An Imitation Learning Method with Data Augmentation and Post Processing for Planning in Autonomous Driving

Weitao Xi Liangchao Shi Guangzhi Cao
Pegasus Technology Holdings Inc.
{xiweitao, shiliangchao, gary.cao}@pegasus.tech

Abstract

Motion planning with imitation learning in autonomous driving is challenging, because of the complexity of scenarios and the distribution difference between training data and actual driving. In this paper, we propose an novel imitation learning method that incorporates data augmentation and post-processing techniques. Our model utilizes recent advance in motion prediction, such as vectorized map representation, transformer encoder and decoder, and GMM output with predefined initial queries. Data augmentation is used to generate data which deviates from expert trajectories. In post-processing, we employ a low planning rate to mitigate error accumulation, and a collision detector to trigger timely replanning. Finally, our method achieved 3rd place in the CVPR 2023 Autonomous Driving Challenge - Track 4 Nuplan Planning.

1. Introduction

Recent advancements have been observed in the field of motion prediction for autonomous driving, with numerous techniques proposed to enhance performance. Vector map representation for HD maps was proposed and achieve better performance than rastering presentation [3]. In order to fuse a heterogeneous mix of modalities, such as agents' past trajectories, static road information, and time-varying traffic light information, the transformer encoder has been extensively employed to encode various types of input simultaneously [5]. In modeling the multi-modality of output trajectories, the transformer decoder cross attend appropriate queries, such as the target position or trajectory modes, with scene embedding from the encoder to generate trajectories [4, 6]. Notably, the Wayformer has combined these techniques and achieved state-of-the-art performance [5].

Pure imitation learning proves to be insufficient for autonomous driving [1]. For long series decision-making problems, imitation learning tends to suffer from issues related to error accumulation and distribution mismatch. These issues can be alleviated through data augmentation and a decreased planning rate.

Nuplan is the world's first large-scale planning benchmark for autonomous driving [2]. The Nuplan planning challenge is consisted of three modes: open-loop, closed-loop with non-reactive agents, and closed-loop with reactive agents. The open-loop mode bears similarities with the motion prediction problem, while the closed-loop modes align more with the standard motion planning problem.

Our approach leverages imitation learning, supplemented by data augmentation and post-processing. Our model use vectorized map representation. Transformer encoder is employed to fuse different types of input feature. Latent query attention is used to reduce computational cost. To represent multi-model output, Gaussian mixture model (GMM) with predefined initial queries is used. We apply random perturbation to the current state of the ego to augment data. To mitigate error accumulation, we use a lower planning frequency. Moreover, we employ collision detection and in-map detection to trigger re-planning and select an optimal trajectory among candidate trajectories.

2. Method

2.1. Input Feature

Ego states, with dimensions $[1, 1, D_{ego}]$, encapsulate the current kinematic of the ego vehicle. Unlike most methods that employ both current and past ego states, we opt to use solely the current kinematic states to simplify the data augmentation process, discussed later in 2.3. The feature dimension encompasses information like position, speed, acceleration, heading, heading rate, and steering angle.

Agent states incorporate information concerning surrounding agents, carrying the dimensions $[T, S_{agent}, D_{agent}]$. Here, T and S_{agent} are pre-defined constant. When the number of surrounding agents is less than S_{agent} or an agent has fewer time-steps than T, zeros are padded and a mask is added to the encoder. Conversely, if the number of surrounding agents exceeds S_{agent}, the agents are filtered based on their current distance to the ego

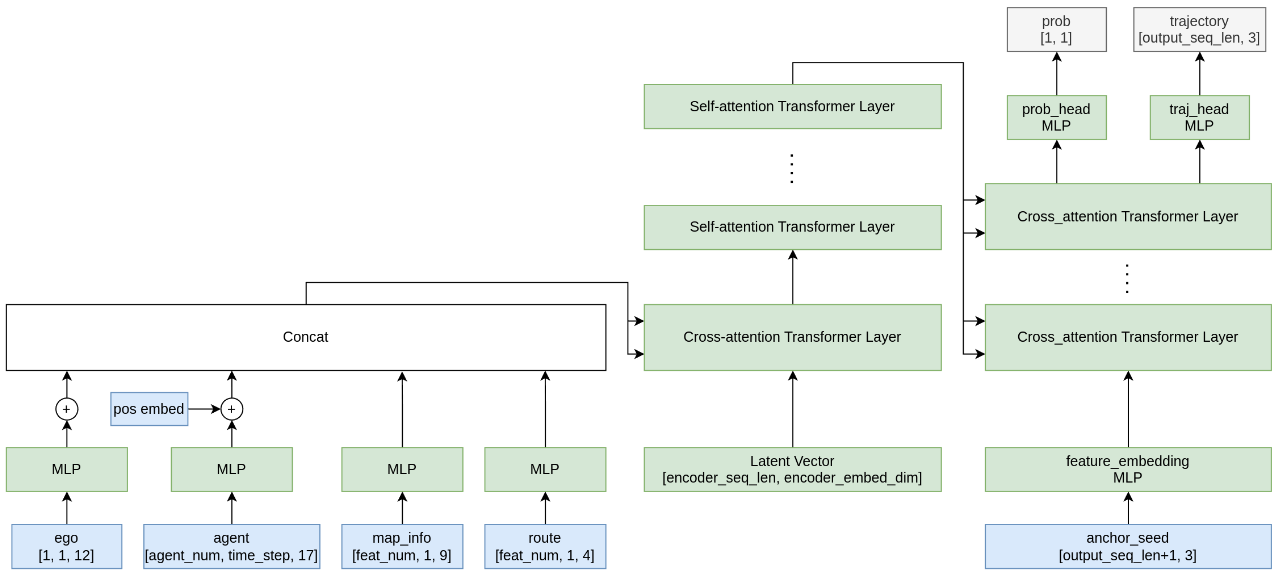
prob
[1, 1]
trajectory
[output_seq_len, 3]
prob_head
MLP
traj_head
MLP
Self-attention Transformer Layer
Self-attention Transformer Layer
Cross_attention Transformer Layer
Cross_attention Transformer Layer
Concat
Cross-attention Transformer Layer
pos embed
MLP
MLP
MLP
MLP
Latent Vector
[encoder_seq_len, encoder_embed_dim]
feature_embedding
MLP
ego
[1, 1, 12]
agent
[agent_num, time_step, 17]
map_info
[feat_num, 1, 9]
route
[feat_num, 1, 4]
anchor_seed
[output_seq_len+1, 3]

