

Aware of the History: Trajectory Forecasting with the Local Behavior Data

Yiqi Zhong¹[0000-0002-0928-8018], Zhenyang Ni²[0000-0001-7134-620X], Siheng Chen^{2,*}[0000-0001-6199-529X], and Ulrich Neumann^{1,*}[0000-0001-8977-7112]

¹ Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA {yiqizhon, uneumann}@usc.edu

² Cooperative Medianet Innovation Center (CMIC), Shanghai Jiao Tong University, Shanghai, China {0107nzy, sihengc}@sjtu.edu.cn

Abstract. The historical trajectories previously passing through a location may help infer the future trajectory of an agent currently at this location. Despite great improvements in trajectory forecasting with the guidance of high-definition maps, only a few works have explored such local historical information. In this work, we re-introduce this information as a new type of input data for trajectory forecasting systems: the *local behavior data*, which we conceptualize as a collection of location-specific historical trajectories. *Local behavior data* helps the systems emphasize the prediction locality and better understand the impact of static map objects on moving agents. We propose a novel local-behavior-aware (LBA) prediction framework that improves forecasting accuracy by fusing information from observed trajectories, HD maps, and local behavior data. Also, where such historical data is insufficient or unavailable, we employ a local-behavior-free (LBF) prediction framework, which adopts a knowledge-distillation-based architecture to infer the impact of missing data. Extensive experiments demonstrate that upgrading existing methods with these two frameworks significantly improves their performances. Especially, the LBA framework boosts the SOTA methods' performance on the nuScenes dataset by at least 14% for the $K = 1$ metrics.³

Keywords: Trajectory forecasting, Historical data, Knowledge-distillation

1 Introduction

Trajectory forecasting aims to predict an agent's future trajectory based on its past trajectory and surrounding scene information. This task is essential in a variety of applications, including self-driving cars [27], surveillance systems [8], robotics [2], and human behavior analysis [36]. Prior prediction systems primarily use deep-learning-based models (e.g., LSTM, temporal convolutions) to exploit limited information such as past trajectories [1,21,30]. Recent efforts

* Corresponding authors

³ Code is at <https://github.com/Kay1794/LocalBehavior-based-trajectory-prediction>

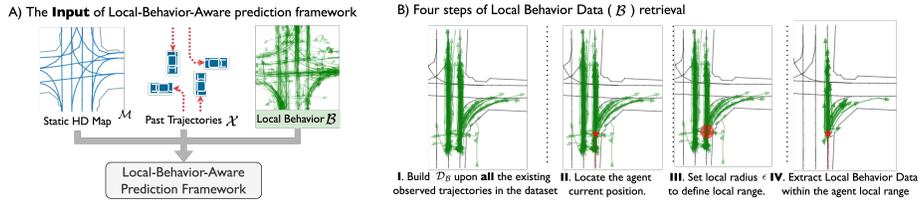


Fig. 1: A) Compared to previous works that mostly rely on HD maps and agents’ past trajectories, we additionally input the local behavior data to the prediction framework. B) For agents to be predicted in the scene, we follow these steps to retrieve their local behavior data for the framework input composition.

also reveal that forecasting ability will improve as more scene information is introduced into the input. One type of scene information, for example, is the past trajectories of a target agent’s neighboring agents. To date, many graph-neural-network-based methods have explored the potential of agents’ motion features and interactive motion relations to improve predictions [20,35,42]. Recently, high-definition (HD) maps are incorporated as an additional type of scene information [9,11,13,23,41] to provide geometric priors.

Besides the widely used HD maps and agents’ past trajectories, we propose a novel Local-Behavior-Aware (LBA) prediction framework that takes a new type of scene information as the input, which we term as *local behavior data*. The local behavior data is defined as a collection of historical trajectories at an agent’s current location. Fig. 1 (A) shows the three components of the LBA framework input. Taking local behavior data as the input brings two benefits to the task. First, the data provides location-specific motion patterns, which helps the model effectively narrow down the search space for future trajectories. Most of the existing prediction models solely rely on the features learned from the static HD map to infer such information [15,10,13]. In comparison, taking local behavior data as the input immediately equips the model with this information, making the model more tractable and robust. Second, local behavior data provides complementary information to augment static maps into dynamic behavioral maps. The map prior in the current literature is limited to static geometric information. The rich dynamic information brought by this new input would help the model better understand the impact of map objects on moving agents.

Many car companies and navigation apps are collecting such local behavior data. Yet, sometimes this data is insufficient or is yet to be gathered (e.g. when a self-driving car explores new areas). Therefore, we further propose a Local-Behavior-Free (LBF) prediction framework that only takes the current agents’ observed trajectories and HD maps as the input, when the local behavior data is unavailable. Inspired by recent development in knowledge distillation, we use our pre-trained LBA prediction framework as the teacher network during the training phase. This teacher network guides the LBF student network in inferring the features of the absent local behavior data. The intuition behind this design is

that a traffic agent’s movement at a particular location is confined to a limited number of possibilities. The teacher network essentially provides the ground truth of the movement pattern, making it plausible for the student network to learn the inference of the pattern given the current scene information.

LBA and LBF frameworks both have strong generalizability to be adopted by a wide range of existing trajectory forecasting methods. In Sec. 4.2 and Sec. 5.2, we showcase the implementation methodology on how to upgrade existing systems to the LBA/LBF framework respectively. We then implement and validate the two frameworks based on three state-of-the-art trajectory forecasting methods, P2T [9], LaneGCN [23], and DenseTNT [15].

