LTP: Lane-based Trajectory Prediction for Autonomous Driving

Jingke Wang[1]*, Tengju Ye[1]*, Ziqing Gu[1, 2], Junbo Chen[1]†
[1]Alibaba Group [2]Tsinghua University
[1]{jingke.wjk, tengju.ytj}@alibaba-inc.com, junbo.chenjb@taobao.com
[2] {irene.gooqi}@gmail.com

Abstract

The reasonable trajectory prediction of surrounding traffic participants is crucial for autonomous driving. Especially, how to predict multiple plausible trajectories is still a challenging problem because of the multiple possibilities of the future. Proposal-based prediction methods address the multi-modality issues with a two-stage approach, commonly using intention classification followed by motion regression. This paper proposes a two-stage proposal-based motion forecasting method that exploits the sliced lane segments as fine-grained, shareable, and interpretable proposals. We use Graph neural network and Transformer to encode the shape and interaction information among the map sub-graphs and the agents sub-graphs. In addition, we propose a variance-based non-maximum suppression strategy to select representative trajectories that ensure the diversity of the final output. Experiments on the Argoverse dataset show that the proposed method outperforms state-of-the-art methods, and the lane segments-based proposals as well as the variance-based non-maximum suppression strategy both contribute to the performance improvement. Moreover, we demonstrate that the proposed method can achieve reliable performance with a lower collision rate and fewer off-road scenarios in the closed-loop simulation.

1. Introduction

Forecasting possible trajectories of surrounding agents is a crucial as well as challenging problem for autonomous driving, especially for multi-mode traffic scenes with uncertain future possibilities.

Early methods [11, 16] only learn a deterministic trajectory, resulting in output averaging in multi-mode scenes, also called the mode collapse issue. Probabilistic approaches [2, 4] are proposed to generate multi-modal outputs by modeling diverse trajectories with a probability distribution (e.g., GMM). However, the predicted trajectories of these methods are poor in interpretability and depend heavily on the predefined distribution.

To address the aforementioned multi-modality problem, proposal-based methods [8, 14, 24] decouple the trajectory prediction problem into classification-based intention prediction and regression-based motion prediction. Some works [8, 24] artificially sample target points, unique for each predictable agent, as proposals, which could not be reused for other predictable agents. Lapred [10] utilizes entire lanes as proposals, making it hard to model fine-grained intentions. Therefore, it is crucial to choose proposals that can (i) model the intention accurately, and (ii) be shared among all agents.

Furthermore, how to select representative trajectories is also a vital issue. The proposal-based methods generate a cluster of candidate trajectories based on the proposals. It is defective to output top-k trajectories merely ordered by the probability of the predicted proposals, since some intentions, though slightly lower in probability, are very critical for safety. Traditional methods utilize the Non-Maximum Suppression (NMS) algorithm, given a fixed threshold, to select the trajectories outside the threshold area into the output set greedily and successively. However, a fixed threshold cannot balance accuracy and variety. Therefore, some methods [7, 8] try to output multiple trajectories through optimization-based methods, causing complicated workflow and increased computation cost.

In this paper, we propose a novel Lane-based Trajectory Prediction (LTP) method for autonomous driving. We argue that there is no need for any complicated hand-craft proposals because the lane segments in the map are suitable proposals for intention modeling by nature. There are three advantages of using lane segments as proposals: **(i) Each lane segment explicitly represents a fine-grained intention.** Lane segments can represent tactical-level intentions, such as "change right for overtaking" or "change right for parking." While the method using the entire road lanes as proposals can only learn the road-level intention like "change left" or "change right." [10] **(ii) The interac-**

*These authors contributed equally to this work.
†Corresponding author.

tion between the lane segments and the agents can be well captured. When learning interaction with the map, the representation of the lane segments will naturally pass through the network. Thereby, it will generate a more informative embedding, unlike the TNT [24] which directly concatenates the unlearned point-level proposals to the agents embedding. **(iii) The lane segments-based proposals can be shared among agents.** The lane segments are independent of the agents, which means that once learned through the backbone, the embedding of the lane segments is fixed, no matter how many agents to predict. Just by concatenating the embedding of the agents and the lane segments, the different agents in the same scene can be predicted in parallel, unlike the DenseTNT [8] which needs to learn unique proposals embedding for different agents.

Based on these proposals, our method predicts which lane segment will be the future destination and regresses the corresponding trajectory. Furthermore, we found that the variance of the learned lane segments probability scores represents the uncertainty of the model to the current situation. Therefore, we propose a variance-based NMS method, which dynamically adjusts the NMS threshold according to the variance of the top-k prediction scores and achieves the balance of accuracy and variety.

The contributions of this paper are summarized as follows:

- A lane-based trajectory prediction method with explicit intention modeling is proposed, which uses lane segments as proposals to predict all agents in one shot.
- A novel variance-based NMS algorithm is proposed, which achieves the balance of accuracy and diversity by dynamically adjusting the NMS threshold according to the uncertainty of the prediction output.
- We outperform all published methods on the Argoverse forecasting benchmark. We also show that the proposed LTP could achieve reliable performance as a planner in closed-loop simulation.

2. Related Works

The trajectory prediction problem mainly contains three sub-tasks: representing the environment, learning interaction, and generating multi-modal outputs. We review some relevant works in this section.

