SUPPLEMENTARY MATERIAL

CMTS for Rare Driving Scenarios Synthesis

Wenhao Ding, Mengdi Xu and Ding Zhao

A Neural Network

A.1 Network Structure

The dimension of all layers in our model is shown in three tables. The structure of data encoder, condition encoder and decoder are respectively shown in Table. 1, Table. 2 and Table. 3. The dimension of the input sequence is 50×4. The length of the sequence is 50 and for each timestamp, there is one data array (x_1, y_1, x_2, y_2).

Table 1: Structure of Data Encoder

Description		Output dimension
Sequence 1 (x_1, y_1)	Sequence 2 (x_1, y_1)	$50 \times 2, 50 \times 2$
Concatenated sequence (x_1, y_1, x_2, y_2)		50×4
Input layer (MLP)		4×64
ReLU		-
RNN Encoder (GRU)		1×128
Linear Encoder		128×20
z (mean)	z (logvar)	$1 \times 32, 1 \times 32$

Table 2: Structure of Condition Encoder

Description		Output dimension
Input image		$1 \times 128 \times 128$
Convolution Layer 1 (kernel size: 3×3)		$16 \times 64 \times 64$
ReLU		-
Convolution Layer 2 (kernel size: 3×3)		$32 \times 32 \times 32$
ReLU		-
Convolution Layer 3 (kernel size: 3×3)		$64 \times 16 \times 16$
ReLU		-
Convolution Layer 4 (kernel size: 3×3)		$64 \times 8 \times 8$
ReLU		-
Linear Layer 1 (mean)	Linear Layer 1 (logvar)	$1 \times 256, 1 \times 256$
Linear Layer 2 (mean)	Linear Layer 2 (logvar)	$1 \times 32, 1 \times 32$
y (mean)	y (logvar)	$1 \times 32, 1 \times 32$

A.2 Training Details

We use Pytorch[1] to implement our framework and all models are trained on a NVIDIA RTX2080Ti GPU. The training loss consists of three parts: KL divergence, reconstruction loss, and fusion loss. All of them are displayed in Fig .1 as well as the total loss. All hyper-parameters in our experiments are listed in Table. 4.

[1] `https://pytorch.org`

Table 3: Structure of Decoder

Description		Output dimension
z (shared)	z (shared)	$1 \times 32, 1 \times 32$
Hidden layer (Linear)	Hidden layer (Linear)	$1 \times 256, 1 \times 256$
ReLU		-
Hidden2StartPoint Layer (Linear)	Hidden2StartPoint Layer (Linear)	$1 \times 32, 1 \times 32$
ReLU		-
Start point 1	Start point 2	$1 \times 2, 1 \times 2$
Input layer (MLP)	Input layer (Linear)	$1 \times 32, 1 \times 32$
ReLU		-
Concatenate	Concatenate	$1 \times 64, 1 \times 64$
RNN Decoder 1 (GRU)	RNN Decoder 2 (GRU)	$1 \times 256, 1 \times 256$
Linear Decoder 1	Linear Decoder 2	$1 \times 2, 1 \times 2$
Ponit 1	Ponit 2	$1 \times 2, 1 \times 2$
Collector	Collector	$50 \times 2, 50 \times 2$

A.3 Baseline Details

We use MTG [1] as our first baseline, whose structure can be found in the supplementary material of [1]. We trained this model with both the original dataset and collision dataset. The second baseline we used is the perturbation method proposed in [2], which fixes the start and end points and adds perturbation to the mid point. The disturbance is uniformly at random in the range $[-0.5, 0.5]$ meters in both axes.

We use three trajectory prediction methods in our experiments:

- Vanilla-LSTM: we implement a simple *seq2seq* structure to predict the future trajectory and use MSE as the loss function.
- Social-LSTM: we modify the data loader part in this code repo[2].
- CS-LSTM: we modify the data loader part in this code repo[3].