In summary, this work has three major contributions to the literature:

- We propose a Local-Behavior-Aware prediction framework. It enables most of the existing methods to incorporate the local behavior data, a new type of system input which contains the local historical information.
- We further introduce a Local-Behavior-Free prediction framework, which adopts a knowledge-distillation-based architecture to infer local behavioral information when it is temporarily unavailable.
- We conduct extensive experiments on published benchmarks (Argoverse [6], nuScenes [3]), and validate that upgrading the SOTA methods to LBA/LBF frameworks consistently improves their performances by significant margins on various metrics. Especially, the LBA framework improves the SOTA methods on the nuScenes dataset by at least 14% on the K=1 metrics.

2 Related Work

2.1 Historical Behaviors in Trajectory Forecasting

Historical behaviors are very helpful to trajectory forecasting since they reveal the motion pattern of agents. Several previous works have made significant progress in this direction by adopting memory-based designs. MANTRA [28] uses the observed past trajectories and the map as the key to query the possible hidden features of future trajectories. Similarly, Zhao et al. [45] build an expert repository to retrieve the possible destinations by matching the current trajectory with the ones in the repository. MemoNet [40] also considers memorizing the destinations of each agent and first applies the memory mechanism to multi-agent trajectory forecasting. Compared to those memory-based methods, our work: i) regards the historical behaviors as system inputs to benefit the task from the perspective of enriching scene information; ii) directly uses geometric coordinates to query related historical information which emphasizes the data locality and is more interpretable and robust.

2.2 Scene Representation in Trajectory Forecasting

To use a new type of scene information in systems requires fusing it with existing scene information sources. After reviewing how scene information is encoded in

previous methods, we see two main types of scene representations: 1) rasterized-image-based representations [5,20], 2) graph-based representations [11,12,23,39]. Rasterized-image-based representations render static HD maps and motion data into a birds’ eye view (BEV) image, using various colors to represent different objects. Systems with this scene representation tend to use standard convolutional neural networks (CNNs) backbones to extract scene features [5,9,13,20,31]. These methods transform scenes into image coordinates systems.

Graph-based scene representations become popular with the recent development of graph learning [34] and attention mechanism [37]. These methods build a graph that can be either directional or non-directional, and use techniques such as graph neural network (GNN) [34] or attention-based operations [37] to enable the interaction among map objects and agents. The nodes are the semantic map objects, e.g., lane segments and agents. The edges are defined by heuristics, which can be the spatial distances between the two nodes or the semantic labels of the two nodes. Systems using graph-based scene representations [11,12,15,23,44] independently encode each map object to make the graph more informative. Graph-based scene representations have an outstanding information fusion capability and are substantially explored recently.

Other methods that do not strictly fall into the above two categories can share some properties with one or both of them. For example, TPCN [41] uses point cloud representations for the scene and does not manually specify the interaction structures. Yet, by using PointNet++ [32] to aggregate each point’s neighborhood information, TPCN still technically defines local complete sub-graphs for the scene where each point is a node connected with its neighbors.

To show the generalizability of the proposed frameworks, in this work we introduce an implementation methodology for upgrading forecasting systems that use either rasterized-image-based or graph-based scene representations to the LBA and LBF frameworks in Sec 4.2 and 5.2 respectively.

2.3 Knowledge Distillation

Our LBF framework is inspired by the recent study of knowledge distillation (KD). Knowledge distillation is a technique that compresses a larger teacher network to a smaller student network by urging the student to mimic the teacher at the intermediate feature or output level [19]. KD is widely used for various tasks, including object detection [7,16], semantic segmentation [18,25] and tracking [26]. Its usage is still being explored. For example, researchers start to use KD for collaborative perception tasks [22]. In our work, the proposed LBF framework uses KD in the trajectory forecasting task. Compared to previous works that pay more attention to model compression, we seek to compress the volume of input data. We use the offline framework as the teacher network, which takes each agent’s local behavior data as the input along with HD maps and the agents’ observed trajectories, while the student network (i.e., the online framework) only uses the later two data modalities as the input without requiring local behavior data. Our experiments show that KD-based information compression significantly boosts the performance of trajectory forecasting.

3 Formulation of Local Behavior

In trajectory forecasting, the observed trajectories of the agents previously passing through a location may help infer the future trajectory of an agent currently at the location. In this work, we collect such historical information and reformulate it to make it become one of the inputs for the trajectory forecasting system. We name this new type of scene information as the *local behavior data*. In this section, we introduce its formulation and the methodology about how to retrieve such data from existing datasets.

Consider a trajectory forecasting dataset with S scenes and the p th scene has K_p agents. The observed trajectory and ground truth future trajectory of agent i in scene p are denoted respectively as $\mathbf{X}^{(i,p)}$ and $\mathbf{Y}^{(i,p)}$, where $\mathbf{X}^{(i,p)} \in \mathbb{R}^{\mathbf{T}^- \times 2}$ and $\mathbf{Y}^{(i,p)} \in \mathbb{R}^{\mathbf{T}^+ \times 2}$. Each $\mathbf{X}^{(i,p)}$ or $\mathbf{Y}^{(i,p)}$ consists of two-dimensional coordinates at \mathbf{T}^- or \mathbf{T}^+ timestamps. Note that the coordinate system used for the trajectories is the global coordinate system aligned with the global geometric map. In this work, we specifically denote two special items in $\mathbf{X}^{(i,p)}$. We use $\mathbf{X}_1^{(i,p)} \in \mathbb{R}^2$ to represent the location of agent i in scene s at the *first* timestamp, i.e., the first observed location of the agent. Accordingly, we use $\mathbf{X}_{\mathbf{T}^-}^{(i,p)} \in \mathbb{R}^2$ to denote the agent location at timestamp \mathbf{T}^- , i.e., its current location. By gathering all the observed trajectories in this dataset, we build a behavior database $\mathcal{D}_B = \{\mathbf{X}^{(i,p)}, p \in \{1, 2, \dots, S\}, i \in \{1, 2, \dots, K_p\}\}_{i,p}$. We can query the local behavior from \mathcal{D}_B ; namely, the local behavior data of agent i in scene p is