State representation. Representing states of maps and agents is crucial for extracting effective features in the motion prediction task. A typical set of approaches renders states as multi-channel rasterized images [2, 4] with the bird-eye view. Convolutional neural network (CNN) [22] is a common selection, mainly focusing on spatial features. However, the rasterized representation leads to accuracy loss and suffers from capturing spatially distant interactions. Recently, the graph architecture has attracted much attention, such as VectorNet [5] and LaneGCN [13], utilizing GNNs to encode High Definition (HD) maps and generate vectorized representation. We draw inspiration from the two, while the differences lie in that we characterize the map through connected lane segments. At the same time, the lane segments can also be used as intention proposals and shared among agents.

Interaction modeling. The future trajectories are greatly influenced by the complex but subtle interactions among relevant agents and the environment. With the development of natural language processing, the attention mechanism [21] is introduced to capture long-term relationships. In the motion forecasting field, various attention-based methods have been employed to model relationships in different levels: spatial-level, temporal-level, and entity-level (e.g., road elements, agents trajectory). VectorNet [5] introduces a unified self-attention model to directly learn the interactions between all the sub-graphs in the environment, but it lacks more detailed modeling of interactions. Goal-oriented lane attention [15] is proposed to emphasize the relationship between agents and lanes, while the attention is applied at road level, which makes it difficult to model the fine-grained interactions. Further, LaneGCN [13] models four types of interactions between actors and lanes with a split-joint attention mechanism, which effectively captures the complex topology of lane graphs and long-range dependencies. Inspired by the VectorNet, we also model the interactions with the attention model. Unlike VectorNet, we utilize self-attention and cross-attention to process the information from different domains and introduce multi-layer attentions to fuse the learned embedding.

Multi-modal output. Prediction models should output multi-modal trajectories to adapt to environmental uncertainty [4, 15, 19]. Relevant works mainly adopt two categories: implicitly modeling the multi-modes as latent variables and explicitly generating multiple guiding proposals with the model or prior knowledge. Specifically, the former train Gaussian Mixture Models (GMMs) [2] or Mixture Density Networks (MDNs) [17] that generate a distribution over possible trajectories, then utilize VAEs, or GANs [2, 9, 12] to sample diverse future modes from latent variables. Drawbacks lie in the interpretability, making it difficult to guarantee the probability associated with each explicit intention that humans understand.

The proposal-based methods design diverse proposals to decouple the prediction problem into intention prediction and motion prediction. Region-level proposals are proposed by mmTransformer [14], but these proposals rely on manually predefined regions and the final performance is sensitive to the number of proposals. Some methods [8, 24] sample the proposal points around the centerline of the lane, so

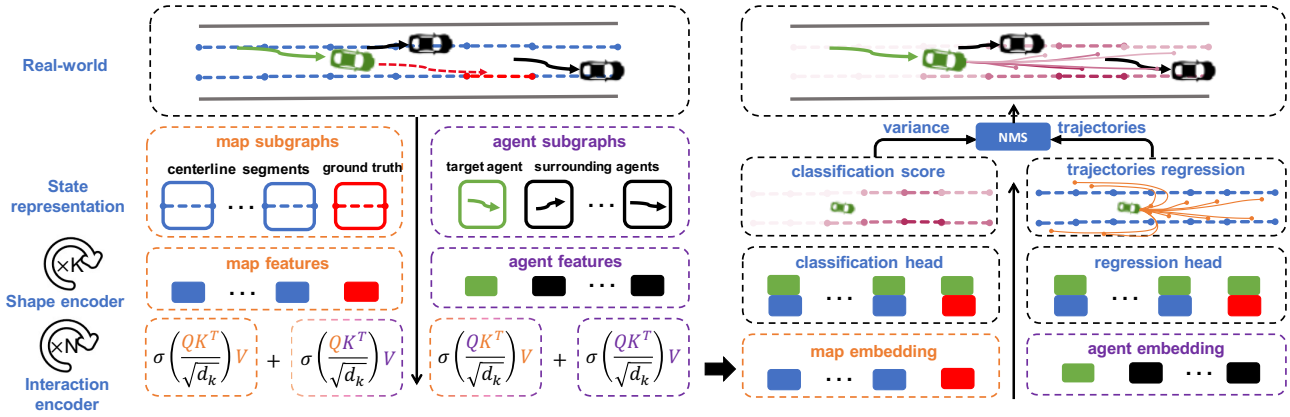

Figure 1. LTP architecture: (1) The vectorized lane segments and agents trajectories are firstly input to the shape encoder to extract shape features. (2) The self-attention and cross-attention layers are used to capture the long-term interaction between map information and agent information. (3) The target agent embedding is concatenated with each lane segment embedding in the map and input to the classification head and regression head. The classification head predicts which lane segment is more likely to be the destination and the regression head generates a trajectory for each lane segment. (4) A variance-based NMS algorithm is proposed to select representative trajectories from all generated trajectories.

that the detailed intentions can be well captured. However, their sampling relies on artificially designed rules. Different agents cannot share proposals, so the algorithm can only predict each agent serially. LaPred [10] proposes a lane-aware prediction method, which uses lanes as proposals to model vehicle intentions. But it directly inputs the entire lane, making it impossible to model fine-grained intent. Different from the above methods, we use the lane segments from the map as the proposals, which can clearly describe the agents' fine-grained intentions and be shared globally.

