TPCN: Temporal Point Cloud Networks for Motion Forecasting

Maosheng Ye[1‡] Tongyi Cao[2] Qifeng Chen[1]
[1]Hong Kong University of Science and Technology [2]DEEPROUTE.AI

Abstract

We propose the Temporal Point Cloud Networks (TPCN), a novel and flexible framework with joint spatial and temporal learning for trajectory prediction. Unlike existing approaches that rasterize agents and map information as 2D images or operate in a graph representation, our approach extends ideas from point cloud learning with dynamic temporal learning to capture both spatial and temporal information by splitting trajectory prediction into both spatial and temporal dimensions. In the spatial dimension, agents can be viewed as an unordered point set, and thus it is straightforward to apply point cloud learning techniques to model agents' locations. While the spatial dimension does not take kinematic and motion information into account, we further propose dynamic temporal learning to model agents' motion over time. Experiments on the Argoverse motion forecasting benchmark show that our approach achieves state-of-the-art results.

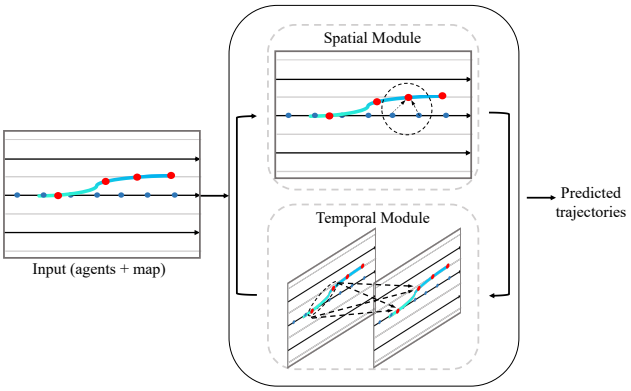

Figure 1. A high-level illustration of our approach. Red points represent the agent history trajectory points, while blue points are discrete map lane points. We use a spatial module based on point cloud learning to extract geometric features and a temporal module to extract sequential features. Both modules utilize the output of the other module and propagate mutually to output future trajectory points.

1. Introduction

Motion forecasting in autonomous driving concerns future trajectories of agents, including vehicles and pedestrians. For a self-driving car, the predicted future trajectories of surrounding traffic participants serve as key information to plan its future trajectories. A self-driving car should be able to predict the distribution or a few possible future trajectories of each agent as the future is full of uncertainty, given the relevant sensor input information in the past.

Traditional methods for motion forecasting [15, 28, 34, 39] are based on kinematic constraints and road map information with handcrafted rules. Though these approaches are sufficient in many simple situations, they fail to capture the rich behavior strategies and interaction in complex urban scenarios. Great progress has been made to explore the power of data-driven methods in motion forecasting with deep learning [2, 4, 6, 7, 25]. These methods encode the agents (e.g., vehicles, pedestrians, and cyclists) and high-definition map (HD map) information by rasterizing the corresponding elements (lanes, crosswalks) as lines and polygons with different colors. A standard image backbone network [12, 31] is then applied to the rasterized image to extract the map and agent features and perform motion prediction.

However, the rasterized image is an overly complex representation for environment and agent history and requires significantly more computation and data to train and deploy. More succinct representations have been explored to avoid this heavy process. VectorNet [10] proposes a vector representation to exploit the spatial locality of individual road components with graph neural networks. LaneConv [19] constructs a lane graph from vectorized map data and proposes LaneGCN to capture the topology and long dependency of the agents and map information. Both VectorNet [10] and LaneConv [19] can be viewed as extensions of graph neural network in prediction with strong capability to extract spatial locality. However, both works fail to fully utilize the temporal information of agents with less focus on temporal feature extraction.

In this work, we extend ideas from 3D point cloud learning to the motion forecasting task. Previous works on point cloud networks focus on spatial points. We extend the met-

‡Part of the work was done during an internship at DEEPROUTE.AI.

ric space to the joint spatial-temporal space and represent the agents' history observations and map data as points in this space. Since the raw input data of prediction is a set of points that contain different agents with historical observations and map data, spatial and temporal learning will be two key components in prediction learning. Ignoring either information will lead to information loss and reduce the model's capability of context learning.