$$\mathcal{B}_\epsilon^{(i,p)} = \{\mathbf{X}^{(j,q)} \mid \|\mathbf{X}_{\mathbf{T}^-}^{(i,p)} - \mathbf{X}_1^{(j,q)}\|_2 < \epsilon, \mathbf{X}^{(j,q)} \in \mathcal{D}_B\}, \quad (1)$$

where ϵ is an adjustable hyper-parameter defining the radius of the neighboring area of a location. The *size* of $\mathcal{B}_\epsilon^{(i,p)}$ refers to the number of observed trajectories in $\mathcal{B}_\epsilon^{(i,p)}$. Fig. 1 (B) shows the steps to query local behavior data from \mathcal{D}_B .

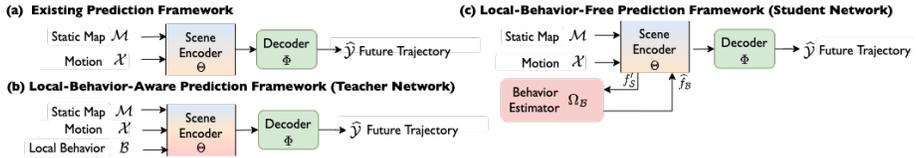


Fig. 2: General pipelines of: (a) the baseline model that uses HD map data and observed trajectory data, which is typical for SOTA trajectory forecasting systems; (b) the LBA framework that takes local behavior data as an extra input; (c) the LBF framework that uses estimated local behavior features during the inference phase. (c) has fewer constraints in the use cases compared to (b).

4 Local-Behavior-Aware Prediction

In real life, local behavior data has been widely collected by navigation apps and car companies. To use such data to benefit the trajectory forecasting task, we propose a Local-Behavior-Aware (LBA) prediction framework. In this section, we first demonstrate the generic pipeline of the LBA prediction framework and its major components. Then, we introduce the implementation strategy of the framework which especially emphasizes the representation of local behavior data and its corresponding scene encoder design.

4.1 Framework Pipeline

Like a typical prediction framework pipeline adopted by previous works [11,20,31] (see Figure 2 (a)), our LBA framework follows an encoder-decoder structure (see Figure 2 (b)). The encoder Θ extracts features from multiple scene information sources. Then, the subsequent decoder Φ generates the predicted future trajectories $\hat{\mathcal{Y}}$ based on the scene features.

The main difference between LBA and previous frameworks is the design of the encoder Θ . Our scene encoder involves two steps: 1) scene modeling, which assembles data from all three sources to generate a comprehensive representation of the current scene, and 2) scene feature extraction, which extracts the high-dimensional features from the generated scene representation. The way to represent a scene would largely determine the inner structure of a scene encoder.

In the current literature, the two most frequently used scene representations are **graph**-based and **rasterized-image**-based representations. To show that most of the existing forecasting systems can be upgraded to fit the proposed LBA prediction framework, we will describe the implementation strategies for the graph-based and rasterized-image-based systems.

4.2 Implementation

Graph-based Systems. The existing graph-based systems represent the whole scene information into a scene graph $G(V, E)$, where V is the node set that contains both the map object node set $V_{\mathcal{M}}$ and the agent node sets $V_{\mathcal{A}}$, and E is the edge set that reflects the internal interactions between nodes. For each map object node, its associated node attributes include the geometric coordinates, reflecting the static physical location of the map object. For each agent node, its associated node attributes include the two-dimensional coordinates across various time-stamps, reflecting the movement information of the agent. The characteristics of the graph-based representation are that i) it is compact and effective, leading to an efficient system; ii) it enables effective modeling of the interactions among objects (both map objects and agents) in the scene, which is crucial for understanding complicated dynamic relations in traffic scenarios.

To implement the graph-based LBA prediction system, we emphasize the strategy of incorporating local behavior data into the scene graph and enabling the feature interaction between local behavior data and other nodes.

Representation of Local Behavior Data. Given the i th agent in the p th scene, we can query its specific local behavior data $\mathcal{B}_\epsilon^{(i,p)}$ from the behavior database $\mathcal{D}_\mathcal{B}$. For each individual observed trajectory in $\mathcal{B}_\epsilon^{(i,p)}$, we create a local behavior node. This step results in a local behavior node set $V_\mathcal{B}$, which has the same size as the local behavior data.

Scene Encoder. As demonstrated in Fig. 3, the scene encoder of the LBA graph-based systems includes the scene graph initialization, individual node feature extraction, and interactive node feature extraction. To initialize the local-behavior-aware scene graph $G'(V', E)$, we add the local behavior node set to the original graph $G(V, E)$, where the updated node set is $V' = V \cup V_\mathcal{B}$ and the updated edge set is $E' = E \cup \{(v_m, v_n) | v_m \in V', v_n \in V_\mathcal{B}\}_{m,n}$. With this graph, the local behavior data will participate in the feature interaction procedure (some methods may further update the edge set based on the node distance [22,43,15]). The output of the scene encoder f_S will also be local-behavior-aware. We use three feature encoders ($\Theta_\mathcal{M}$, $\Theta_\mathcal{X}$, $\Theta_\mathcal{B}$) to extract the features of map objects $f_\mathcal{M}$, agents' observed trajectories $f_\mathcal{X}$, and the local behavior data $f_\mathcal{B}$, respectively. Here each scene node will obtain its corresponding node features individually. To capture internal interactions, we use an interaction module \mathcal{I} , which is either GNN-based or attention-based, depending on the design of the original system. The interaction module aggregates information from all three scene components.