Figure 1. The general model structure.

vehicle.

Map information is organized via vector representation with the dimensions $[1, S_{map}, D_{map}]$. In practice, we categorize map information into several types, with each type structured into a tensor. These types include lane center, boundary, stop region, and crosswalk. In the Nuplan dataset, traffic light information is embedded within the lane center tensor. Padding and filtering follow same rules in agent state.

Route information follows a similar organizational structure as the lane center, having tensor dimensions $[1, S_{route}, D_{route}]$. In the Nuplan dataset, an additional information called mission goal representing a long term target position is also added to input feature.

2.2. Model

Our neural network model, as depicted in Figure 1, follows the same architecture in Wayformer [5]. Specifically, we employ an early fusion encoder to amalgamate different modalities. To reduce computational costs, latent query attention is implemented. Details are explained in the Wayformer paper.

Position embedding is only added to time dimensions in agent state tensor using sinusoidal embedding. No position embedding is added to represent spatial relationship, since all features in both the agent and map tensors, which comprise start and end points, inherently capture these relationships.

Anchor seeds are predefined trajectories which will not updated in training. In previous studies [5, 6], anchor seeds are calculated by k-means method from all ego trajectory data. In our implementation, we tried manually designed trajectories characterized by constant speed and steering and learned trajectories by k-means. Our experiments indicate that both methods achieve similar results.

Loss function comprises both classification loss and regression loss. For the regression loss, we use the L2 distance between expert trajectory and output trajectory. In terms of the classification loss, we have two options for target index. One is using the index of the closest output trajectory from the ground truth trajectory, and the other is using the index of the closest trajectory seed from the ground truth trajectory. Our experiments shows that using the index of the closest trajectory seed yields better performance.

2.3. Data Augmentation

Original data only contains expert driving data. If current ego state deviates from expert trajectory, policy should be capable to drive the vehicle back to expert trajectory.

Past and current trajectory of the ego vehicle demonstrate how the vehicle reach the deviated position and it influences model input. Generating plausible past perturbations is not straightforward. Past trajectories have many possibility and past states at different times are correlated, as shown in Figure 2 (a). It is not a good idea to directly add random noise to past states. To circumvent this issue, we find that past ego states are not important in our problem. Provided the current kinematic states are accurate — a condition typically met for the ego vehicle — past states offer minimal supplementary information for decision-making. Models, with or without past ego states, can achieve similar results in open-loop problems. Therefore, we omit past ego states from the input tensor and only consider current ego states. Current state perturbation can be easily achieved by

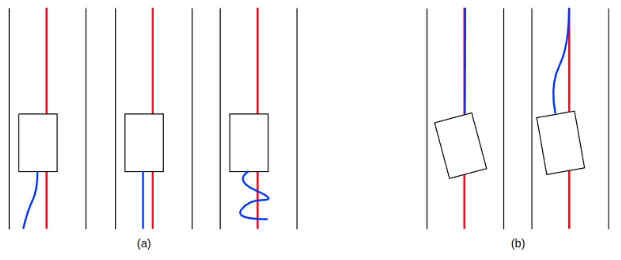

Figure 2. Methods to generate reasonable augmented data. Blue lines are expert trajectories. (a) The left and middle graphs show different possibilities of past trajectories, and the right graph shows a problematic past trajectory. We can not simply add noise in past trajectory. We avoid this problem by not using it. (b) Two possible future target trajectory. In the left graph, we simply use original target trajectory. In the right graph, we generate a smooth trajectory converging to expert trajectory using QP optimization. Both methods achieve similar performance.

adding a random value.