In order to combine spatial and temporal learning in a flexible and unified framework, we propose Temporal Point Cloud Networks (TPCN). Compared with GCN based methods [10, 19], our TPCN does not manually specify the interaction structures (e.g., connectivity in the graph) and avoid the complex correlation learning process. TPCN models the prediction learning task as joint learning between a spatial module and a temporal module. In the spatial module, note that the waypoints and map points have very similar properties as point clouds, both being sparse and permutation invariant, and have a strong geometric correlation. Thus, point cloud learning strategies can be effective for spatial feature extractions. Instead of directly applying works [26, 27, 33, 36] whose computation cost is high, we propose our novel spatial learning layer, namely **Dual-representation Spatial Learning** to obtain pointwise and voxelwise features through point cloud learning. Meanwhile, we propose **Dynamic Temporal Learning** in the temporal module to effectively extract the time-series information and motion estimation. Compared with traditional *Hard Temporal Learning* [1, 18, 19], our dynamic temporal learning layer naturally handles variant time lengths of different agents in the same sequence without the need to pad the history. By switching between the two modules, the spatial features and temporal features from these two modules are propagated mutually, each module taking the features of the other module as input. As such, spatial learning will utilize the temporal information (e.g., motion state), while the temporal learning will have some spatial guidance (e.g., map information), namely **Joint Learning**. Fig. 1 illustrates the overall architecture of our approach. Note that we model the selection of multi-modal trajectories problem as displacement regression rather than classification.

Our contributions are summarized as follows:

- We propose a novel and flexible architecture for prediction learning, which models the complex process as joint spatial and temporal learning. Dual-representation Spatial Learning for feature extraction of waypoints and map data is proposed as the spatial module. Meanwhile, we propose novel Dynamic Temporal Learning, consisting of Multi-interval Learning and Instance Pooling.

- We propose displacement prediction for multi-modal trajectories selection, which alleviates the hard assignment in classification through regression.

- Extensive experiments are conducted on the large-scale Argoverse motion forecasting benchmark to show the effectiveness of our approach.

2. Related Work

Most existing works on prediction can be roughly divided into three categories according to their representation and architecture.

Rasterization based methods. Rasterization BEV images are the most common and direct ways to represent the structure of map and neighborhood relationships among agents. Some methods [2, 3, 21, 25] render the HD map elements (junctions, lanes) as BEV images with different colors according to their types. In the format of images, a series of standard convolution layers or backbones [12] can be applied to simplify the prediction task as trajectories selection and offset regression problems. Furtherly, some works [4, 25] propose to use anchor trajectories with human prior knowledge based on motion constraint to make the results more consistent with the current dynamic state and alleviate the difficulties in multi-modal prediction. However, these approaches have internal limitations since the performance is highly related to the spatial resolutions of the rasterized images. The temporal information can not be represented or modeled in the rendered images intuitively.

GCN based methods. Graph Convolutional Network (GCN) [8, 13, 30] nowadays gains its popularity in processing non-structural data and dealing with correlation relationship. Compared with traditional CNN, GCN shows its great promise in capturing the spatial locality on both euclidean and non-euclidean structures. With adjacency matrices, GCN focuses on learning the relationship between graph nodes and vertices. VectorNet [10] introduces a novel vector representation and applies a graph neural network to predict the intent of vehicles. M Liang *et al.* [19] proposes LaneGCN based on GCN, which is a specialized version designed for lane graphs. In order to capture the complex topology of HD maps effectively, it combines with multi-scale dilated convolution. Social-STGCNN [24] models interactions as a graph by defining a novel kernel function to learn spatial and temporal patterns from pedestrian trajectories and behavior. However, GCN based methods are often faced with an efficiency problem when dealing with large-scale scenarios that contain lots of nodes and vertices.

Hybird Methods. To provide more interpretable and kinematic constraints, some works [3, 22, 38, 37] decouple the task as two-stage. Firstly, they discretize search space via uniform sampling or based on HD maps to generate some proposals. Compared with anchor trajectories, these proposals can be more stable and more informative to capture the uncertainty of the multi-modal prediction. Secondly,

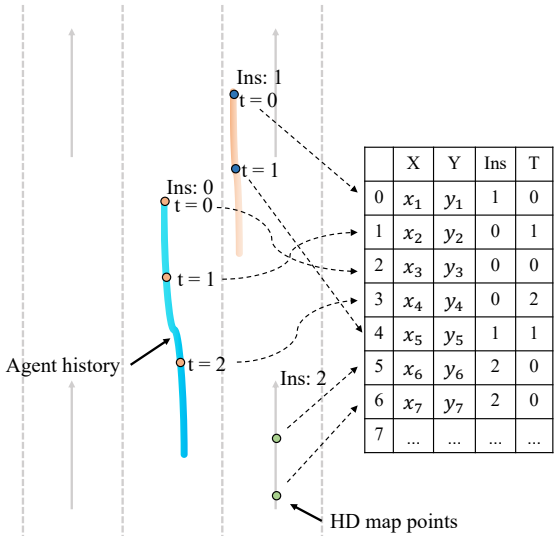

Figure 2. An example that shows the instance time indexing system. The points with the same color belong to the same instance.

they encode the HD map and agents via vector representation or rasterized images to furtherly refine each proposal. To some extent, two-stage methods can fully incorporate expert knowledge or classical planning or prediction approaches. On the other hand, it means the final outputs of refinement networks have a strong dependency on the quality of the proposals, which requires a reasonable sampling strategy or mature planning module.