The architecture of $\Theta_\mathcal{M}$, $\Theta_\mathcal{X}$ and \mathcal{I} can remain unchanged from the original system structure for our implementation. As for $\Theta_\mathcal{B}$, since each independent behavior data is essentially an observed trajectory, we can directly adopt the structure of the trajectory encoder $\Theta_\mathcal{X}$ in the system as our behavior encoder structure. We can also use simple encoder structures, such as multi-layer perceptrons (MLPs), to keep the system light-weight.

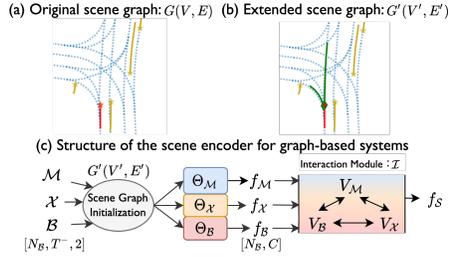


Fig. 3: Scene encoder implementation for graph-based systems. In (c): i) $V_\mathcal{B}, V_\mathcal{X}, V_\mathcal{M} \subseteq V'$ are respectively the behavior node set, motion node set, and map node set; ii) $N_\mathcal{B}$ is the size of the local behavior data; iii) C is the feature channel; iv) each trajectory in the local behavior data is encoded independently.

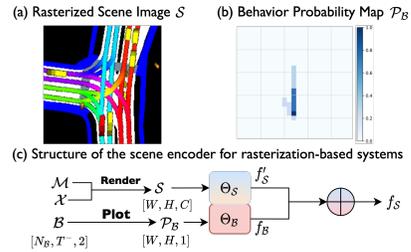


Fig. 4: Scene encoder implementation for rasterized-image-based systems. (a) is generated with P2T's [9] official code. In (c): features of the rasterized scene image and the local behavior are concatenated channel-wise to generate the final output features.

Rasterized-image-based Systems. Rasterized-image-based systems represent the whole scene information as a rasterized scene image (see an example in Fig. 4). The rasterized scene image reflects the HD map objects (junctions, lanes) and the agent trajectory information as BEV images with various colors according to their semantic labels. This representation method essentially transforms a global coordinate system into an image coordinate system, where each location on the map can be represented by a pixel coordinate. The main characteristic of rasterization-based representations is that they can leverage established CNN backbones, such as ResNet [17], to extract image features.

To implement the rasterized-image-based LBA prediction system, we need to render local behavior data to an image that has the same coordinate system with the original scene image, which ensures consistency and compatibility.

Representation of Local Behavior Data. We seek to render local behavior data into a *behavior probability map* $\mathcal{P}_{\mathcal{B}}$. It is an image whose pixel value reflects an agent’s moving probability from the current pixel to another pixel in the image. For the i th agent at the p th scene, $\mathcal{P}_{\mathcal{B}}^{(i,p)}$ is formulated as a single channel image of size $(H, W, 1)$ that shares the same coordinate system with the rasterized scene image, where W, H are the width and the height of $\mathcal{P}_{\mathcal{B}}$. To generate such a behavior probability map, we initialize each pixel value in the image $\mathcal{P}_{\mathcal{B}}^{(i,p)}(x, y)$ to be 0. We then enumerate each trajectory $f_{\mathcal{X}} \in \mathcal{B}_{\epsilon}^{(i,p)}$ and add the corresponding information into $\mathcal{P}_{\mathcal{B}}^{(i,p)}(x, y)$ by adding 1 to the pixel value once each trajectory covers that pixel. In the end, we normalize $\mathcal{P}_{\mathcal{B}}^{(i,p)}$ by dividing every pixel by the maximum value of the pixels, specifically

$$\mathcal{P}_{\mathcal{B}}^{(i,p)}(x, y) = \frac{\mathcal{P}_{\mathcal{B}}^{(i,p)}(x, y)}{\max(\mathcal{P}_{\mathcal{B}}^{(i,p)}(x, y))}, \quad (2)$$

where $0 \leq x < W, 0 \leq y < H$. Fig. 4 (b) illustrates the local behavior data of the agent represented as a red rectangle in Fig. 4 (a). Since the configuration of the rasterized scene image and the behavior probability map are the same, this probability map indicates how likely, according to local behavior data, the agent at the current pixel will pass a certain pixel on the scene image.

Scene Encoder. As shown in Fig. 4, the scene encoder of the rasterized image-based LBA systems includes image rendering, image feature extraction, and concatenation-based aggregation. To achieve scene modeling, we render both the rasterized scene image and the behavior probability map. To comprehensively extract scene features, we use two separate encoders, $\Theta_{\mathcal{S}}$, which extracts features from the rasterized scene image \mathcal{S} , and $\Theta_{\mathcal{B}}$, which extracts features from the behavior probability map $\mathcal{P}_{\mathcal{B}}$. The architecture of $\Theta_{\mathcal{S}}$ can remain identical with the one in the original systems. To implement $\Theta_{\mathcal{B}}$, we can adopt any established images’ feature extraction networks, such as ResNet [17]. Afterwards, the output of the two encoders will be concatenated channel-wise to build the output of the scene encoder, which is now local-behavior-aware.

5 Local-Behavior-Free Prediction

As mentioned in Sec 1, there will be scenarios where the local behavior data is yet to be gathered or insufficient. To handle such situations, we propose a Local-Behavior-Free (LBF) prediction framework based on knowledge distillation.

The training of the LBF prediction framework follows a teacher-student structure. The teacher network is implemented using the LBA framework introduced in Sec 4, which uses local behavior data as the third input; while the student network only takes static HD map information and the agents’ observed trajectories as the input. The knowledge-distillation-based training strategy enhances the training of the LBF prediction framework by urging the student network to imitate the teacher network in the intermediate feature levels when processing the same data samples.

