Deep Predictive Autonomous Driving Using Multi-Agent Joint Trajectory Prediction and Traffic Rules

Kyunghoon Cho, Timothy Ha, Gunmin Lee, and Songhwai Oh

Abstract— **Autonomous driving is a challenging problem because the autonomous vehicle must understand complex and dynamic environment. This understanding consists of predicting future behavior of nearby vehicles and recognizing predefined rules. It is observed that not all rules have equivalent values, and the priority of the rules may change depending on the situation or the driver's driving style. In this work, we jointly reason both a future trajectories of vehicles and degree of satisfaction of each rule in the deep learning framework. Joint reasoning allows modeling interactions between vehicles, and leads to better prediction results. A rule is represented as a signal temporal logic (STL) formula, and a robustness slackness, a margin to the satisfaction of the rule, is predicted for the both autonomous and other vehicle, in addition to future trajectories. Learned robustness slackness decides which rule should be prioritized for the given situation for the autonomous vehicle, and filter out non-valid predicted trajectories for surrounding vehicles. The predicted information from the deep learning framework is used in model predictive control (MPC), which allows the autonomous vehicle navigate efficiently and safely. We test the feasibility of our approach in publicly available NGSIM datasets. Proposed method shows a driving style similar to the human one and considers the safety related to the rules through the future prediction of the surrounding vehicles.**

I. Introduction

Research on autonomous vehicles has been grown exponentially in the past few years. Recently, Waymo's autonomous cars have successfully driven 8 million miles on public roads. Still, one of the greatest challenges to build fully autonomous cars is the understanding dynamic driving scene. Especially, the prediction of surrounding vehicles is difficult to achieve, since drivers have different driving patterns and are dependent on surrounding vehicles' movement. In addition, the decision on how to drive based on the understanding of the surrounding environment remains a problem; Autonomous vehicles must be controlled taking into account the movements of surrounding vehicles and safety problems, such as traffic regulations.

Unlike other robot applications, rule information acts as a critical role in autonomous driving problem. It is important to observe that rules do not have the same importance and that there are rules that, depending on the situation, should be prioritized or, in some cases, ignored. For example, in autonomous driving, there may be a situation in which some rules must be disobeyed, such as changing lanes behind a slowly moving vehicle, passing a speed limit to avoid collisions. Such dilemma makes it difficult to determine how autonomous vehicles should comply with rules to a certain extent, and it is difficult to find control inputs for autonomous vehicles in view of these restrictions.

In this paper, we propose a deep predictive control approach for the autonomous driving problem. We combine the benefits of deep learning and a traditional control method, which is model predictive control (MPC), so that the proposed controller can reason about the future behavior of nearby vehicles and behave closely to human experts by learning to obey each rule to some extent depending on the situation. Both the future trajectory and margin to the satisfaction of each rule are predicted through the deep learning framework, and a valid control is computed through the model predictive control procedure. The key traits of our prediction on future trajectories are the following:

- **Multimodal**: In a given situation, the movement of the vehicles is not fixed in one way. There can be diverse plausible futures for the current driving situation. To take this into account, we use a conditional variational autoencoder (CVAE) [1] combined with a recurrent neural network (RNN) so that it can generate various prediction candidates to reflect a distribution on plausible futures.
- **Interaction**: Each vehicle in the driving environment is dependent on the surrounding vehicles. The interaction between these vehicles must be considered in the process of making predictions. Our work jointly predicts the future trajectories of several vehicles to capture these interactions between agents.
- **Selection**: Not all prediction candidates are valid, and invalid predictions must be eliminated for safe autonomous driving. We use the predicted degree of satisfaction of each rule to select a valid prediction candidates. For example, if you know the vehicle will not speed up, the range of predictions becomes narrower.

In the control procedure, we use our previous work [2]. The predefined rules are defined as signal temporal logic (STL) [3], [4], Due to the advantage of STL, which is able to measure the degree of satisfaction, the MPC method [2] can manage rule constraints with respect to their importance; From the predicted margin to satisfaction of each rule, the controller can figure out which rules to follow, rather than maintaining strict compliance with all rules. The predicted trajectories of the surrounding vehicles are also considered in the control procedure so that the autonomous driving vehicle can be operated safely.

K. Cho, T. Ha, G. Lee, and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul 08826, Korea (e-mail: {kyunghoon.cho, timothy.ha, gunmin.lee}@rllab.snu.ac.kr, songhwai@snu.ac.kr).