In comparison with these methods above, our TPCN has a novel representation and architecture, including spatial learning based on dual-representation point cloud learning and dynamic temporal information learning. We split the task into submodules to capture both spatial and temporal information effectively.

3. Approach

The overall network architecture of our TPCN approach consists of two modules: 1) Dual-representation Spatial Learning and 2) Dynamic Temporal Learning. The **Dual-representation Spatial Learning** serves as the spatial module to model spatial features, and the **Dynamic Temporal Learning** is the temporal module to extract temporal features. Both modules are integrated to propagate features mutually in spatial and temporal dimensions to achieve **Joint Learning**. As shown in Fig. 1, each module takes the pointwise features of the other module as input to generate corresponding pointwise output features, which is a natural foundation for fusing pointwise context information across multiple domains.

Typically, the motion prediction task will contain agents' data and environment information encoded with map data. We define an agent instance as an agent with a set of trajectory points. While map data refers to static objects without temporal information, a map instance can be described as an ordered point set for one specific element (e.g., a piece of lane centerline points). Thus, agent data will be represented by $\{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \ldots, \mathbf{p}_{i,T_i}\}$, where $\mathbf{p}_{i,t}$ means the i-th agent's coordinate at time t, and T_i is the time sequence length for i-th agent. Meanwhile, we represent map data in the format of $\{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \ldots, \mathbf{p}_{i,N_i}\}$, where $\mathbf{p}_{i,j}$ is the j-th point of i-th map element instance with N_i points in total.

Voxelization. Given a grid size s, we can construct the mapping from a point $\mathbf{p}_i = (x_i, y_i)$ to its voxel index $\mathbf{v}_i$:

$$\mathbf{v}_i = (\lfloor x_i/s \rfloor, \lfloor y_i/s \rfloor), \tag{1}$$

where $\lfloor \cdot \rfloor$ is a floor function. Thus we can build a hash table for the conversion between point coordinate space and voxel coordinate space $\{\mathbf{p}_i, \mathbf{v}_i\}$.

Instance Time Indexing. Apart from voxel and point spaces, we formulate the temporal space indexing system to address the dynamic and different sequential lengths of different agents. We represent all the instances over time as $\{\mathbf{m_i}\}$, where the i-th element $\mathbf{m}_i = (ins_i, t_i)$ is an instance time index referring to the t_i-th trajectory point of instance ins_i. See Fig. 2 for an example. A special case is that t_i is zero for all static instances in the map data.

Both voxelization and instance time indexing aim to provide space mapping or hashing: voxelization maps from the Cartesian coordinate system to a structural grid representation and instance time indexing maps from an index to a trajectory point of an instance. These mapping or hashing systems provide convenience for feature transformation between different spaces or representations.

3.1. Dual-representation Spatial Learning

For the spatial module, we choose a point cloud learning approach to retrieve the spatial features of waypoints and map data with their locality and spatial geometry. Inspired by recent multiple representations learning [29, 32] that shows advantages over a single representation [27, 33], we propose a dual-representation method to leverage the merits of mutually complementary information between voxel and point representations. The overall architecture of the spatial module is illustrated in Fig. 3.

Pointwise Feature Learning. Pointwise features maintain geometric information and neighborhood relationship for interactions among points. To this end, we utilize the hierarchical feature learning from PointNet++ [27] for pointwise feature extraction at different levels of the local neighborhood to exploit more local structure and correlation.

Point-Voxel Feature Propagation. Inspired by HVNet [36], we can transform pointwise features to voxel space by scattering operations. In this process, we maintain the hash table for each point, which stores the key and value pairs to map a Cartesian coordinate to a voxel index.