To select representative trajectories from all predicted trajectories, NMS is widely used in proposal-based methods. TNT [24] and mmTransformer [14] employ traditional NMS as a selector after model output. However, it is difficult to balance accuracy and multi-modal output with a fixed NMS threshold. Optimization-based approaches are introduced to select the end-points of the trajectories [7, 8]. DenseTNT [8] further designs a network to learn the result of optimization. But the prediction system is still complicated and the computation cost is increased. We propose a variance-based NMS algorithm that can dynamically adjust the NMS threshold according to the uncertainty of the model to the scene, which takes into account both accuracy and multi-modality.

3. Method

The architecture of our LTP is shown in Fig 1, and we introduce the details in this section.

3.1. Environment Representation

Inspired by VectorNet [5], we represent the lane segments and agents history trajectories as vectors connected end to end. The difference with VectorNet is that our map consists of sliced lane segments to model the intention precisely.

Specifically, given an environmental scenario $E : \{\mathcal{C}, \mathcal{A}\}$, where $\mathcal{C} = \{c_i\}$ represents the lane segments in the scene and $\mathcal{A} = \{a_i\}$ represents all agents' trajectories. The length of each lane centerline segments c_i is cut to 5m or less if the lane is shorter than 5m and annotated as $c_i = \left\{c_i^j, j \in [1, N]\right\}$. N controls the resolution of the map. Each c_i^j describes a section of the $\frac{5}{N}$-meters lane, defined as $[sx, sy, ex, ey, th, le, ty]$, where (sx, sy) and (ex, ey) are the start and end point of the vector, th and le represent the heading angle and the length of the vector, and ty represents the type of map elements, such as centerlines or sidelines. Similarly, for each agent trajectory $a_i = \left\{a_i^j, j \in [1, T]\right\}$, where T represents the total steps of the trajectory, each a_i^j describes a vector that makes up the trajectory, defined as $[sx, sy, ex, ey, th, le, ty, ts]$. The definitions of the first six dimensions are the same as c_i^j. ty represents the type of agents, such as vehicles or cyclists. Further, ts describes the absolute time relative to the current frame to overcome noise caused by data acquisition frequency jitter.

3.2. Embedding Extracting

Shape encoder. Since there are usually a large number of lanes and agents in a scene, it is necessary to aggregate the local information of each subgraph first. Here we follow VectorNet to encode the local shape feature, which draws on the idea of PointNet [20], encoding all the information of a lane segment or agent into a fixed-length embedding

through a fully connected graph neural network.

For each polyline $\mathcal{P}$ (i.e. c_i or a_i above) with its vector nodes $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_P\}$, each layer of the local shape encoder is defined as

$$\mathbf{v}_i^{(l+1)} = \varphi_{rel}\left(g_{enc}(\mathbf{v}_i^{(l)}), \varphi_{agg}\left(\left\{g_{enc}(\mathbf{v}_j^{(l)})\right\}\right)\right), \quad (1)$$

where $\mathbf{v}_i^{(l)}$ is the input of the l-th layer and $\mathbf{v}_j$ are the other vectors connected with $\mathbf{v}_i$ in $\mathcal{P}$. We treat each subgraph as fully-connected. φ_{rel} is feature concatenation operation, g_{enc} is a multi-layer perception (MLP) and φ_{agg} is the max-pooling operation. Finally, for the output embedding $\{\mathbf{v}_1^o, \mathbf{v}_2^o, ..., \mathbf{v}_P^o\}$, max-pooling is used to further generate unique embedding that integrates the shape information of the entire polyline.

Global interaction modeling. Through shape encoder, we generate an embedding for each lane segment and agent. After that, a multi-layer attention network is introduced to capture the global interaction between agents and lane segments. Given arbitrary feature matrices $\mathcal{O}$, $\mathcal{U}$ and theirs linear projections $\mathcal{O}_\mathcal{Q}$, $\mathcal{O}_\mathcal{K}$, $\mathcal{O}_\mathcal{V}$ and $\mathcal{U}_\mathcal{Q}$, $\mathcal{U}_\mathcal{K}$ and $\mathcal{U}_\mathcal{V}$, the $SelfAttn(\mathcal{O})$ and $CrossAttn(\mathcal{U}, \mathcal{O})$ are defined as

$$SelfAttn(\mathcal{O}) = \frac{Softmax(\mathcal{O}_\mathcal{Q}\mathcal{O}_\mathcal{K}^T)}{\sqrt{d_k}}\mathcal{O}_\mathcal{V}, \quad (2)$$

$$CrossAttn(\mathcal{U}, \mathcal{O}) = \frac{Softmax(\mathcal{O}_\mathcal{Q}\mathcal{U}_\mathcal{K}^T)}{\sqrt{d_k}}\mathcal{U}_\mathcal{V}, \quad (3)$$

where $\sqrt{d_k}$ is the dimension of the key vectors. We design a multi-layer cross-attention network to model the interaction between agents and lane segments. For each layer l, the agents embedding $A^{(l)}$ and lane segments embedding $C^{(l)}$ are fused by the $SelfAttn$ and $CrossAttn$ operation,

$$\mathcal{A}^{(l+1)} = SelfAttn(\mathcal{A}^{(l)}) + CrossAttn(\mathcal{C}^{(l)}, \mathcal{A}^{(l)}), \quad (4)$$

$$\mathcal{C}^{(l+1)} = SelfAttn(\mathcal{C}^{(l)}) + CrossAttn(\mathcal{A}^{(l)}, \mathcal{C}^{(l)}). \quad (5)$$

3.3. Multi-Modal Trajectory Prediction

The road lanes are designed to guide vehicles, so they play a pivotal role in predicting vehicles' motion. Once the lane segment is known, it is easy to regress trajectory based on the target lane segment. Therefore, a two-stage trajectory prediction framework is designed, to classify lane segments first and then predict directional trajectories based on different target lane segments.