The intuition behind this design is that given a specific location, the number of possible movement patterns of an traffic agent is limited. With the guidance from the teacher network that is trained on local behavior data, it is feasible for the student network to learn the reasoning of the movement pattern based on the static map objects and agents’ observed trajectories.

5.1 Framework Pipeline

The LBF framework includes a teacher network and a student network. See Fig. 2 (b), (c). The teacher network follows the LBA framework. For the student network, we remove the input stream of local behavior data in the LBA framework and add a behavior estimator $\Omega_{\mathcal{B}}$ to the pipeline. $\Omega_{\mathcal{B}}$ takes the intermediate scene features $f'_{\mathcal{S}}$ from the scene encoder Θ as the input, and outputs the estimated local behavior features $\hat{f}_{\mathcal{B}}$. Note that $f'_{\mathcal{S}}$ is not involved with local behavioral information. Next, we let $\hat{f}_{\mathcal{B}}$ join the scene encoder along with $f'_{\mathcal{S}}$ for the final feature generation. The scene encoder outputs the updated $f_{\mathcal{S}}$, which contains the estimated local behavioral information, to the decoder Φ . The decoder then processes $f_{\mathcal{S}}$ and generates the predicted future trajectories $\hat{\mathcal{Y}}$. The core step of the LBF student network is to link the behavior estimator and the scene encoder.

5.2 Implementation

Like Sec 4.2, this section introduces the implementation of the student networks in the LBF prediction framework based on two scene representations.

Graph-based Systems. In the teacher network (the LBA prediction system), the local behavior data goes through an encoder to obtain behavior features $f_{\mathcal{B}}$. In the student network (the LBF version), we use a behavior estimator $\Omega_{\mathcal{B}}$ to estimate the behavior features even when the original local behavior data is not available. We implement the behavior estimator $\Omega_{\mathcal{B}}$ by a graph neural network. Its input includes the features of the map objects $f_{\mathcal{M}}$ and the features of the agents’ observed trajectories $f_{\mathcal{X}}$, which include all scene information in hand. It

outputs the estimated behavior features \hat{f}_B . After f_M and f_X are interacted in the interaction module, we aggregate its output f'_S and the estimated behavior features \hat{f}_B in a fusion module Ψ to form the final scene feature f_S . We implement Ψ by an attention-based network. The pipeline of this procedure is shown in Fig. 5. During training, the estimated behavior features \hat{f}_B in the student network can be supervised by the behavior features f_B in the teacher network through a knowledge-distillation loss.

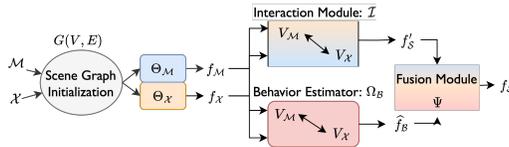


Fig. 5: Local-Behavior-Free implementation for graph-based systems. During training, \hat{f}_B and f_S can be supervised by the corresponding features from the teacher network.

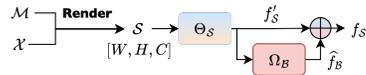


Fig. 6: Local-Behavior-Free implementation for rasterization-based systems.

Rasterized-image-based Systems. In the teacher network (the LBA prediction system), the behavior probability map leads to behavior features f_B . In the student network (the LBF version), we use a behavior estimator Ω_B to estimate such behavior features even the behavior probability map is not available. We implement the behavior estimator Ω_B by a CNN-based network. Its input is the output of the scene encoder Θ_S , which is the intermediate scene feature f'_S . Its output is the estimated behavior features \hat{f}_B . We then concatenate \hat{f}_B and f'_S to form the final scene feature f_S . The pipeline of this procedure is shown in Fig. 6. During training, the estimated behavior features \hat{f}_B in the student network can be supervised by the behavior features f_B in the teacher network through a knowledge-distillation loss.

Note that both graph-based and rasterized-image-based LBF systems follow the same design rationale: estimating local behavior data based on the other known scene information. The difference between the two types of systems is in the fusion step. In the graph-based system, we use a trainable fusion module; and in the rasterized-image-based system, since the scene image and the behavior probability map share the same coordinate system, we simply concatenate them.

5.3 Training with Knowledge Distillation

To train such a teacher-student framework, we pre-train an LBA prediction network as the teacher network, and then use the intermediate features from the teacher network as the supervision signals to guide the training of the student

network. We consider features from the teacher network that are leveraged to guide the student network training as \mathbf{F}_t and the corresponding features from the student network as \mathbf{F}_s . Note that \mathbf{F}_s may include but are not limited to the reconstructed local behavior features and the the scene encoder’s outputs .

The training loss thereafter contains the trajectory forecasting loss \mathcal{L}_{pred} , which is defined by the original system and identical with the training loss of the teacher network as well as a KD loss. In this work, we use ℓ_2 loss as the KD loss and set λ_{kd} as the adjustable weight of the KD loss. The overall loss function aggregates the loss over all the samples; that is,

$$\mathcal{L} = \Sigma_i \left(\lambda_{kd} \left\| \mathbf{F}_s^i - \widehat{\mathbf{F}}_t^i \right\|_2 + \mathcal{L}_{pred}^i \right). \quad (3)$$

6 Experiments

6.1 Experimental Setup

Datasets. We consider two widely used trajectory forecasting benchmarks, *nuScenes* [3] and *Argoverse* [6]. *nuScenes* collected 1000 scenes. Each scene is 20s long with a sampling rate of 2 Hz. The instances for trajectory forecasting tasks were split into train, validation and test splits, which respectively entail 32186, 8560 and 9041 instances. Each agent in the instances had 2s’ observed trajectories and the ground truth of 6s’ future trajectories. *Argoverse* collected over 30K scenarios. Each scenario is sampled at 10 Hz. The train/ val/ test splits had 205942/ 39472/ 78143 sequences respectively. Each agent in the scene had 2s’ observed trajectories and the ground-truth of 3s’ future trajectories.