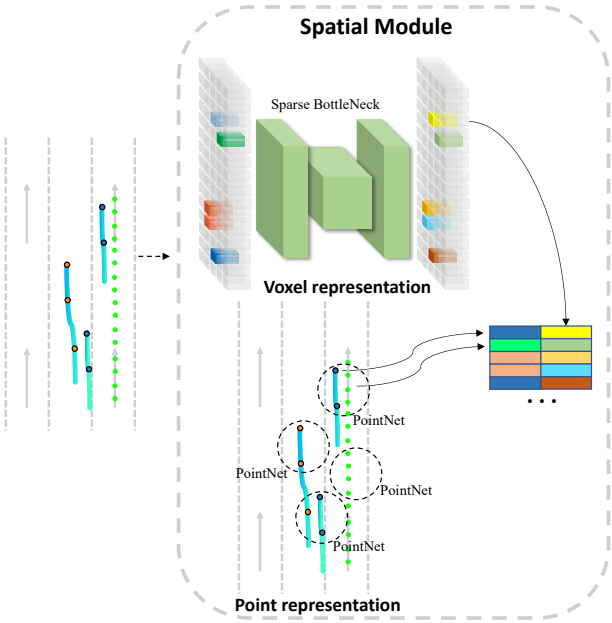

Figure 3. Dual-representation Spatial Learning.

Then, following the Feature Transformation Propagation algorithm (FTP) 1, we put all the points that share the same voxel index into the same cluster (each voxel may contain more than one point). Finally, we calculate the mean features over the points in each cluster as final features for the corresponding voxel, which is similar to average pooling.

Algorithm 1 Feature Transformation Propagation

Require: All pointwise features PF with corresponding indexing hash table H

clusters = $\{\}$
for each $PF_i \in PF$ **do**
    clusters.at(H_i).append(PF_i)
end for
for each $H_i \in H$ **do**
    clusters.at(H_i) = mean(clusters.at(H_i))
end for
return clusters

Voxelwise Feature Learning. Voxelwise features have a strong capability to extract semantic context information [20]. Most existing works apply 2D or 3D CNN to the rasterized or voxelized images for feature extraction in order to exploit the structural data with the current popular backbone [12]. One of the key parameters for these methods is the grid size. Smaller grid size leads to less information loss but brings higher computation cost and latency. Meanwhile, compact 2D or 3D tensor of rasterized images neglect the sparsity of the input data and involve plenty of non-activated regions that may mess up feature learning.

Therefore, we employ the sparse convolution [11, 35] as our feature extractor to afford a smaller grid size for fine-grained voxelwise features. Furthermore, we build a Sparse BottleNeck network with skip connections, which replaces the bottleneck blocks with sparse convolutions in ResNet [12]. Stacking Sparse BottleNeck layers not only quickly expands the receptive field at a low computational cost but also keeps the activation sparse.

Voxel-Point Feature Propagation. With voxelwise features, feature propagation from voxel representation to point representation can be performed by the naive nearest neighbor interpolation. PVCNN [20] interpolates the pointwise features with corresponding neighboring voxelwise features. Since the interpolation weights are based on the physical distance to the neighboring grids accordingly, we extend the weights to be learnable by applying MLP to distance embedding that also concatenates associated voxelwise features.

Dual-representation Fusion. With pointwise and voxelwise features, we fuse these two types of features by feature concatenation. Thus, we obtain the features with dual representations and higher context information, which will be passed to the next stage of dynamic temporal learning.

3.2. Dynamic Temporal Learning

In a motion prediction task, different agents have different lengths of observed past trajectories due to the different lifespan of each agent. Existing methods [14, 19] pad the agents' data whose size is smaller than a given size T with zero in order to process data with the same length. We name this operation as *Hard Temporal Learning* (HTL). HTL has two main drawbacks: 1) padding data will introduce extra unnecessary computation cost, especially when the agents only appear in very few shots; 2) processing padded data will lead to the feature confusion problem, especially when invalid padding data involves the feature propagation. HTL forces the network to capture useless information.

We propose Dynamic Temporal Learning to address these limitations. Instead of padding, we only preserve the originally provided information without the requirement for a fixed time buffer size for each agent data. Therefore, we can retain each agent with a dynamic time sequence length. Furthermore, Dynamic Temporal Learning consists of the following two parts.