We concatenate the embedding a_i of the target agent with the embedding c_j of all the lane segments in the map, merged as a tensor $e_i = \{[a_i, c_0], [a_i, c_1], ..., [a_i, c_M]\}$. each element in e_i represents a kind of intention of the target agent. Then we input e_i into classification head to predict the probability of each intention and regress motion trajectory based on the given intention. Since c_j is the independent proposal, it is only required to concatenate different agent embeddings with lane segments together so as to predict the future trajectories of multiple agents in parallel.

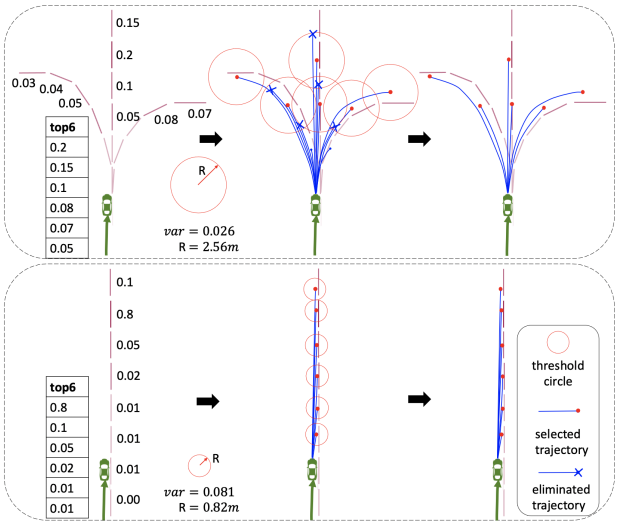

Figure 2. Schematic diagram of our variance-based NMS method. We dynamically adjust the NMS threshold R according to the uncertainty of the model prediction.

Lane segments classification. We use a three-layer MLP to score each lane segment and predict whether it is the future destination of the agent. Therefore, it becomes a binary classification problem for each lane segment. Before the start of training, we design a function f to score the lane segments so as to generate a reasonable ground truth. It is designed as

$$f = \alpha_1 \|o_i - l_i\| + \alpha_2 |\theta_{o_i} - \theta_{l_i}| + \alpha_3 \sin|\theta_{o_i} - \theta_{l_i}|. \quad (6)$$

The first item measures the Cartesian distance between the trajectory end-point o_i and the center point of the lane segment l_i. The second item measures the angle error between the agent heading θ_{o_i} and the direction of the lane segment θ_{l_i}, and the last item penalizes the case that the trajectory heading is perpendicular to the lane direction. Lane segments with scores less than D are marked as ground truth to provide dense supervision.

Trajectory regression. The regression head is also a three-layer MLP. For each element of e_i, one predicted trajectory will be generated. Each predicted trajectory is defined as $[x_0, y_0, x_1, y_1, ..., x_T, y_T]$, where T is the maximum time step of the prediction, with 0.1 second intervals.

Loss function. Our loss function consists of three items, namely classification loss $\mathcal{L}_{clf}$, regression loss $\mathcal{L}_{reg}$ and diversity loss $\mathcal{L}_{div}$. For the classification output of M lane