This work was supported in part by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01190, (SW Star Lab) Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning) and by the ICT R&D program of MSIT/IITP (No. 2019-0-01309, Development of AI Technology for Guidance of a Mobile Robot to its Goal with Uncertain Maps in Indoor/Outdoor Environments).

II. Related Work

Path prediction. Due to advance in deep learning method, research on trajectory prediction has been extensively studied. Exploiting the power of RNNs for sequence-to-sequence modeling, there are many studies that predict future trajectories from history trajectories. A RNN encoder-decoder framework with conditional variational autoencoder (CVAE) has been applied for trajectory prediction in [5], [6]. They are structurally similar to our method, but we additionally considered interaction between multiple vehicles. The interaction between humans has been considered in [7]–[9]; They focused on how to efficiently capture dependencies between multiple pedestrians. Although we consider that the interaction is similar to the study, we focus on capturing the dependencies between vehicles in autonomous driving scenarios and, based on this, we ensure that the correct control can be generated.

Control under temporal logic specifications. Computing optimal control under temporal logic specifications has been considered in the context of linear temporal logic (LTL) and signal temporal logic (STL). Mixed-integer linear programming was proposed to generate trajectories for continuous systems with finite-horizon LTL specifications in [10]–[12]. Also sampling-based motion planning methods have been successfully find an optimal trajectory meeting a given LTL specification [13], [14]. The MPC scheme has been applied to signal temporal logic (STL) [15], [16]. MPC with STL specifications is formulated as mixed integer linear programming in [15], and authors in [16] use probabilistic predicates to reason about safety under uncertainty.

III. Preliminaries

A. Vehicle Model

We let the state of the system at time t be $\mathrm{x}_t = [x_t, y_t, \theta_t, v_t]^T$, where x_t, y_t denote the position of the vehicle, θ_t is the heading, and v_t is the linear velocity. Further, the control inputs of the system is $\mathrm{u}_t = [w_t, a_t]^T$, where w_t is the angular velocity, and a_t is the acceleration. The dynamic of the vehicle is defined as follows:

$$\dot{x}_t = v_t cos(\theta_t),\ \dot{y}_t = v_t sin(\theta_t),\ \dot{\theta}_t = v_t \kappa_1 w_t,\ \dot{v}_t = \kappa_2 a_t,$$

where κ_1, κ_2 are constants.

For a fixed horizon H, let $\mathbf{x}(\mathrm{x}_n, \mathbf{u}^{H,n})$ be a generated trajectory starting from x_n with control inputs $\mathbf{u}^{H,n} = \mathrm{u}_n, ..., \mathrm{u}_{n+H-1}$. A *signal* is a sequence of states and controls, which is defined as:

$$\xi(\mathrm{x}_n, \mathbf{u}^{H,n}) = (\mathrm{x}_n, \mathrm{u}_n), \ldots, (\mathrm{x}_{n+H-1}, \mathrm{u}_{n+H-1}). \quad (1)$$

With a slight abuse of notation, $\xi(n)$ is a signal starting from time step n.

B. Signal Temporal Logic

Signal temporal logic (STL) is a logical formalism which is able to specify the properties of real-value, dense-time signals [3], [4]. A STL formula is a composition of boolean and temporal operations on the defined predicates. The notation $(\xi, t) \vDash \varphi$ denotes that a signal ξ satisfies the STL formula φ at time t. For example, $(\xi, t) \vDash \mathrm{G}_{[a,b]}\varphi$ means that φ holds for the signal ξ between $t + a$ and $t + b$. One great advantage of STL is that it is provided with a metric called *robustness degree* that measures how well a given signal ξ satisfies a STL formula φ. The robustness degree can be defined as a real-valued function of signal ξ and t.

We apply a notation $(\xi, t) \vDash (\varphi, r)$ representing that the signal ξ satisfies the STL formula φ at time t with *robustness slackness* r, which can be defined as follows [2]:

$$(\xi, t) \vDash (\varphi, r) \quad \equiv \quad \rho^{\varphi}(\xi, t) > r. \quad (2)$$

Eqn (2) states that the signal ξ satisfies φ at least with the minimum robustness degree r. The robustness slackness r acts as a margin to satisfaction of STL formula ρ^{φ}. As r increases, strong constraints for the signal ξ to satisfy φ at time t are given, while relaxed constraints are granted for small r. Especially, when $r < 0$, it allows violation of φ.