Multi-interval Learning. The time interval is a key factor for time series data feature learning since it determines the time window size, similar to the receptive field in a CNN. Inspired by the multi-scale or multi-resolution hierarchy that has been proven its effectiveness in capturing local correlations and context, we also exploit Multi-interval Learning with prediction data. However, the main challenge for our application lies in the dynamic property. As a result, we utilize the Instance Time Indexing system and previously introduced FTP 1, and then perform high ef-

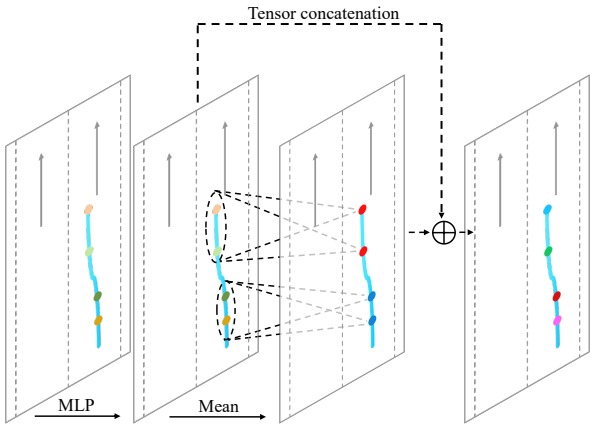

Figure 4. An example of multi-interval learning with an interval size of 2. $\oplus$ means tensor concatenation and points stand for the sequential historical waypoint data for one specific instance.

ficient Multi-interval Learning (MIL) according to the following Multi-interval Learning algorithm 2. Given a set of different intervals, we first regroup the instance time indexing to ensure that each point with the same instance ID and timestamp within the same interval will be clustered into the same group. Consequently, we employ the FTP algorithm 1 to obtain the mean features of the corresponding interval and map to pointwise features by the following two steps; 1) slicing by using the inverse hash or indexing table to gather the pointwise features and 2) concatenating the input tensor as a shortcut connection. Note that the output feature of the current time interval will be passed to the next level as input to progressively and aggressively capture features at increasingly larger intervals over a multi-resolution hierarchy. Fig. 4 illustrates a special case for Multi-interval Learning with an interval size of 2.

Algorithm 2 Multi-interval Learning

Require: All pointwise features PF with instance time indexing **m** and the predefined set of time intervals T

Ensure: All pointwise output features O_p

$O_p = PF$
for each $t \in$ T **do**
  $m^* = (m[0], \lfloor m[1]/t \rfloor)$
  $F_t = \text{MLP}(O_p)$
  $O = \text{FTP}(F_t, m^*)$
  $O_p = \text{slice}(O, m^*)$
  $O_p = \text{concat}(O_p, F_t)$
end for
return O_p

Instance Pooling. Instance-level features that aggregate information of an instance over time is a crucial component beyond the point, voxel, and time level features, as the fundamental data in this task is composed of instances as mentioned [10]. Moreover, instance-level features can capture the long-range or large time interval dependency under some scenarios. For example, when the start point and endpoint of a lane centerline are far away from each other, it is hard to design a suitable architecture to handle this dependency or correlation. To this end, we propose our Instance Pooling (Ins-Pool) to provide a more flexible way for the instance feature extraction. In this process, the Instance Pooling can be viewed as a special case of Multi-interval Learning as it applies pooling operation along with the set of points with the same instance ID to extract global entity features.

For implementations of the above algorithm 1 and 2, we optimize and utilize GPU-based scatter, gather and hash table operations to increase runtime efficiency by parallel.

3.3. Displacement Prediction and Learning

A prediction header is built to predict the final forecasting that takes the features from fusion features of spatial and temporal modules as input. Most existing works [4, 6, 19] predict K possible trajectories with their confidence scores respectively to model the multi-modal property. The loss is often split into regression and classification parts. During training, the loss will only be backpropagated at the trajectory that has the minimum displacement error at the endpoint. However, loss of classification for trajectory selection based on positive and negative sample assignment is not reasonable and too handcrafted, especially when two trajectories have very close displacement error. Inspired by the popular concept of IoU prediction [16] in object detection, we predict the final displacement error rather than the trajectory's confidence. Therefore, we alleviate the problem of classification that requires hard assignments by displacement regression. We define the output of our prediction header as τ_{disp} and τ_{reg}:

$$\tau_{disp} = \{d_0, d_1, \ldots, d_{K-1}\}, \tag{2}$$

$$\tau^k_{reg} = \{(x^k_1, y^k_1), (x^k_2, y^k_2), \ldots, (x^k_T, y^k_T)\}, \tag{3}$$

where τ^k_{reg} is the k-th predicted trajectory among K possible trajectories, which contains T waypoints with 2D (x, y) vector representation. τ_{disp} is the predicted displacement error at endpoint associated with each possible trajectory.

Loss Functions. We sum the trajectory regression $\mathcal{L}_{reg}$ and trajectory displacement $\mathcal{L}_{disp}$ as final loss for training.