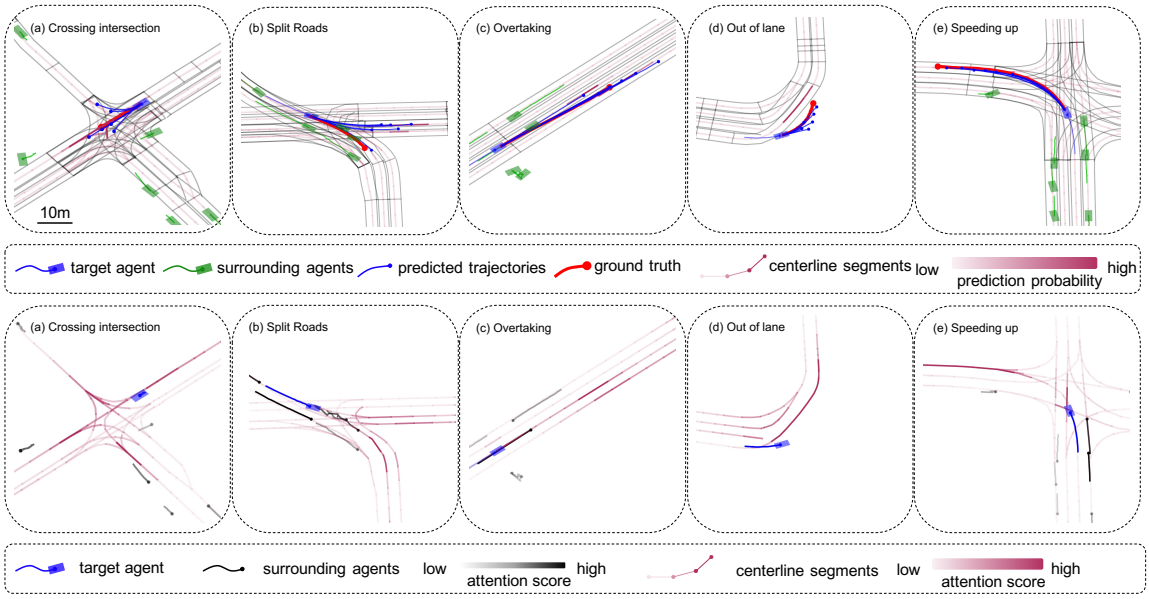

Figure 3. (Upper) The predicted trajectories in Argoverse validation set. The shade of purple on the lane segments represents the level of the classification score, which proves that LTP can give interpretable predictions in challenging scenarios. (Lower) Attention maps corresponding to the upper scenes. The depth of the color represents the degree of attention paid by the target agent to each lane segment and surrounding agent. We can see that the surrounding agents with higher interaction with the target agent have received more attention. The attention to the lane segments shows that the model has learned the correct road topology connection.

segments $p_0, p_1, ..., p_M$, the classification loss function is defined as

$$\mathcal{L}_{clf} = -\sum_{i=1}^{M}(y_i \log \sigma(p_i) + (1-y_i)\log \sigma(1-p_i)), \quad (7)$$

where y_i is 1 if the i-th lane segment is the ground truth, and 0 otherwise. σ is the $sigmoid$ function. For the regressed trajectories $\tau_1, \tau_2, ..., \tau_N$ generated from the ground truth lane segments, the regression loss function is defined as

$$\mathcal{L}_{reg} = \frac{1}{N}\sum_{i=1}^{N} SmoothL1Loss(\tau_i, \tau_{gt}), \quad (8)$$

where τ_{gt} is the ground truth trajectory, which ensures the closeness of the predicted trajectory to the provided lane segment. In addition, to generate diverse trajectories, we design a diversity loss to the trajectories $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_6\}$ which are generated from the top-6 highest-scoring lane segments,

$$\mathcal{L}_{div} = \min_{\tau \in \mathcal{T}} SmoothL1Loss(\tau, \tau_{gt}). \quad (9)$$

The final loss function is a weighted addition of the above three items,

$$Loss = \lambda_1 \mathcal{L}_{cls} + \lambda_2 \mathcal{L}_{reg} + \lambda_3 \mathcal{L}_{div}, \quad (10)$$

where λ_1, λ_2, and λ_3 are set to 0.5, 1.0 and 1.0 in practice.

3.4. Variance-based Non-Maximum Suppression

As shown in Fig 2, the probability value will be averaging when the model is unsure about its prediction. While when the agent's future trajectory is determined, the variance of the top-6 classification will be large. Based on this observation, we design an adaptive threshold function, written as

$$thr = \max(\min(\frac{k}{var(p)}, \gamma_1), \gamma_2), \quad (11)$$

where $var(p)$ is the variance of the top-6 probabilities, γ_1 and γ_2 are the upper and lower bounds of the threshold, and k is a constant coefficient. With the calculated threshold, we greedily select trajectory from the generated trajectory clusters in descending order of probability. When a trajectory is selected, other trajectories within thr from its endpoint will be discarded. Through this low-cost design, the density of the output trajectory will depend on the uncertainty of the model, so that both accuracy and diversity are taken into account.

4. Experiments

4.1. Experimental Settings

Dataset. We evaluate our method on Argoverse dataset and our internal dataset. Argoverse Motion Forecasting dataset provides a commonly used benchmark for trajectory prediction, consisting of 205,942 trajectories for the training set,

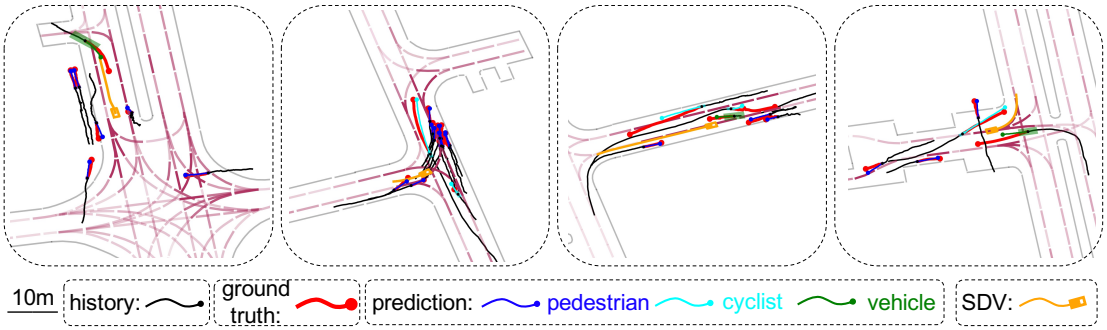

Figure 4. The prediction results on our internal dataset. The orange vehicle is a self-driving vehicle (SDV) for data acquisition. The black curve represents the history trajectory of the surrounding agents, and the red curve is its ground truth trajectory. The blue, cyan, and green curves are the predicted future trajectories of pedestrians, cyclists, and vehicles, respectively. For simplicity, we only show the top-1 trajectory. Note that with the unified lane segments as map representation, the prediction for all surrounding agents can work in parallel.