5 different rules are defined with the corresponding STL formulas $\varphi = [\varphi_1, ..., \varphi_5]$, which have some meaning in autonomous driving situations.

1) Lane keeping (left): $\varphi_1 = y_t \leq y_{l,max}$
2) Lane keeping (right): $\varphi_2 = y_t \geq y_{l,min}$
3) Collision avoidance (Front vehicle): $\varphi_3 = (x_t \leq x_{c,min}) \vee (x_t \geq x_{c,max}) \vee (y_t \leq y_{c,min}) \vee (y_t \geq y_{c,max})$
4) Speed limit: $\varphi_4 = v_t \leq v_{max}$
5) Slow down before the front vehicle: $\varphi_5 = (v_t \leq v_{th}) \mathrm{U}_{[t_a,t_b]} (x_t \leq x_{c,min})$

Also we let φ_c be the collision between vehicles beside the front vehicle. We denote robustness slackness of the STL rules φ as r.

IV. Problem Definition

Our objective is to find a control sequence of some finite time horizon under the pre-defined rules φ in an autonomous driving environment. In order to compute control sequence of the autonomous vehicle, our approach requires two following terms: (a) future trajectories of other vehicles, (b) robustness slackness of each rule in φ. The first term directly related to the safety of autonomous vehicle, and the second term helps the controller to imitate the behavior of human experts by knowing the lower bound of degrees of satisfaction for each rule.

We assume that six nearby vehicles that are in adjacent lanes affect the control of the ego vehicle (Figure 1). Notation for vehicles on track and input feature are shown in Figure 1. We mark the vehicle to be controlled as V_{ego}. The nearby vehicles are denoted as $V_{near} = \{V_{lf}, V_{lr}, V_{cf}, V_{cr}, V_{rf}, V_{rr}\}$. From the rest of paper, we use the subscript ego for the ego vehicle, and $\{lf, lr, cf, cr, rf, rr\}$ are used for the surrounding vehicles. A feature representation f_{ego} consists of distances to nearby vehicles V_{near} and lane deviate distance $f_{ego} = (d_{lf}, d_{lr}, d_{cf}, d_{cr}, d_{rf}, d_{rr}, d_{dev})$.

Let $X = \{s_{t-H_X}, ..., s_t\}$ and $Y = \{s_{t+1}, ..., s_{t+H_Y}\}$ be previous and future trajectory , where $s_t \in \mathbb{R}^3$ is the position x_t, y_t, heading θ_t at time t and H_X, H_Y are time horizons. r_{ego} and $r_{near} = \{r_{lr}, r_{lf}, ..., r_{rf}\}$ refer robustness slackness of the STL formulas φ for the ego vehicle and the surrounding vehicles.

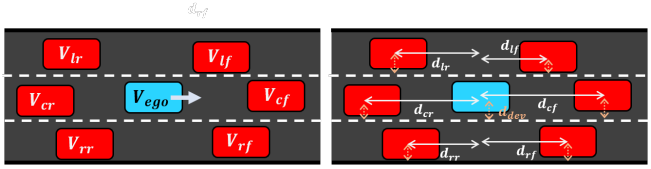

Fig. 1. Vehicle and feature descriptions in the track driving scenarios with respect to the ego vehicle (blue). Up to six nearby vehicles are considered; The front and rear vehicles are considered, which are in the left, middle and right lanes with respect to the ego vehicle V_{ego}.

V. Proposed Method

The proposed method consists of four modules (Figure 2); encoder module, interaction module, prediction module and control module. Through the deep neural networks including encoder module, interaction module and prediction module, we jointly reason about the future trajectories and robustness slackness of STL formulas φ for both the ego vehicle V_{ego} and near vehicles V_{near}. Since our model makes joint predictions for vehicles V_{ego}, V_{near}, it has the capacity to model interactions between vehicles. From predicted robustness slackness and trajectories, the ego vehicle is controlled through MPC procedure under the rule constraints φ. Instead of the strict satisfaction of rule restrictions, the designed controller is able to decide which rules should be prioritized and how satisfied they should be depending on the situation. We choose to use long short-term memory units (LSTMs) [17] as the RNN in both encoder and decoder module. Next we will detail these four modules sequentially.

