Joint-Multipath++ for Simulation Agents

Wenxi Wang Haotian Zhen

Abstract

The field of autonomous driving technology is experiencing rapid advancements, and simulation has emerged as a critical tool for enhancing its robustness and scalability. Otherwise, the simulation agents face challenges in generating intelligent reactions. To enhance the realism of the simulator, we propose a solution named Joint-MultiPath++ ranked 2nd in the Waymo Sim Agents Challenge 2023. Our source code is publicly available on GitHub[1].

1. Introduction

Simulation has become crucial for advancing the development of autonomous driving vehicles (ADVs). A simplistic approach to simulation involves replaying the sensor data captured by the ADV in the real world, making minor software modifications, and observing the outcome of the scenario. However, this approach presents inherent challenges, such as the lack of responsiveness of playback objects to ADV behavior changes. Consequently, simulation agents are necessary to realistically react to our actions. In this paper, we propose a solution named Joint-MultiPath++ which efficiently scores the 2nd on Waymo Sim Agents Challenge 2023.

2. Related Work

To enhance the realism of trajectories in simulators, a typical model comprises an encoder and a decoder. The encoder is for encoding both the map information and the historical trajectory data.

When it comes to map information, there are two primary methods that are commonly used. The raster method[1, 2, 6, 8, 9, 11] represents the world by creating a series of images from a top-down perspective. It simplifies the process as it combines different types of input information, such as road configurations, agent state history, and spatial relationships, into a multi-channel image. This input modality is naturally aligned with Convolutional Neural Network (CNN) model inputs. Instead, the polyline method [4, 7, 10] presents a different approach for representing curves such as lanes, crosswalks, and boundaries. It achieves this by breaking them down into smaller linear segments. This method is more efficient and concise because road networks tend to have sparse characteristics.

The decoder is designed to accurately predict multimodal trajectories. This prediction process can be divided into two approaches: one-by-one prediction and joint prediction. One-by-one prediction[5, 10, 12] adopts the agent-centric strategy and predicts each agent individually, without taking into account the interactions among their future trajectories. Recognizing the limitations of this method, we have opted for the joint prediction approach[3], where we simultaneously predict the trajectories of different agents, capturing their interactions and dependencies. By employing the joint prediction method, we aim to enhance the accuracy and comprehensiveness of our predictions.

3. Method

JointMultiPath++ means it is a joint prediction model. The network structure is shown in Figure 1. In the original version of Multipath++[10], the model inference can only predict the future trajectory of one agent at a time, and there is a serious drawback if we directly apply Multipath++[10] to the Sim Agent Challenge: there is no interaction between the corresponding modalities of each agent. In fact, due to the concept of anchor in Multipath++[10], the *mth* modality of each agent always converges to the same shape, which leads to potential collision problems.

To address this point, we made corresponding modifications in the data preprocessing, encoder and decoder, so that the model can obtain 32 future world distributions by one inference.

3.1. Data preprocessing

We have made three adjustments to the data preprocessing based on Multipath++[10]:

1. There is no distinction between the target agent and other agents. In each scene, n agents are fixedly selected for prediction. The agent selecting method includes only selecting available agents, selecting agents sorted by available time and selecting all agents. And we use zero padding when the number of seleted agents is less than n.