	b-$minFDE_6$	$minFDE_k$		$minADE_k$	
		k=1	k=6	k=1	k=6
Vnet [5]	\	4.01	\	1.81	\
Lapred (v) [10]	\	3.29	1.44	1.48	0.71
TNT [24]	2.14	4.95	1.45	2.17	0.91
LGCN [13]	2.05	3.78	1.36	1.70	0.87
mmTrans [14]	2.03	4.00	1.34	1.77	0.84
DTNT [8]	1.98	3.63	1.28	1.68	0.88
SceneTrans [18]	1.89	4.05	**1.23**	1.81	**0.80**
Gohome [6]	1.86	3.68	1.29	1.70	0.89
ours	**1.856**	**3.55**	1.29	**1.62**	0.83

Table 1. Comparisons with SOTA in Argoverse Leader board. The results of Lapred are from the validation set.

$39,472$ trajectories for the validation set, and $78,143$ trajectories for the test set. Each trajectory is divided into two parts, past 2 seconds as input and future 3 seconds as the label. The dataset was collected on public roads in Miami and Pittsburgh, and only vehicles need to be predicted.

We collect an internal dataset from the campus roads to further evaluate our method, consisting of $7,897,558$ trajectories, including $1,331,862$ vehicles, $1,962,140$ cyclists, and $4,603,556$ pedestrians. Our map format and trajectory duration are consistent with Argoverse dataset. The difference is that our internal dataset mainly focuses on campus roads and crowded traffic. In the experiments, the data and models of the two datasets are not mixed in any form.

Metrics. We follow the metrics that the Argoverse benchmark used in CVPR-2021 competition. For 2021 competition, the leaderboard is sorted by brier minimum Final Displacement Error (b-$minFDE_6$), which is defined as $(1.0-p)^2+minFDE_6$, where p corresponds to the probability of the best-forecasted trajectory and $minFDE_6$ is the $L2$ distance between the end-point of the best-forecasted trajectory of 6 candidate trajectories and the ground truth. We also report the metrics $minFDE_6$, $minADE_6$, FDE_1, ADE_1 in test set, corresponding to the final point displacement error and average displacement error of top-6 and top-1 trajectories.

4.2. Results

Argoverse dataset. We compare LTP with several state-of-the-art (SOTA) methods published recently on Argoverse test set, and report results in Table 1. LTP outperforms all methods published recently on the main official metrics b-$minFDE_6$. In addition, our model achieves comparable results on other metrics terms. Compared with the recent proposal-based SOTA methods [8, 10, 14, 24], our method achieves the lead in almost all indicators. Note that Lapred [10]'s results are from the Argoverse validation set, but our method still surpasses it on $minFDE_6$, demonstrating the superiority of our LTP.

Fig 3 shows the qualitative results generated by LTP on the Argoverse validation set. Results show that LTP can yield multiple reasonable trajectories in multi-mode scenes. The model can predict trajectories with different intentions in the slow-moving scenes, such as slowing down to wait or accelerating to overtake. In addition, LTP predicts accurate and diverse trajectories in challenging scenes, such as driving out of the lane and sudden speeding-up. The visualization of attention scores further shows that the model learns topological relationships of the map and interactions among agents. When following a slow vehicle, the model will pay close attention to the vehicle in front, the distant overtaking lane segments, and the front following lane segments, for example, the overtaking scene shown in Fig 3.

Internal dataset. Table 2 shows the results on our internal dataset, which demonstrate that our model can achieve accurate predictions for different types of agents. Fig 4 shows that LTP can generate reasonable trajectories for various kinds of agents around autonomous vehicles with a gen-

	Type	Top-1 (m)		Top-6 (m)	
		FDE	ADE	FDE	ADE
agent-centric	Ped	0.64	0.32	0.38	0.20
	Cyc	1.63	0.73	0.86	0.39
	Veh	2.19	0.99	1.15	0.52
ego-centric	Ped	0.66	0.33	0.40	0.21
	Cyc	1.68	0.75	0.89	0.41
	Veh	2.27	1.02	1.18	0.54

Table 2. The prediction results of LTP for pedestrians (Ped), cyclists (Cyc) and Vehicles (Veh) in agent-centric and ego-centric coordinate system on our internal dataset.

eral model. Note that the origin of the coordinate system in our internal dataset is the current position of our ego self-driving vehicle. The prediction process can work in parallel with a shared encoder backbone, which significantly improves the prediction efficiency. The experiments show that the prediction accuracy in the ego-centric coordinate system is comparable with the agent-centric coordinate system.

Model size and computational time. The model size and computational time comparisons are shown in Table 3. We select scenes with multiple predictable agents from the Argoverse validation set for experiments and compare the relationship between the inference time and the number of predictable agents. Considering that the time consumption of the GNN-based method is related to the complexity of the scene, we carry out the test in 100 scenes and take the average value as the result.

Experiments show that LaneGCN has the largest model and time consumption. The time consumption of VectorNet and TNT is smaller when agents number is 1 but larger than LTP (Para) when agents number is 8. LTP is at a moderate level in terms of model size, but it is better than other methods on FDE and $minFDE_6$ results. At the same time, with the ego-centric coordinate system, LTP (Para) can predict for surrounding agents in parallel, which makes it scale well as the number of agents to be predicted increases.