A. Encoder Module

The encoder module is designed to model pattern of vehicles, including different history path and direction. The history of trajectories for each selected vehicles are encoded through LSTM networks. As input to the encoder network, difference values are used for each points of the trajectory. For each input trajectory, single layer multi-layer perceptron (MLP) to get fixed length vector ϕ. These embeddings are used as input to the LSTM cell of the encoder at time t introducing the following recurrence:

$$e^t = \phi(\Delta x_t, \Delta y_t, \theta_t; W_{emb})$$
$$h^t = LSTM(h^{t-1}, e^t; W_{enc})$$

where $\phi(\cdot)$ is an embedding function with tanh nonlinearity, W_{emb} is the embedding weight. The parameters of LSTM are shared between both the ego vehicle and near vehicles. Notice that future trajectories are also encoded during the training procedure, and the encoded future trajectories are used in the interaction module. The encoder module returns LSTM state h^t for each vehicle, and We will notate the output of the encoder module for each vehicle as follows: hx for history trajectories and hy for future trajectories.

B. Interaction Module

Based on the outputs of the encoder module, the interaction module measures joint information between vehicles. We focused on two terms; The first is to consider the diversity that can occur in the current situation (multimodality), and second, to exclude scenarios that can not occur (robustness slackness). As we approach from these two perspectives, we predict futures multimodally in a given situation and exclude invalid predictions among the various predictions based on robustness slackness. The interaction module composed of two components which will be described in the following sections.

1) Diverse sample generation: Since future prediction is inherently ambiguous, there are various interpretations of the present driving situation. In order to capture the ambiguity and multiple plausible scenarios, we use CVAE structure which is able to explain uncertainties and generate plausible scenarios, which is to learn a distribution of $p(Y|X, f_{ego})$. The CVAE framework introduces a latent variable z so that $p(Y|X, f_{ego}) = \int_z p(Y|X, z)p(z|X, f_{ego})dz$. The latent variable z models inherent structure in the interaction of multiple vehicles, and it also helps to describe underlying ambiguity of future behaviors of other vehicles.

The CVAE structure composed of the three neural networks, which are generation network $p_{\nu_g}(Y|X, z)$, recognition network $q_{\nu_r}(z|Y, X, f_{ego})$ and (conditional) prior network $p_{\nu_p}(z|X, f_{ego})$, where ν_g, ν_r, ν_p are network parameters. $q_{\nu_r}(z|Y, X, f_{ego})$ is trained to give higher probability to z that is likely to produce of prediction $\hat{Y}$ close to the ground truth Y. While at test time, z is sampled randomly from the prior distribution $p_{\nu_p}(z|X, f_{ego})$ and transferred to the prediction module for generating a prediction candidate. Such a structure enables probabilistic inference which serves to handle multi-modalities in trajectory prediction.

In this work, the distribution of the latent variable z is modeled as Gaussian distribution:

$$z \sim q_{\nu_r}(z|Y, X, f_{ego}) = \mathcal{N}(\mu_z, \sigma_z),$$

where μ_z, σ_z are outputs of fully-connected layers. Since back-propagation is not feasible through random sampling, reparameterization trick [18] is used to train the CVAE structure. The KL divergence loss is considered for regularization:

$$l_{KLD} = \mathcal{D}_{KL}(q_{\nu_r}(z|Y, X, f_{ego})|p_{\nu_p}(z|X, f_{ego})).$$

2) Concatenating layer: Apart from diver sample generation, we have additionally examined how well each vehicle conforms to the rules φ. For this purpose, we have bundled the encoded information of adjacent vehicles for each vehicle. Concatenating layer plays the role described above, which is implemented in several MLPs, and returns concatenated states $\tilde{c} = [c_{ego}, c_{lf}, c_{lr}, c_{cf}, c_{cr}, c_{rf}, c_{rr}]$ (Figure 2). c_{ego} use all encoded states of near vehicles V_{near}, and other terms in c beside c_{ego} only use three spatially adjacent encoded states. Notice that we do not apply interaction models presented in [7]–[9] because we only consider up to 6 near vehicles and the previous works are suitable for predicting pedestrians in crowded scenes.