Database Construction. To evaluate local-behavior-aware (LBA) framework without ground-truth data leakage, we only use the 2s’ observed trajectories of all agents in each data split to build the behavior database \mathcal{D}_B for each corresponding phase (e.g. testing phase only uses the test split). For the local-behavior-free (LBF) framework, we only use the observed trajectories from the training split during the training phase. Please see appendix for detailed information of behavior database construction.

Metrics. We adopt three widely used metrics for quantitative evaluation: minimum average displacement error among K predictions (minADE_K), minimum final displacement error among K predictions (minFDE_K) and the missing rate (MR). minADE evaluates the minimum average prediction error of all timestamps among the predicted K trajectories; minFDE is the minimum error of the final position among the K predictions; MR_K is the ratio of agents whose minFDE_K is larger than 2.0 meters.

6.2 Implementation Details

We pick three SOTA trajectory forecasting methods: LaneGCN [23], DenseTNT [15] and P2T [9], and adapt them to our behavior-aware framework. We use their official code packages as the implementation start code. The baseline performances in Table 1 and 2 are reproduced by ourselves for variables controlling.

LaneGCN and DenseTNT are graph-based methods. We use stacked linear residual blocks mentioned in [23] as our behavior encoder $E_{\mathcal{B}}$ for both methods. In the LBF prediction, we use attention based architecture to implement the behavior estimator $\Omega_{\mathcal{B}}$ as well as the auxiliary fusion module Ψ .

P2T is a rasterization-based method. We use *behavioral probability map* to represent the local behavior data, and ResNet [17] as the encoder backbone to extract features from the behavioral probability map. For the LBF prediction, we use three 1D Convolutional Layers to implement the behavior estimator $\Omega_{\mathcal{B}}$.

To train the network, we adopt the hyper-parameter configuration from each method’s official instructions. More implementation details are in the appendix.

6.3 Evaluation

We evaluate LaneGCN and P2T on nuScenes (see the quantitative results in Table 1). For Argoverse, we evaluate LaneGCN and DenseTNT (see Table 2). We also show the qualitative evaluation in Fig. 7.

In all experiments, when upgraded to either the LBA or LBF framework, the baseline methods significantly improve in performance. Unconventionally, our proposed frameworks bring consistent improvements to various models (P2T, LaneGCN, DenseTNT) across **all** metrics. This is a substantive progress compared to previous methods, because SOTA methods [12,14,41,43] usually only show improvements in one or some metrics but not all metrics. Furthermore, on both datasets, the gains brought by local behavior data are consistently larger on $K = 1$ metrics than on other metrics. This may result from the raise of average performance of the worst prediction among K predictions, as local behavior data efficiently narrows down the solution search space (explained in Sec 1).

Table 1: Evaluation results on nuScenes [3] dataset test split.

Method	Framework	minADE ₁	minFDE ₁	minADE ₁₀	minFDE ₁₀
P2T [9]	Baseline	4.60	10.80	1.17	2.15
	LBA	3.78 ↓ 18%	9.25 ↓ 14%	1.08 ↓ 8%	2.04 ↓ 5%
	LBF	4.04 ↓ 12%	9.54 ↓ 12%	1.15 ↓ 2%	2.11 ↓ 2%
LaneGCN [23]	Baseline	6.17	12.34	1.82	2.98
	LBA	2.72 ↓ 56%	6.78 ↓ 45%	0.95 ↓ 48%	1.85 ↓ 38%
	LBF	5.58 ↓ 10%	11.47 ↓ 7%	1.67 ↓ 8%	2.66 ↓ 11%

Interestingly, the prediction performance of the LBF framework occasionally surpasses that of the LBA framework, even though the LBF framework lacks local behavior data input. Table 2 shows an example of this case, with the LaneGCN results on Argoverse. Our educated conjecture is that sometimes the LBF framework enjoys more data representativeness thanks to the reconstructed behavioral features; whereas, in the meantime, the LBA framework may be using pre-gathered local behavior data of a small size in its testing phase.

Besides the comparison with the baselines, we also show the comparison with the other published works on the benchmarks; see Tables 3 and 4.

Table 2: Evaluation results on Argoverse [6] dataset test split.

Method	Framework	minADE ₁	minFDE ₁	minADE ₆	minFDE ₆
LaneGCN [23]	Baseline	1.74	3.89	0.87	1.37
	LBA	1.64 ↓ 6%	3.61 ↓ 7%	0.84 ↓ 3%	1.30 ↓ 5%
	LBF	1.61 ↓ 7%	3.54 ↓ 9%	0.85 ↓ 2%	1.31 ↓ 4%
DenseTNT [15]	Baseline	1.70	3.72	0.90	1.33
	LBA	1.65 ↓ 3%	3.57 ↓ 4%	0.88 ↓ 2%	1.26 ↓ 5%
	LBF	1.67 ↓ 2%	3.63 ↓ 2%	0.89 ↓ 2%	1.29 ↓ 3%

Table 3: nuScenes benchmark comparison.

Method	minADE ₁₀	MR ₅	MR ₁₀	minFDE ₁
P2T[9]	1.16	64%	46%	10.50
MHA-JAM[29]	1.24	59%	46%	8.57
SGNet[38]	1.40	67%	52%	9.25
Trajectron++[33]	1.51	70%	57%	9.52
M-SCOUT[4]	1.92	78%	78%	9.29
GOHOME[12]	1.15	57%	47%	6.99
P2T-LBA	1.08	57%	41%	9.25
P2T-LBF	1.15	61%	46%	9.37
LaneGCN-LBA	0.95	49%	36%	6.78
LaneGCN-LBF	1.67	75%	68%	11.47

Table 4: Argoverse benchmark comparison.