Table 4: Hyper-parameters of our model

Notation	Value	Description
lr	0.001	learning rate
B	512	batch size
α	1	weight of reconstruction error
β	0.5	weight of KL divergence
γ	0.1	weight of fusion loss
z_{dim}	32	dimension of the latent code
c_{dim}	128	dimension of the road map image
o_{len}	20	observe length in trajectory prediction
λ_g	0.3	interpolating parameter in generating stage

B Dataset Generation Settings

B.1 Collision Dataset Generating Rules

We generated the collision dataset from the original dataset with simple rules. There are two sequences in one data sample, representing two vehicles. Firstly, we randomly select a collision point in sequence 1 and then translate sequence 2 to make sure that these two sequences have a collision on the pre-defined point, which means the index

[2] https://github.com/quancore/social-lstm

[3] https://github.com/nachiket92/conv-social-pooling

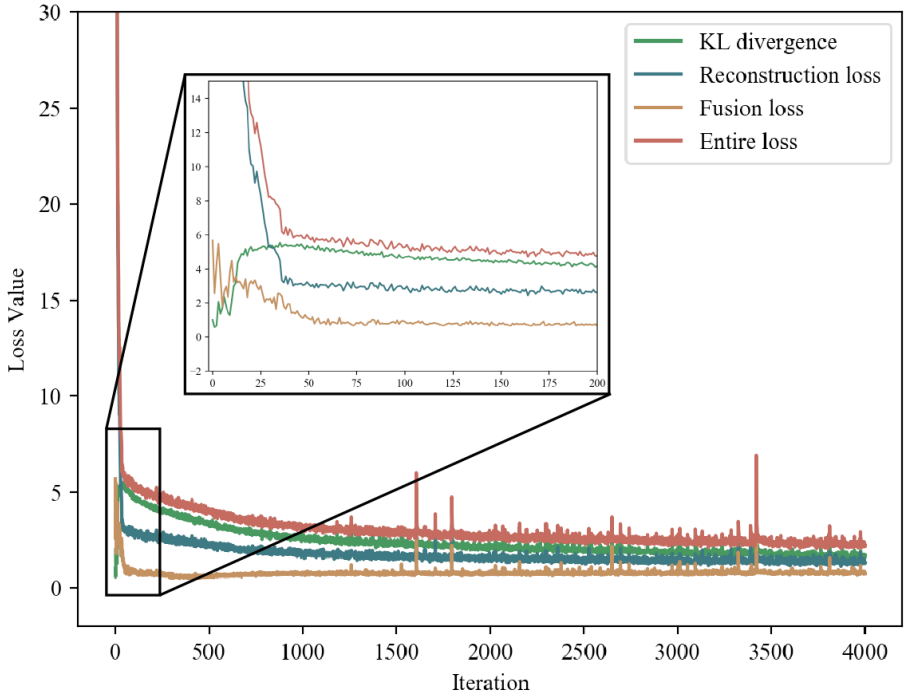

Figure 1: Training loss of our CMTS.

of the point should be the same for both sequences. Secondly, we randomly rotate sequence 2 until all points in sequence 2 satisfy the road constraint.

B.2 Risky Scenarios Design Rules

Based on experience in real traffic, we design 6 risky scenarios, the first 3 of which are more likely to happen to daily life and the rest 3 are more fatal to the traffic participations. The descriptions of these 6 scenarios are as follow (V1 is the ego vehicle, V1 is the short of vehicle 1 and V2 is the short of vehicle 2):

- V1 is driving on a lane, while V2 is driving in the same direction on a neighbor lane. V2 suddenly cuts in line in front of V1. If a human driver encounters this situation, he will slow down to avoid a collision.
- V1 and V2 are driving on the same lane and V2 is in front of V1. Both vehicles are driving fast while V2 suddenly slows down. If a human driver encounters this situation, he will also slow down to avoid a rear-end-collision.
- V1 is going straight to an intersection and V2 is going to turn right. When the road is narrow, these two vehicles are very likely to have a collision if V1 does not change to the left lane.
- V1 is driving on a lane, while V2 is driving in the opposite direction. Due to some unknown reason, V2 rushes to the opposite lane towards to V1. Although this is a very risky situation and rarely happens in real life, a reasonable response for V1 is turning left and slow down.
- V1 is driving in the middle of an intersection, while V2 breaks the red light and drives towards V1.
- V1 is turning left in an intersection, while V2 breaks the red light and drives towards V1.

Although these examples cannot cover all risky scenarios, they indeed give some demonstrations of near-miss scenarios that is rarely included in real-life dataset. It is because of this, that we propose to automatically generate these risky scenarios.

C More Experiment results

More results of trajectory prediction in risky scenarios are shown in Fig. 2.