For trajectory regression, we follow the previous routine that we choose the best trajectory k^* whose displacement error with ground-truth trajectory is minimum:

$$\mathcal{L}_{reg} = \frac{1}{T}\sum_{i=1}^{T} \rho(x^{k^*}_i, x^{gt}_i) + \rho(y^{k^*}_i, y^{gt}_i), \tag{4}$$

where (x^{gt}_i, y^{gt}_i) represents the ground-truth coordinate at

timestamp i, and ρ is a smooth L_1 loss function:

$$\mathcal{L}_{disp} = \frac{1}{K}\sum_{i=1}^{K}\rho(d_i, d_i^*), \tag{5}$$

where d_i^* is the ground-truth displacement between the k-th trajectory and ground-truth trajectory. For inference, we sort the trajectories according to their predicted displacement in an ascending order.

4. Experiments

In this section, we conduct extensive experiments on Argoverse [5], which is one of the largest public motion forecasting datasets with rich HD map information. We also evaluate the proposed modules with ablation studies to show their effectiveness.

4.1. Experimental setup

Dataset. Argoverse [5] is a public motion forecasting dataset. It has more than 300K 5-second sequences collected in Pittsburgh and Miami. For each sequence, the sampling rate is $10Hz$, meaning that the interval of the same object that appears in the next timestamp is about 0.1s. There are multiple objects with centroid coordinates of time series trajectories within one sequence, with each object tagged as one of the three types, agent, AV, and others. Moreover, each sequence has only one object, tagged with type agent that is required to be predicted the next 3 seconds future horizon in this challenge. We name this agent as the target agent and other vehicles, including AV as other agents similar to Gao et al. [10]. The whole sequences can be split into training, validation, and test set, with 205942, 39472, and 78143 sequences, respectively. The training and validation sets provide the full 5 seconds trajectories for each target agent data. For the test set, only the first 2 seconds trajectories are given. In addition to trajectories data, we could query the map data represented by lane centerlines points via a given location and city name.

Metrics. Following the previous works, we also adopt the widely used metrics Average Displacement Error (ADE) and Final Displacement Error (FDE) as criteria. ADE is the average displacement error with ground-truth labels over the entire time steps, and FDE is defined as the displacement error at the endpoint. For multi-modal prediction evaluation on Argoverse, minADE, and minFDE are also used since it allows multiple forecasted trajectories. During the evaluation, it selects K trajectories and computes minimum ADE and minimum FDE as metrics. Miss Rate(MR) is also considered in this task, which is the percentage of the predicted trajectories within a certain threshold $(2m)$ of ground truth according to endpoint error. We take minADE, minFDE, MR for $K = 1$ and $K = 6$ as evaluation metrics in our experiments.

Data split	Speed distribution(%)			
	$[0, 5)$	$[5, 10)$	$[10, 15)$	≥ 15
Train	26.4	42.4	26.7	4.5
Val	18.6	38.7	29.8	12.9
Test	33.4	52.6	13.1	0.9

Table 1. Speed distribution on different data splits.

Data Augmentation. As shown in Tab. 1, the speed distribution of target agents varies on different data splits. The validation set has a larger proportion of high-speed agents, while the average speed of agents on the test set is lower. Therefore, we apply global random scaling with the scaling ratio between $[0.8, 1.25]$. Global random scaling can simulate agents' dynamics at different speeds, which can improve the model's generalization ability. Besides that, we apply randomly point dropout with probability 0.9 and points location perturbation under normal distribution with mean 0 and standard deviation 0.2.

Experiment details. We apply some similar standard preprocessing steps as previous works [19, 9]. First, we translate all the point data to be centered by the coordinate of agent data at $t = 0$. We use the orientation between the agent locations at $t = 0$ and its previous location as the positive x-axis. Then, we set the range starting from $[-48, -48]$ to $[48, 48]$ and filter the points outside the region. For voxelization, we set the grid size g to 0.2m to keep the balance between efficiency and performance. The intervals for Multi-interval Learning are predefined as $[2, 4, 6, 8, 16]$ that is enough for capturing sequential information. TPCN is trained for 36 epochs using a batch size of 32 with Adam [17] optimizer with an initial learning rate of 0.001. Besides that, the learning rate decays at every 10 epochs in a ratio of 0.1.