Method	minADE ₁	minADE ₆	Brier-FDE ₆
LaneRCNN[43]	1.69	0.90	2.15
DenseTNT[15]	1.68	0.88	1.98
GOHOME[12]	1.69	0.94	1.98
MMTransformer[24]	1.77	0.84	2.03
LaneGCN[23]	1.71	0.87	2.06
LaneGCN-LBA	1.64	0.84	2.00
LaneGCN-LBF	1.61	0.85	2.00
DenseTNT-LBA	1.65	0.88	1.93
DenseTNT-LBF	1.67	0.89	1.96

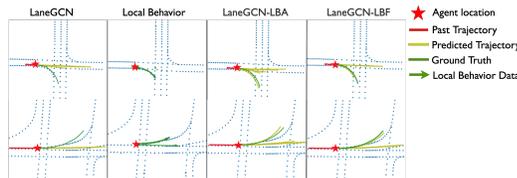


Fig. 7: LaneGCN qualitative results on Argoverse val split.

 Table 5: minFDE₁ of LaneGCN on Argoverse val split among agents having different sizes of local behavior data.

Size	% of agents	Baseline	LBA
[0, 4)	24%	3.33	3.36 ↑ 1%
[4, 8)	14%	3.25	3.19 ↓ 2%
[8, 12)	10%	3.22	3.13 ↓ 3%
[12, 16)	9%	3.00	2.87 ↓ 4%
[16, ∞)	43%	2.66	2.60 ↓ 2%

6.4 Ablation Study

We conduct the ablation study on the hyper-parameters regarding local behavior data generation and our prediction frameworks.

Local Behavior Data Size. Regarding the relationship between prediction performance and the local behavior data size of each agent, Table 5 shows: when there is few local behavior data, prediction is not as accurate as the baseline; but, even a small size of local behavior data can improve the prediction.

Local Range ϵ . The adjustable parameter ϵ defines the local radius (see Sec 3). See Table 6 for the impact of ϵ on LaneGCN performance with Argoverse test split. When ϵ increases, the performance of the LBA framework first go up but later go down. It matches our intuition that a large ϵ will introduce distractions. The LBF framework, however, shows less confusion brought by the pre-trained LBA network, demonstrating stronger robustness to ϵ .

Knowledge Distillation Parameters. We study the impact of knowledge-distillation-related parameters, i.e., the loss weight λ_{kd} and the number of times that intermediate features are involved in the KD loss (denoted as KD times). See the results in Table 7. Detailed information about the choices of intermediate features is in the appendix. We see that firstly, the knowledge distillation structure does help the framework infer the impact of local behavior data. The framework, across all parameter settings in Table 7, outperforms the baseline method (in Table 2, row 1) and the model without KD loss (in Table 7, row 1). Secondly, the comparatively similar results across all the settings show that the LBF framework is relatively robust to the KD hyper-parameters.

Table 6: Impact of the behavioral data’s local range. A larger ϵ brings a larger size of local behavior data, but when ϵ is too large, it can also introduce confusion due to the loss of the data locality.

Framework	ϵ	minADE ₁	minFDE ₁
LBA	0.5m	1.64	3.61
	1.0m	1.62	3.58
	1.5m	1.71	3.86
LBF	0.5m	1.61	3.54
	1.0m	1.62	3.57
	1.5m	1.63	3.59

Table 7: Impact of the number of times that KD supervision gets applied in the training phase and the weight of the KD loss.

KD times	λ_{kd}	minADE ₁	minFDE ₁
N/A	0	1.68	3.72
2	1	1.62	3.57
2	1.5	1.61	3.54
2	2	1.62	3.56
1	1.5	1.64	3.61
3	1.5	1.61	3.56

7 Conclusion

In this work, we re-introduce the local historical trajectories as a new type of data input to the trajectory forecasting task, referred as *local behavior data*. To adapt to this new data input and fully exploit its value, we propose a *behavior-aware* framework and a *behavior-free* framework for trajectory forecasting. The behavior-free framework, especially, adopts a knowledge-distillation architecture to estimate the impact of local behavior data. Extensive experiments on published benchmarks validate that the proposed frameworks significantly improve the performances of SOTA methods on prevalent metrics.

Limitations. Local historical information reveals local motion patterns with high fidelity, but there are always outliers. For use cases of great safety concerns (e.g. autonomous driving), historical data may provide good reference but should not be the only reference. Also, the motion patterns of a certain location can vary over time. To optimize the benefits of the LBA and LBF framework, future research should explore the historical data gathering strategies.

Acknowledgement: National Natural Science Foundation of China under Grant 62171276, the Science and Technology Commission of Shanghai Municipal under Grant 21511100900, CCF-DiDi GAIA Research Collaboration Plan 202112 and CALT 2021-01.