References

[1] W. Ding, W. Wang, and D. Zhao, "Multi-vehicle trajectories generation for vehicle-to-vehicle encounters," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[2] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[3] 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.

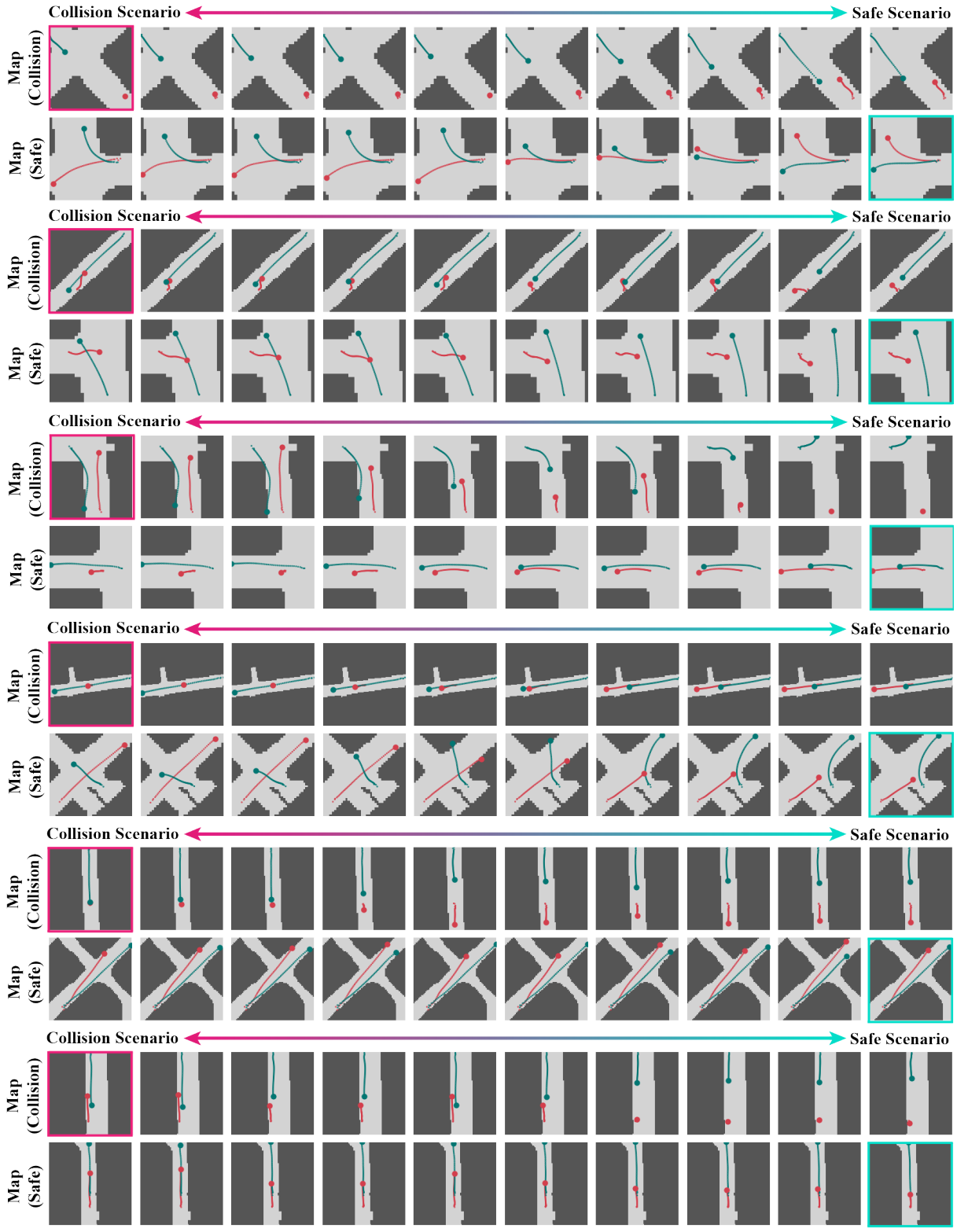

Figure 2: Trajectory prediction results in 6 risky scenarios with CS-LSTM [3]. Red points and green points represent the history and predicted trajectory, respectively. Bigger green points are last predictions.