4.3. Ablation Study

To demonstrate how each module in LTP contributes to the final result, we conduct the ablation study as shown in Table 4. All ablation experiments are performed on the Argoverse validation set, and we mainly verify the role of lane segments length, the attention modules, and the NMS algorithm.

Impact of lane segments length. In order to verify the influence of lane segments length, we compare the result of splitting the lane into 10m, 5m, 2m, and no split. The experiment results show that not splitting lanes is worse than splitting. Splitting with 5m intervals performs best, which takes into account the fine-grained intention while keeps the number of proposals from being too numerous.

Method	Params	FDE_k (m)		Inference Time (ms)		
		k=1	k=6	n=1	n=4	n=8
LGCN	3.7M	2.96	1.08	43.0	179.1	356.2
Vnet	179K	3.30	-	**3.6**	15.0	29.4
TNT	269K	3.95	1.29	4.5	16.8	33.5
LTP (Seq)	1.1M	**2.81**	**1.07**	11.4	45.2	92.0
LTP (Para)	1.1M	2.91	1.11	11.4	**11.9**	**12.6**

Table 3. Model parameters and time consumption comparisons with SOTA methods. LTP (Seq) stands for LTP with agent-centric coordinate system and predicting each agent in sequence. LTP (Para) stands for LTP with ego-centric coordinate system and predicting all agents in the scene in parallel. n stands for the number of agents to be predicted in the same scene. The Vnet and TNT are implemented by ourselves. The time data is tested with a GeForce RTX 3080 graphics card with 10G memory.

Lane	Attn	NMS	Top-1 (m)		Top-6 (m)	
			FDE	ADE	FDE	ADE
entire	C-4	No	3.71	1.66	1.61	0.96
10m			3.03	1.38	1.30	0.87
5m			**2.81**	**1.29**	**1.23**	**0.85**
2m			2.91	1.34	1.25	0.86
5m	No	No	3.66	1.68	1.40	0.90
	C-1		3.07	1.40	1.34	0.86
	C-4		**2.81**	**1.29**	**1.23**	**0.85**
5m	C-4	Fix	2.81	1.29	1.14	0.80
		Adap	**2.81**	**1.29**	**1.07**	**0.78**

Table 4. Ablation on lane segments length, attention methods and NMS policy. C-N stands for Cross-attention with N layers.

Impact of attention modules. Experiments show that thanks to the concatenation of the lane segments embedding and the agent embedding, the model can learn well prediction results even without attention, especially in top-6 metrics. It can also be seen that deeper attention layers show greater potential, which demonstrates that the model can learn more implicit interactions between the agents and the map with a deeper network.

Impact of NMS policy. We compare the impact on top-6 results of the three methods of not using NMS, using a fixed radius for NMS, and using our adaptive NMS. For the latter two methods, we manually adjust the thresholds and report the best results. The results demonstrate that variance-based NMS selects more representative trajectories from the candidate trajectories cluster and significantly improves the $minFDE_6$ result.

4.4. Closed-loop Simulation

Vanilla Closed-loop inference. The trajectory prediction problem is similar to the behavior cloning in imitation learn-

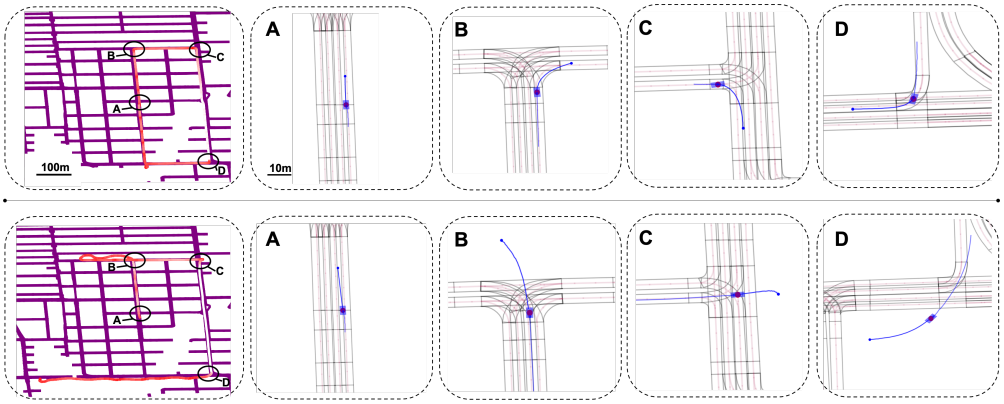

Figure 5. Comparisons of LTP (upper) and VectorNet (lower) in closed-loop experiment. The leftmost figure is the test map, which is a part of the Miami map. The red path is the path traversed by two methods from a global perspective. Brighter color represents higher speed. The right figures show scenes where LTP can generate high-quality trajectories while VectorNet keeps entering the non-drivable area.

ing. A natural idea is whether the prediction model can be used for trajectory planning. However, the distribution drift [3] problem in imitation learning will cause the predicted trajectory to deviate from the lane self-reinforcingly. Therefore, to complete a closed-loop map roaming, it is necessary to have a deep understanding of the map instead of simply following past trajectory trends.