[1] https://github.com/wangwenxi-handsome/Joint-Multipathpp

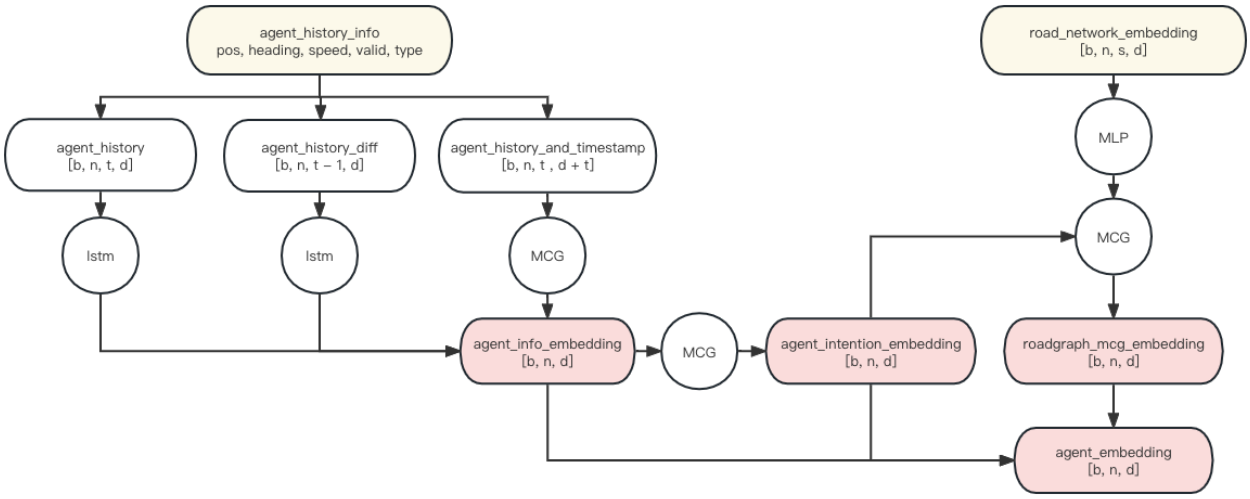

Figure 1: The architecture of Joint-Multipath++ Encoder(Encoder only). Every MCG block is composed of three layers. b is batch size which is equal to the number of scenes. n is the number of modeled agents, t is the history timestamp. s is the polyline number for each agent. d is the hidden dimension of embeddings. Different embeddings have different hidden dimensions. Please refer to 4.1 for default values.

2. Transform the coordinates and yaws of all agents into the ADV's coordinate frame(a scene-centric encoding).

3. For each agent, select the nearest s polylines from their current positions as map features.

3.2. Encoder

Like Multipath++[10], the agent history encoder extracts information in the time dimension in three ways, as follows:

$$\begin{aligned} &agent\_info\_embedding_i = Concat(\\ &\quad LSTM_1(agent\_history\_info_i), \\ &\quad LSTM_2(agent\_history\_info\_diff_i), \\ &\quad MCG(agent\_history\_info_i), \\ &) \quad i \in [1, n] \end{aligned} \tag{1}$$

The shape of $agent\_history\_info$ is (b, n, d_1). $agent\_history\_info_i$ is defined as:

$$\begin{aligned} &agent\_history\_info_i = Concat(\\ &\quad x_i, y_i, z_i, yaw_i, speed_i, width_i, length_i, valid, \\ &) \quad i \in [1, n] \end{aligned} \tag{2}$$

And $agent\_history\_info\_diff_i$ is defined as:

$$\begin{aligned} &agent\_history\_info\_diff_i(j) = \\ &\quad agent\_history\_info\_diff_i(j) - \\ &\quad agent\_history\_info\_diff_i(j-1) \quad j \in [1, t] \end{aligned} \tag{3}$$

We use MCG block proposed in Multipath++[10] to model the interaction between agents. The specific method is to take out the *agent_info_embedding* of each agent and perform MCG calculations with all other agents.

$$\begin{aligned} &agent\_intention\_embedding_i = Concat(\\ &\quad agent\_info\_embedding_i, \\ &\quad MCG(agent\_info\_embedding_i, \\ &\qquad agent\_info\_embedding_i), \\ &) \quad i \in [1, n] \end{aligned} \tag{4}$$

The shape of $agent\_intention\_embedding$ is (b, n, d_2).

For the map features, we use MLP as polyline encoder and use a MCG block to capture the interaction between the trajectory and the map features.

$$\begin{aligned} &roadgraph\_mcg\_embedding_i = MCG(\\ &\quad agent\_intention\_embedding_i, \\ &\quad MLP(road\_network\_embedding_i), \\ &) \quad i \in [1, n] \end{aligned} \tag{5}$$

The shape of $roadgraph\_mcg\_embedding$ is (b, n, d_3).