C. Prediction Module

Prediction module predict both future trajectories $\hat{Y}$ and robustness slackness $\hat{r}$ for each vehicle by incorporating output of encoder module and interaction module. For given history trajectory X and the sampled latent variable z, the LSTM decoder generates future trajectory with respect to the distribution $p(Y|X, z) = \prod_{i=1}^{H_Y} p(s_{t+i}|X, z, s_{1:i-1})$,

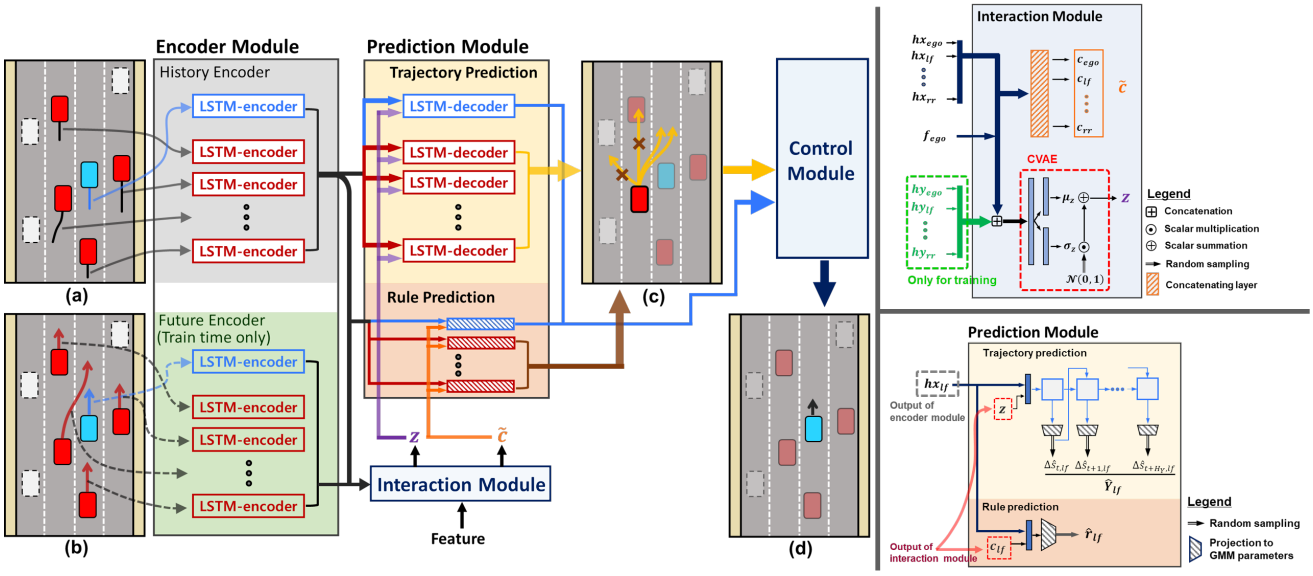

Fig. 2. Overall procedure for the proposed deep predictive control framework. (a, b) Past trajectories and future trajectories of vehicles are encoded in encoder module. Encoded information are feed into the interaction module, which captures the current situation. Interaction module produces stochastic latent variable z and concatenated states $\tilde{c}$ for each variables. The latent variable z helps to jointly predict plausible trajectories of vehicles, and concatenated states $\tilde{c}$ are for predicting rule information of each vehicle, which is the robustness slackness of STL formulas. (c) Combined with encoded states and sampled latent variable z, joint prediction on future trajectories are conducted. With predicted robustness slackness of STL formula, we can filter out invalid prediction candidates. (d) Control module finds out the valid control from the predictions through optimization process.

where $Y = \{s_{t+1}, ..., s_{t+H_Y}\}$. Notice that we model outputs of both trajectory and rule prediction parts as a Gaussian mixture model (GMM) to better represent the distribution of real data, and the input of the next LSTM cell is sampled from the GMM.

During the test phase, K_p prediction candidates are generated from the prior distribution $z^{(k)} \sim p_{\nu_p}(z|X, f_{ego})$, $i.e., \hat{Y}^{(1)}, ..., \hat{Y}^{(K_p)}$. We measure the robustness degree of each prediction candidates and filter out invalid ones which have lower value than the predicted robustness slackness $\hat{r}$. This process helps us achieve more accurate results and allows the controller to increase the safety of the control.

D. Control Module

In the control procedure, model predictive control (MPC) under signal temporal logic [2] is conducted. The cost function J is designed as quadratic so that the above optimization is now mixed integer quadratic programming.:

$$J(\mathbf{x}(\mathrm{x}_t, \mathbf{u}^{H,t}), \mathbf{u}^{H,t}) = \frac{1}{2}(\mathrm{x}_{t+H} - \mathrm{x}_g)^\intercal Q_x(\mathrm{x}_{t+H} - \mathrm{x}_g) + \mathbf{u}^{H,t\intercal} Q_u \mathbf{u}^{H,t}.$$

The goal point x_g is selected as the last state of the predicted trajectory $\hat{Y}_{ego}$ and modified to the middle point of the lane. Since the rule φ_5 is hard to be computed, we neglect the constraints related to φ_5 during the optimization process when the predicted robustness slackness $\hat{r}_{ego}(5))$ is below the certain threshold.