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 961–971 (2016)
2. Bennewitz, M., Burgard, W., Thrun, S.: Learning motion patterns of persons for mobile service robots. In: Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292). vol. 4, pp. 3601–3606. IEEE (2002)
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020)
4. Carrasco, S., Llorca, D.F., Sotelo, M.Á.: Scout: Socially-consistent and understandable graph attention network for trajectory prediction of vehicles and vrus. arXiv preprint arXiv:2102.06361 (2021)
5. Casas, S., Luo, W., Urtasun, R.: Intentnet: Learning to predict intention from raw sensor data. In: Conference on Robot Learning. pp. 947–956. PMLR (2018)
6. Chang, M.F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al.: Argoverse: 3d tracking and forecasting with rich maps. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8748–8757 (2019)
7. Chen, G., Choi, W., Yu, X., Han, T., Chandraker, M.: Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems* **30** (2017)
8. De Leege, A., van Paassen, M., Mulder, M.: A machine learning approach to trajectory prediction. In: AIAA Guidance, Navigation, and Control (GNC) Conference. p. 4782 (2013)
9. Deo, N., Trivedi, M.M.: Trajectory forecasts in unknown environments conditioned on grid-based plans. arXiv preprint arXiv:2001.00735 (2020)
10. Deo, N., Wolff, E., Beijbom, O.: Multimodal trajectory prediction conditioned on lane-graph traversals. In: 5th Annual Conference on Robot Learning (2021)
11. Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., Schmid, C.: Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11525–11533 (2020)
12. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: Gohome: Graph-oriented heatmap output for future motion estimation. arXiv preprint arXiv:2109.01827 (2021)
13. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: Home: Heatmap output for future motion estimation. arXiv preprint arXiv:2105.10968 (2021)
14. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: Thomas: Trajectory heatmap output with learned multi-agent sampling. arXiv:2110.06607 (2021)
15. Gu, J., Sun, C., Zhao, H.: Densent: End-to-end trajectory prediction from dense goal sets. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15303–15312 (2021)
16. Hao, Y., Fu, Y., Jiang, Y.G., Tian, Q.: An end-to-end architecture for class-incremental object detection with knowledge distillation. In: 2019 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6. IEEE (2019)

17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
18. He, T., Shen, C., Tian, Z., Gong, D., Sun, C., Yan, Y.: Knowledge adaptation for efficient semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 578–587 (2019)
19. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
20. Hu, Y., Chen, S., Zhang, Y., Gu, X.: Collaborative motion prediction via neural motion message passing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6319–6328 (2020)
21. Kim, B., Kang, C.M., Kim, J., Lee, S.H., Chung, C.C., Choi, J.W.: Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). pp. 399–404. IEEE (2017)
22. Li, Y., Ren, S., Wu, P., Chen, S., Feng, C., Zhang, W.: Learning distilled collaboration graph for multi-agent perception. arXiv preprint arXiv:2111.00643 (2021)
23. Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R.: Learning lane graph representations for motion forecasting. In: European Conference on Computer Vision. pp. 541–556. Springer (2020)
24. Liu, Y., Zhang, J., Fang, L., Jiang, Q., Zhou, B.: Multimodal motion prediction with stacked transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7577–7586 (2021)
25. Liu, Y., Chen, K., Liu, C., Qin, Z., Luo, Z., Wang, J.: Structured knowledge distillation for semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2604–2613 (2019)
26. Liu, Y., Dong, X., Lu, X., Khan, F.S., Shen, J., Hoi, S.: Teacher-students knowledge distillation for siamese trackers. arXiv preprint arXiv:1907.10586 (2019)
27. Luo, Y., Cai, P., Bera, A., Hsu, D., Lee, W.S., Manocha, D.: Porca: Modeling and planning for autonomous driving among many pedestrians. *IEEE Robotics and Automation Letters* **3**(4), 3418–3425 (2018)
28. Marchetti, F., Becattini, F., Seidenari, L., Bimbo, A.D.: Mantra: Memory augmented networks for multiple trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7143–7152 (2020)
29. Messaoud, K., Deo, N., Trivedi, M.M., Nashashibi, F.: Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. arXiv preprint arXiv:2005.02545 (2020)
30. Nikhil, N., Tran Morris, B.: Convolutional neural network for trajectory prediction. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops. pp. 0–0 (2018)
31. Phan-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M.: Covernet: Multimodal behavior prediction using trajectory sets. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14074–14083 (2020)
32. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413 (2017)
33. Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16. pp. 683–700. Springer (2020)

34. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. *IEEE transactions on neural networks* **20**(1), 61–80 (2008)
35. Sun, C., Karlsson, P., Wu, J., Tenenbaum, J.B., Murphy, K.: Stochastic prediction of multi-agent interactions from partial observations. *arXiv preprint arXiv:1902.09641* (2019)
36. Sun, C., Shrivastava, A., Vondrick, C., Sukthankar, R., Murphy, K., Schmid, C.: Relational action forecasting. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 273–283 (2019)
37. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: *Advances in neural information processing systems*. pp. 5998–6008 (2017)
38. Wang, C., Wang, Y., Xu, M., Crandall, D.J.: Stepwise goal-driven networks for trajectory prediction. *arXiv preprint arXiv:2103.14107* (2021)
39. Xu, C., Li, M., Ni, Z., Zhang, Y., Chen, S.: Groupnet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 6498–6507 (2022)
40. Xu, C., Mao, W., Zhang, W., Chen, S.: Remember intentions: Retrospective-memory-based trajectory prediction. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 6488–6497 (2022)
41. Ye, M., Cao, T., Chen, Q.: Tpcn: Temporal point cloud networks for motion forecasting. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 11318–11327 (2021)
42. Yu, C., Ma, X., Ren, J., Zhao, H., Yi, S.: Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In: *European Conference on Computer Vision*. pp. 507–523. Springer (2020)
43. Zeng, W., Liang, M., Liao, R., Urtasun, R.: Lanercnn: Distributed representations for graph-centric motion forecasting. *arXiv preprint arXiv:2101.06653* (2021)
44. Zhao, H., Gao, J., Lan, T., Sun, C., Sapp, B., Varadarajan, B., Shen, Y., Shen, Y., Chai, Y., Schmid, C., et al.: Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294* (2020)
45. Zhao, H., Wildes, R.P.: Where are you heading? dynamic trajectory prediction with expert goal examples. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 7629–7638 (2021)