To test the closed-loop planning ability of LTP, we give a 2s initial trajectory from a random start point on the Miami map of Argoverse and let the LTP continuously predict the next 3s utilizing its own predicted trajectory as input. Fig 5 shows the comparisons of planning results from our method and VectorNet in the future 300 seconds. In the experiments, LTP can complete the closed-loop wandering in the map with high quality, while the trajectories from VectorNet keep entering the non-drivable area.

Cost volume optimization. We employ a cost volume optimization method to further reduce the collision rate in dense traffic. From the experiments in vanilla closed-loop inference, we find that LTP can complete safe roaming in empty maps but has a weak awareness of avoiding obstacles. Inspired by the previous works [1, 23], we accumulate the multi-model predictions of the other vehicles into a cost volume. We also predict multi-model trajectories for the ego vehicle as candidates and search for the best ego-vehicle trajectory with the lowest cost in the cost volume.

We test the model in 1000 pieces of challenging driving logs logged from our internal dataset and compare the collision rate for single frame and entire trip. The entire trip consists of about 40 frames, and only the trip with all frames that do not collide is a safe trip. The comparison results are shown in Table 5 and the qualitative results are shown in Fig 6. We can see that the cost volume generated from our multi-model prediction greatly improves the obstacle avoidance ability, which shows that LTP has great potential to be applied to the simulation and closed-loop navigation.

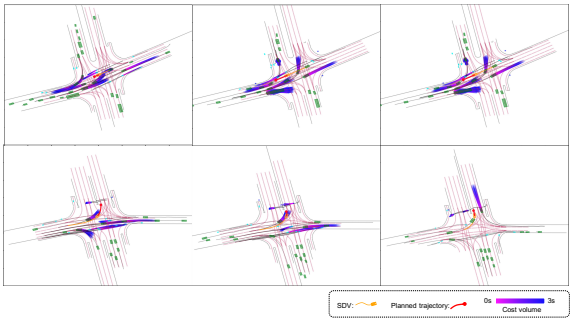

Figure 6. Closed-loop simulation with cost volume in dense traffic.

	Frame Collision Rate (%)	Trip Collision Rate (%)
Without cost volume	5.2	65.0
With cost volume	**0.7**	**17.2**

Table 5. Frame collision rate and trip collision rate of LTP.

5. Conclusions

We propose a lane-based trajectory prediction method called LTP for multi-modal prediction. The proposed method utilizes sliced lane segments to achieve accurate, interpretable, and efficient trajectory prediction. We demonstrate the effectiveness of the proposed LTP with experiments on the Argoverse motion benchmark and our internal dataset. In addition, we demonstrate the closed-loop planning capability of the proposed LTP, with reliable performance in closed-loop simulation experiments.

6. Acknowledgement

This work was supported by Alibaba Group through Alibaba Research Intern Program (AIR).

References

[1] Sergio Casas, Abbas Sadat, and Raquel Urtasun. Mp3: A unified model to map, perceive, predict and plan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14403–14412, 2021. 8

[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, 2

[3] F. Codevilla, E. Santana, AM López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. *IEEE*, 2019. 8

[4] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 2090–2096. IEEE, 2019. 1, 2

[5] 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. 2, 3, 6

[6] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output forfuture motion estimation. *arXiv e-prints*, pages arXiv–2109, 2021. 6

[7] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. *arXiv preprint arXiv:2105.10968*, 2021. 1, 3

[8] Junru Gu, Chen Sun, and Hang Zhao. Densetnt: End-to-end trajectory prediction from dense goal sets. *arXiv preprint arXiv:2108.09640*, 2021. 1, 2, 3, 6

[9] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 2

[10] ByeoungDo Kim, Seong Hyeon Park, Seokhwan Lee, Elbek Khoshimjonov, Dongsuk Kum, Junsoo Kim, Jeong Soo Kim, and Jun Won Choi. Lapred: Lane-aware prediction of multi-modal future trajectories of dynamic agents. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14636–14645, 2021. 1, 3, 6

[11] N. Lee, W. Choi, P. Vernaza, C. B. Choy, Phs Torr, and M. Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[12] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 2

[13] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. In *European Conference on Computer Vision*, pages 541–556. Springer, 2020. 2, 6

[14] Yicheng Liu, Jinghuai Zhang, Liangji Fang, Qinhong Jiang, and Bolei Zhou. Multimodal motion prediction with stacked transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7577–7586, 2021. 1, 2, 3, 6

[15] Chenxu Luo, Lin Sun, Dariush Dabiri, and Alan Yuille. Probabilistic multi-modal trajectory prediction with lane attention for autonomous vehicles. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2370–2376. IEEE, 2020. 2

[16] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[17] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7144–7153, 2019. 2

[18] Jiquan Ngiam, Benjamin Caine, Vijay Vasudevan, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley, Chenxi Liu, Ashish Venugopal, et al. Scene transformer: A unified multi-task model for behavior prediction and planning. *arXiv preprint arXiv:2106.08417*, 2021. 6

[19] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020. 2

[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 3

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2

[22] Takuma Yagi, Karttikeya Mangalam, Ryo Yonetani, and Yoichi Sato. Future person localization in first-person videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7593–7602, 2018. 2

[23] 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. 8

[24] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020. 1, 2, 3, 6