Future trajectory of the ego vehicle indicates how to drive the vehicle back to expert trajectory and serves as the target label. As depicted in Figure 2 (b), several options are available to generate future trajectories for a deviated current state. Our experiments demonstrate that simply using the original ground truth trajectory can yield satisfactory performance.

2.4. Post Processing

Despite the incorporation of data augmentation enabling the model knowing how to drive the vehicle back to the expert trajectory, it remains susceptible to the neural network's error accumulation. Zero error is an uncommon occurrence in supervised learning scenarios. Each planning step incurs a small discrepancy from the expert data. In the context of high-frequency decision-making, this error can accumulate rapidly. In the default settings of Nuplan, a new trajectory is planned every 0.1 second. To mitigate this issue, we decrease the planning frequency to a 2 second interval.

However, a lower planning frequency may result in additional collisions due to the planner's delayed reaction to changes in the environment. To rectify this, we introduce collision detection and in-map detection for the planned trajectory. If a collision or out-of-map condition is detected, the planner triggers an immediate replanning process.

Our neural network model is capable of concurrently outputting multiple trajectories. Typically, we select the trajectory with the highest score. Nevertheless, if this trajectory results in a collision or out-of-map scenario, we resort to other candidate trajectories. If all candidate trajectories yield a collision or out-of-map condition, we utilize a predefined fallback trajectory that implements maximum possible braking.

	overall	1	2	3
Test Phase	0.85	0.88	0.82	0.85
Warm-up Phase	0.92	0.93	0.90	0.92

Table 1. Final results in Nuplan planning challenge

3. Experiment

3.1. Implementation Details

Input feature details. For agent states, the maximum number of agents is capped at 64, incorporating 4 past agent states with a time gap of 0.4 seconds. For map features, the maximum number of features allowed for the lane center, boundary, stop region, crosswalk, and route lane center are set to 512, 256, 128, 64, and 512 respectively. The sampling density for map features is 4 meters per point. All input features are normalized.

Model details. The model employs 4 transformer encode layers for context encoding and 4 transformer decode layers for the generation of multi-modal trajectories. To reduce computational cost, latent query attention is employed with sequence length 256. Both the encoder and decoder have 256 embedding dimensions, 16 heads, and 512 intermediate hidden dimensions. We utilize 27 different anchor seeds, with a variation of 3 speeds and 9 heading rates.

Training details. Our model is trained using the Adam optimizer. A one-cycle learning rate schedule is employed, with a maximum learning rate set at 2e-4 and a division factor of 10. The model is trained over 10 epochs.

Data augmentation and post processing details. Data is augmented with probability 0.5. Uniform random noises are added to current ego states including position, speed, acceleration, heading, steering angle. In every 2 second, ego trajectory will be updated. In collision detection, we assume that other agents move with constant speed. We detect collision for maximum 2 seconds into the future.

In Nuplan planning challenge, we need use a single planner both in closed-loop scenario and open-loop scenario. In open loop scenario, low planning frequency will deteriorate the result. On the other hand, the behavior of closed loop controller is mainly determined by the beginning part of planned trajectory. To achieve good performance on both open loop and closed loop challenge, we concatenate the beginning part of previous trajectory with the end part of new trajectory calculated based on current input.

3.2. Main Results

We achieved 3rd place in Nuplan planning challenge. The final results for test phase and warm-up phase are presented in Table 1.

Additionally, we carried out several ablation experiments on Nuplan validation dataset. As shown in Table 2, we

Method	closed-loop non-reactive
pure imitation	0.77
+ data augmentation	0.78
+ post processing	0.88
+ both	0.91

Table 2. Experiment results on validation dataset for data augmentation and post processing

	open-loop
trajectory seed	0.91
output trajectory	0.88

Table 3. Experiment results on validation dataset for comparing the target index choice in classification loss.

	open-loop
with ego past states	0.916
no ego past states	0.912

Table 4. Experimental results on the validation dataset for determining whether the use of past ego states is effective

find post processing can significant improve performance in closed loop challenge. In Table 4, we show that not using ego past states in input feature get almost same result compared with using past states in open loop challenge. As shown in Table 3, using the index of the closest anchor seed instead of the index of the closest output trajectory in the classification loss significantly improved performance. This improvement could be due to the overcoming of the modal collapse typically encountered in GMM.

During data augmentation, we attempted to generate a smooth future trajectory that converges towards the expert trajectory, with the expectation of better performance than simply using the expert trajectory as shown in Figure 2. We generated these trajectories using a Quadratic Programming (QP) solver, taking the expert trajectory as the reference line and the perturbed current ego states as initial conditions. However, no performance improvement is observed.

4. Conclusion

In this paper, we have presented an imitation learning method for autonomous driving planning. We have applied data augmentation and post-processing techniques to mitigate issues observed in the closed-loop challenge. Our method has proven effective, achieving 3rd place in the 2023 Autonomous Driving Challenge - Track 4 Nuplan Planning.

References

[1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 1

[2] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021. 1

[3] 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

[4] 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

[5] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. *arXiv preprint arXiv:2207.05844*, 2022. 1, 2

[6] 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