4.2. Ablation study

Component study. We conduct an ablation study on the Argoverse validation set to evaluate and analyze the contributions of our proposed components to the final performance. We take the spatial module without dual representations as our baseline. And then, we add other components gradually, as shown in Tab. 2. According to the results, we can draw some conclusions. First, spatial feature extraction based on point cloud learning that takes the unordered set of agents and map points is proven to be effective. With the only spatial module, our model has achieved a strong baseline compared with state-of-the-art methods [4, 10] that have provided results on the validation set. Second, spatial and temporal modules are both indispensable parts of our model. Spatial module models the geometric information, neighborhood relationship, and interaction among points. The temporal module handles the time-series features and focuses more on single instance feature learning. With both

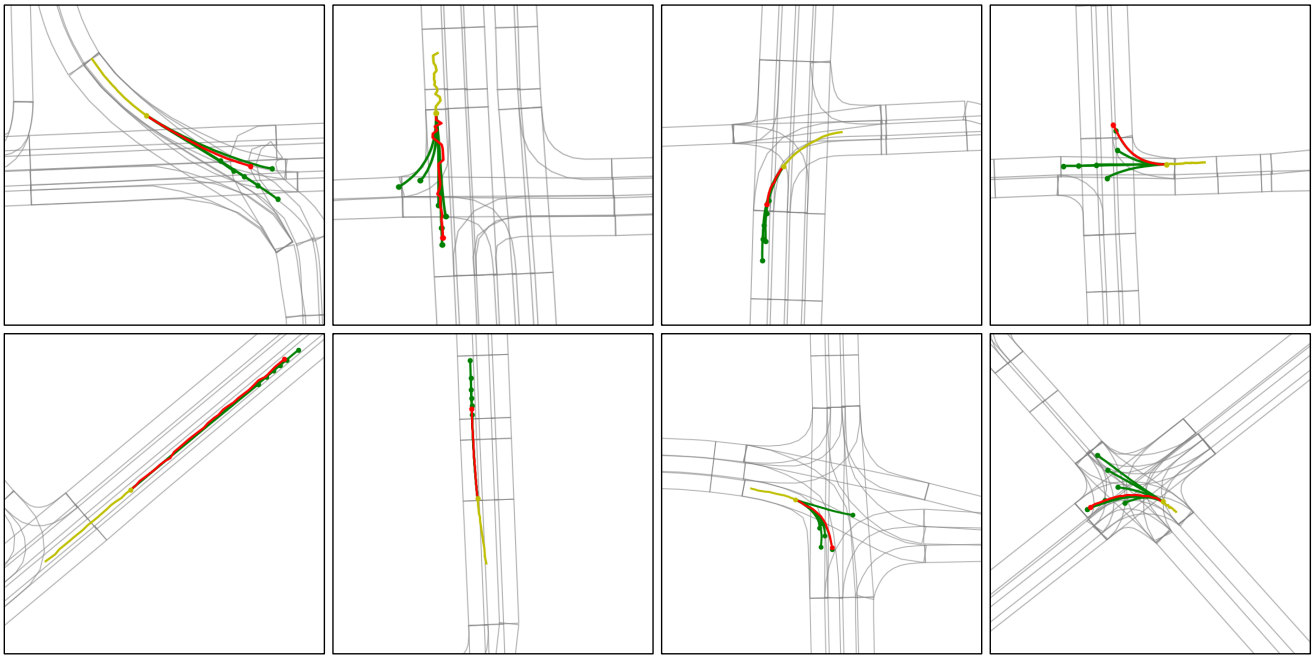

Figure 5. The motion forecasting results on the Argoverse validation set. The past trajectory of the target agent is in yellow, predicted trajectory in green and ground truth in red, respectively. The figures demonstrate the effectiveness of our TPCN on scenarios including left-turn, right-turn, lane change, and so on.

Spatial		Temporal		Aug.	minADE_1	minFDE_1	MR_1	minADE_6	minFDE_6	MR_6
Point	Voxel	MIL	Ins-Pool							
✓					1.75	4.00	0.66	1.01	1.88	0.28
	✓				1.63	3.65	0.62	0.94	1.70	0.23
✓	✓				1.50	3.31	0.57	0.85	1.42	0.16
✓	✓	✓			1.38	3.02	0.52	0.76	1.19	0.13
✓	✓	✓	✓		1.36	2.98	0.51	0.74	1.18	0.12
✓	✓	✓	✓	✓	**1.34**	**2.95**	**0.50**	**0.73**	**1.15**	**0.11**

Table 2. Ablation study of each component on the Argoverse validation set. Point and Voxel represent pointwise feature learning and voxelwise feature learning, respectively. The temporal module includes Multi-interval Learning (MIL) and Instance Pooling (Ins-Pool). "Aug." refers to data augmentation.

modules on, features propagate between spatial and temporal dimensions, and thus we achieve the best performance on the validation set. Third, point and voxel features are important feature compensation between each other due to the observation that when we apply dual-representation learning, the performance greatly outperforms single representation. Multi-interval learning plays a crucial role in temporal learning to capture sequential information, with about 10% improvement on displacement metrics. Furthermore, Instance Pooling retrieves the global instance features that address the long-range dependency problem, which leads to better performance.