E. Implementation Details

We select an arbitrary vehicle as V_{ego} and set its surrounding vehicles as V_{near}. Three modules including encoder module, interaction module and prediction module are trained with the following loss function:

$$l_{train} = l_{KLD} + \frac{1}{K}\sum_{k=1}^{K}\sum_{i\in\{V_{ego},V_{near}\}} l_Y^{k,i} + l_r^{k,i},$$

where K is a sample number of latent variable z, $l_Y^{k,i}$ and $l_r^{k,i}$ refer negative log-likelihoods of GMM in both trajectory and robustness slackness prediction. During training phase in the LSTM decoder in prediction module with a rate of 10% the predicted value as the input into the next cell, otherwise the ground truth value from the training data.

VI. Experimental Results

We conducted a realistic simulation on the public Next-Generation Simulation (NGSIM) dataset [19]. The data in [19] contains the vehicle trajectory data on the US Highway 101 and the Interstate 80. We divided the NGSIM dataset into train and test parts.

The learning part of the proposed method except control module is implemented with tensorflow, and mini-batch based stochastic gradient descent is used to optimize the objective functions. We train our network with the following hyper parameters setting: minimum batch size (128), learning rate (0.001) and number of epochs (100).

Both MPC horizon H and future time horizon H_Y are set as 16, and history time horizon H_X as 8. Control module is implemented with Gurobi [20] as an optimization engine.

We have conducted simulation experiments on the test dataset (Fig. 3). In Fig. 3, 4 scenarios are shown in rows. The first four columns present the movement of the ego vehicle over time as snapshots, and the predicted robustness slackness of the ego vehicle is shown for one of the first four selected columns, which is marked as a yellow box. The ego vehicle V_{ego} is marked as blue and near vehicles V_{near} as

red, and we mark vehicles that are considered to affect the control of the ego vehicle as 'A' and 'F'. The ego vehicle is controlled in consideration of the STL rules, and its resulting trajectory is shown in the green solid line. We can see how the ego vehicle behaves with predicted robustness slackness of rules and predicted trajectories of surrounding vehicles (blue lines). In the first row (a), the ego vehicle decides to change to the left lane after the fast moving adjacent vehicle A is passed, due to the slowly moving front vehicle F. This is well represented in the predicted robustness slackness; the negative robustness slackness values of φ_1, φ_5 allow the ego vehicle to move to the right lane with above a certain speed. In the second row (b), it is similar to the first situation (a). But the ego vehicle changes the lane before the adjacent vehicle A passes because the vehicle A is slow, and this is shown in negative robustness slackness of φ_2 at the second time. The robustness slackness in the row (c) all robustness slackness values are positive, which leads the ego vehicle to keep the current lane with a moderate speed.

The effect of predicted robustness slackness on trajectory prediction of near vehicles is shown in Fig. 4. Trajectory prediction candidates, which are outputs of LSTM-decoder in prediction module, are shown in blue lines. Invalid trajectory candidates are shown in yellow line. In case of the vehicle marked as green box, all predicted robustness slackness values are positive, and this leads to exclude the lane-changing prediction which disobeys φ_1.

We evaluated our method in two perspectives; prediction accuracy and safety (collision avoidance).

A. Comparison: Prediction

In case of prediction, we have compared the proposed method with the following three methods: S-LSTM, S-CVAE, and M-GAN. S-LSTM refers naive LSTM encoder-decoder framework, and S-CVAE means LSTM encoder-decoder combined with CVAE [1]. Both S-LSTM and S-CVAE do not consider the interaction between vehicles; Surrounding vehicle information is not considered during training. M-GAN is a modified version of Social-GAN [8] in our problem; Unlike [8], m-GAN only consider surrounding vehicles V_{near} and V_{ego}.

The comparison result is shown in Fig. 5. Proposed* refers to the proposed approach without using the filtering process with respect to the robustness slackness. We fix number of prediction samples as 30, and also compare worst-case result among generated prediction samples. We use Average Displacement Error (ADE) as metric to measure the performance. ADE represents average *L2* distance between ground truth and prediction over all predicted time steps. From the results, we can see that considering the interaction of the surrounding vehicles can improve the performance of the prediction. Two results of Proposed and Proposed* show that robustness slackness significantly improves the accuracy of the prediction, especially in worst case. It seems that this is the reason why the proposed approach offers better prediction results than other methods that include M-GAN.