Finally, we concatenate all three embeddings to obtain the *agent_embedding*.

$$\begin{aligned} &agent\_embedding_i = Concat(\\ &\quad agent\_info\_embedding_i, \\ &\quad agent\_intention\_embedding_i, \\ &\quad roadgraph\_mcg\_embedding_i, \\ &) \quad i \in [1, n] \end{aligned} \tag{6}$$

3.3. Decoder

In the Decoder, we discard the concept of anchor in Multipath++[10] and map the *agent_embedding* to 32 future trajectories of each agent by multi-layer MLP, which

Table 1: Detailed evaluation of our model on test set of Waymo Open Motion Dataset

Metric Name	Value
Realism meta-metric	0.4888
Linear Speed Likelihood	0.4318
Linear Acceleration Likelihood	0.2304
Angular Speed Likelihood	0.5149
Angular Acceleration Likelihood	0.4521
Distance To Nearest Object Likelihood	0.3440
Collision Likelihood	0.4198
Time To Collision Likelihood	0.8127
Distance To Road Edge Likelihood	0.6394
Offroad Likelihood	0.5830
minADE	2.0517

includes x, y, z and yaw of the agent at every timestamp.

$$Trajectory = MLP_2(MLP_1(agent\_embedding) \\ .reshape(b, n, 32, d_4)).reshape(b, n, 32, 80, 4) \tag{7}$$

The role of MLP_1 is to increase the dimensionality of features in order to generate multimodal predictions, the shape of MLP_1 output is $(b, n, 32 * d_4)$. The role of MLP_2 is to decrease feature dimensionality to map to the corresponding trajectory, the shape of MLP_2 output is $(b, n, 32, 80 * 3)$.

The loss function incorporates the fused weights of the minADE of both the trajectory's coordinates and yaw.

$$Loss = \underset{b,n}{mean}\ \underset{m}{min}\ \underset{k,t}{mean}\ (trajectory(k) - groudtruth(k))^2 \\ k \in [0, 3] \tag{8}$$

4. Experiments

4.1. Implementation Details

Architecture details. The default model is around 27MB and is composed of encoder and decoder. In the encoder, the road map is represented as polylines, where each polyline contains up to 5 map points. Each agent selects the 128 closest polylines from their current positions as map features. For the features of agents, the hidden dimensions of *agents_info_embedding*, *agents_intention_embedding*, *roadgraph_mcg_embedding* are 256, 128 and 128, respectively. The dimension of final *agent_embedding* is set to 512 to get a large model capacity for such a largescale Waymo Open Motion Dataset(WOMD). In the decoder, a three-layer MLP is adopted with agent embedding and increases the hidden dimension to 1024 which will be divided into 32 parts. After that, another three-layer MLP is adopted to generate 32 multimodal future trajectories.

Training details. Our Joint-Multipath++ model is trained end-to-end by Adam optimizer with a learning rate of 0.0001(no decay) and batch size of 128 scenes. We trained the model with 200 epochs on one A100 GPU without any data augmentation or model ensemble method. In the training and validation process, we select first 16 agents sorted by the valid length of history trajectory. In the test process, we select all 128 agents to predict their trajectories.

4.2. Results

Using the mapping data and history tracks from the previous second, 32 realistic future trajectories for each agent present in the 8 seconds should be generated. Table 1 shows the values of eleven metrics which evaluate the model's performance. Due to the joint generation of different agents' trajectories, our model achieves excellent performance on interactive metrics.

References

[1] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956. PMLR, 2018. 1

[2] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 1

[3] Hao Cheng, Mengmeng Liu, Lin Chen, Hellward Broszio, Monika Sester, and Michael Ying Yang. Gatraj: A graph-and attention-based multi-agent trajectory prediction model. *arXiv preprint arXiv:2209.07857*, 2022. 1

[4] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. 1

[5] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15303–15312, 2021. 1

[6] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8454–8462, 2019. 1

[7] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *Com-*

puter Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16, pages 541–556. Springer, 2020. 1

[8] Francesco Marchetti, Federico Becattini, Lorenzo Seidenari, and Alberto Del Bimbo. Mantra: Memory augmented networks for multiple trajectory prediction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7143–7152, 2020. 1

[9] Charlie Tang and Russ R Salakhutdinov. Multiple futures prediction. *Advances in neural information processing systems*, 32, 2019. 1

[10] Balakrishnan Varadarajan, Ahmed Hefny, Avikalp Srivastava, Khaled S Refaat, Nigamaa Nayakanti, Andre Cornman, Kan Chen, Bertrand Douillard, Chi Pang Lam, Dragomir Anguelov, et al. Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7814–7821. IEEE, 2022. 1, 2

[11] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 1

[12] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning*, pages 895–904. PMLR, 2021. 1