B. Comparison: Collision Avoidance

We conducted experiments of the proposed approach in the US highway 101 and the Interstate 80. The following the methods(LBMPC-STL, IL-Proposed) are compared with the proposed approach. LBMPC-STL [2] denotes learning-based MPC with STL constraints. Since LBMPC-STL does not predict future trajectories of surrounding vehicles, we use naive LSTM encoder-decoder network to predict future trajectories. IL-Proposed refers imitation learning approach that does not conduct MPC optimization in proposed approach; We directly apply trajectory prediction result of the ego vehicle. A total of 30 experiments were performed for each of the different starting positions for each track. Table I shows the number of collisions for each method. It definitely shows that the proposed approach is better performance in terms of safety. This is due to better prediction performance and MPC optimization process.

TABLE I
COLLISION EXPERIMENT

	Proposed	LBMPC-STL [2]	IL-Proposed
US101	2	5	6
I80	4	8	8

Number of collisions during a total of 30 experiments.

VII. CONCLUSION

In this paper, we have presented a deep predictive control approach to control for autonomous driving problem. Our novelty lies in combination of deep learning and model predictive control approach. Rather than strict compliance with all rules, the proposed method follows each rule more efficiently, including disobeying certain rules so that it can solve dilemma situations when all rules can not be satisfied. By performing prediction considering interaction between vehicles, the proposed method can generate more accurate prediction results, especially on the worst case.

REFERENCES

[1] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in neural information processing systems*, 2015, pp. 3483–3491.

[2] K. Cho and S. Oh, "Learning-based model predictive control under signal temporal logic specifications," in *Robotics and Automation (ICRA), 2018 IEEE International Conference on*. IEEE, 2018, pp. 7322 – 7329.

[3] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *FORMATS/FTRTFT*, vol. 3253. Springer, 2004, pp. 152–166.

[4] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals." in *FORMATS*, vol. 6246. Springer, 2010, pp. 92–106.

[5] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336–345.

[6] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[7] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[8] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

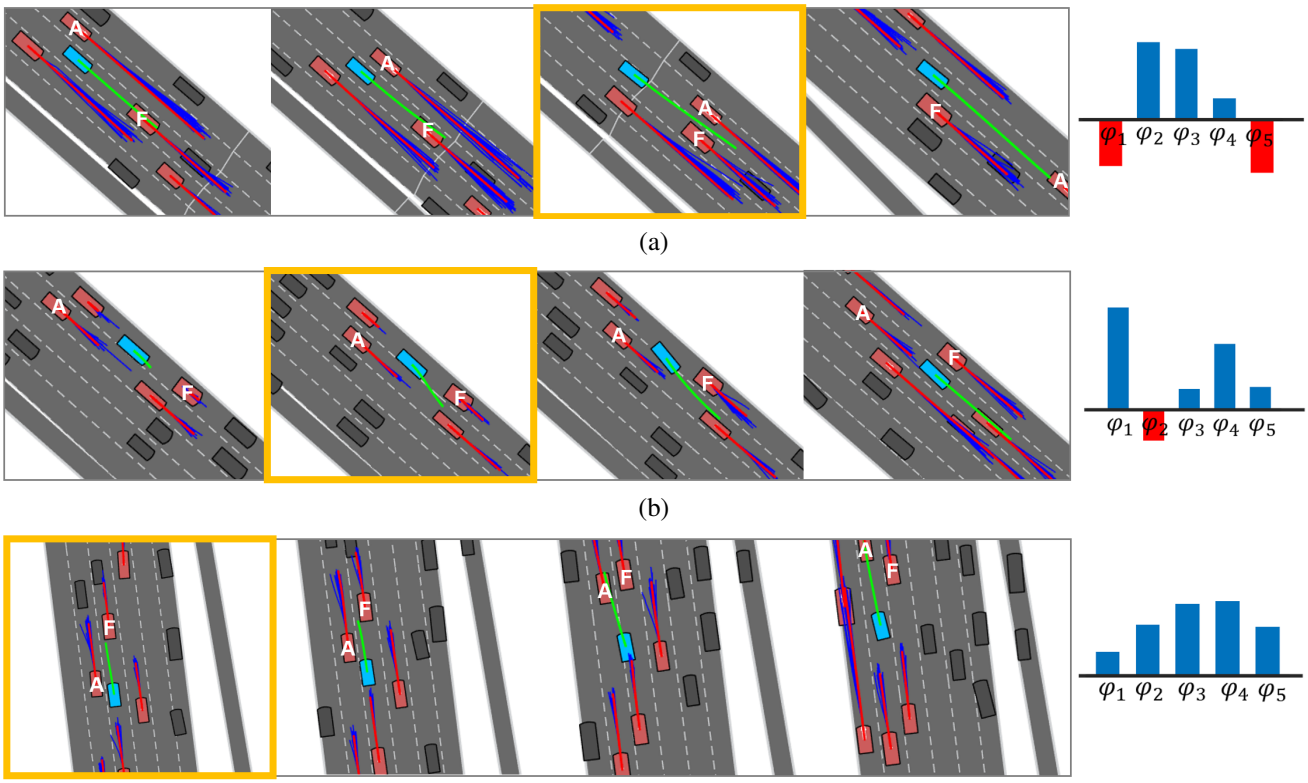

Fig. 3. Simulation results on the US Highway 101 (a,b) and the intersection 80 (c). The ego vehicle V_{ego} is marked as blue and the near vehicles V_{near} as red. The first four columns show the movement of the ego vehicle over time, and the last column represent the predicted robustness slackness of the ego vehicle for one of the first four selected columns (yellow box). Computed trajectory of the proposed method is shown in green solid line. Blue lines are generated trajectory prediction samples of the near vehicles while red line is ground truth trajectory of the near vehicles.

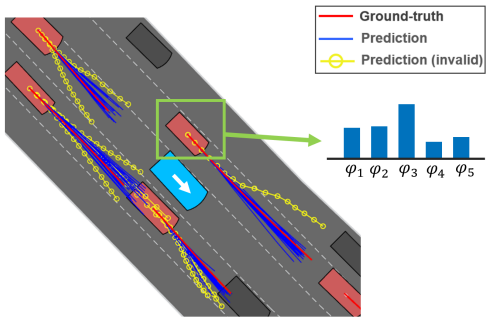

Fig. 4. Exclusion of the invalid prediction candidates. Among prediction trajectory samples (blue line), predicted robustness slackness exclude the invalid prediction samples (yellow line).

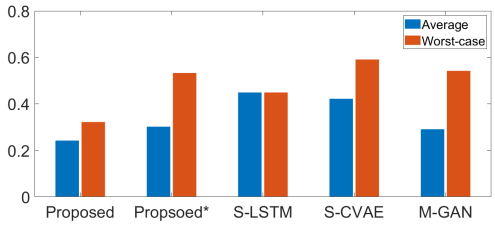

Fig. 5. Prediction comparison. Average and worst-case comparisons are made among trajectory prediction samples.

[9] Y. Xu, Z. Piao, and S. Gao, "Encoding crowd interaction with deep neural network for pedestrian trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5275–5284.

[10] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *Proc. of the IEEE Conference on Decision and Control*, Dec. 2008, pp. 2117–2122.

[11] Y. Kwon and G. Agha, "Ltlc: Linear temporal logic for control," *Hybrid Systems: Computation and Control*, pp. 316–329, 2008.

[12] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based control of nonlinear systems with linear temporal logic specifications," in *Proc. of the IEEE Conference on Robotics and Automation*, 2014, pp. 5319–5325.

poral logic motion planning," in *Proceedings of the 18th International*

[13] S. C. Livingston, E. M. Wolff, and R. M. Murray, "Cross-entropy tem-

Conference on Hybrid Systems: Computation and Control. ACM, 2015, pp. 269–278.

[14] K. Cho, J. Suh, C. J. Tomlin, and S. Oh, "Cost-aware path planning under co-safe temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2308–2315, 2017.

[15] V. Raman, A. Donzé, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *Proc. of the IEEE Conference on Decision and Control*, 2014, pp. 81–87.

[16] D. Sadigh and A. Kapoor, "Safe control under uncertainty," *arXiv preprint arXiv:1510.07313*, 2015.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[19] J. Colyar and J. Halkias, "Us highway 101 dataset," *US Highway 101 Dataset, FHWA-HRT-07-030*, 2007.

[20] G. Optimization, "Inc.,"gurobi optimizer reference manual," 2014," *URL: http://www. gurobi. com*, 2014.