Displacement prediction. We also evaluate the effectiveness of the proposed displacement prediction compared with typical classification. The experiment is based on the model with both spatial and temporal modules. Tab. 4 shows the displacement prediction with regression loss performs better than classification with cross-entropy loss in the aspect of trajectory selection. Displacement prediction gets rid of hard or manual assignment and converts the classification problem into a regression problem, which helps to converge to better results. It is worth noting that this change will not affect results for $K = 6$, demonstrating that trajectory regression and selection are independent tasks.

Data composition. Since there are mainly three types of data (agent, non-agent vehicles, map) in the Argoverse dataset, we conduct experiments to see whether our TPCN can extract the corresponding features, including map topology relationship and locality. During this experiment, we train our model by removing some specific kinds of data.

Models	minADE_1	minFDE_1	MR_1	minADE_6	minFDE_6	MR_6
Argoverse Baseline [5]	2.96	6.81	0.81	2.34	5.44	0.69
Argoverse Baseline (NN) [5]	3.45	7.88	0.87	1.71	3.29	0.54
Jean $(1st)$ [5, 23]	1.74	4.24	0.68	0.98	1.42	**0.13**
uulm-mrm $(2nd)$ [5]	1.90	4.19	0.63	0.94	1.55	0.22
LaneConv [19]	1.71	3.78	0.59	**0.87**	**1.36**	0.16
TNT [38]	1.77	3.91	0.59	0.94	1.54	**0.13**
Ours	**1.66**	**3.69**	**0.588**	**0.87**	1.38	0.158

Table 3. The results of our method and top performing approaches on the Argoverse test set.

Metrics	Loss	
	Classification	Displacement
minADE_1	1.44	**1.34**
minFDE_1	3.14	**2.95**
MR_1	0.54	**0.50**
minADE_6	0.74	**0.73**
minFDE_6	**1.14**	1.15
MR_6	**0.11**	**0.11**

Table 4. Ablation study on loss designs. Classification refers to predict scores and use cross-entropy loss to optimize. Displacement is the displacement prediction with regression loss.

Metrics	Data Composition			
	none	agents	map	agents + map
minADE_1	2.53	1.42	1.40	**1.34**
minFDE_1	3.94	3.08	3.04	**2.95**
MR_1	0.81	0.55	0.54	**0.50**
minADE_6	1.77	0.82	0.79	**0.73**
minFDE_6	3.54	1.32	1.29	**1.15**
MR_6	0.65	0.15	0.138	**0.11**

Table 5. Ablation study of data composition on the Argoverse validation set. Here, agents refer to other agents, and maps refer to map points data.

As shown in Tab. 5, we see that our TPCN can model the internal relationship among different types of input data. Map information brings useful topology of the road networks and semantic guidance since most of the driving behavior is based on lane keep and lane changes. Meanwhile, non-agents vehicles provide interaction under some decisions (i.e., nudge, overtake). Lacking any one of the data will lead to a significant performance drop for our TPCN.

Impact of data augmentation. We conduct an experiment to verify our data augmentation strategy in the prediction task. As shown in Tab. 2, the data augmentation improves TPCN in all the metrics, especially for minFDE.

4.3. Evaluation

Quantitative results. We compare our model with other methods that achieve the state-of-the-art in Argoverse motion forecasting leaderboard. As shown in Tab. 3, our TPCN improves the metrics for $K = 1$ by a large margin and outperforms the existing approaches in minADE_1, minFDE_1 and MR_1 without any complex postprocessing. Moreover, our TPCN is the first method which achieves minADE_1 less than 1.7m, minFDE_1 less than 3.7m and MR_1 less than 0.59. In contrast to existing methods [5] that ignore the temporal information or just use 1D CNN or LSTM to encode agents' temporal features, we mutually propagate the spatial and temporal features in order to maintain both locality and temporality. Furthermore, our proposed spatial module can effectively capture better map information compared with LaneConv [19]. Finally, before the CVPR submission deadline (16/11/2020), we ranked $1st$, $1st$, $1st$, $2nd$, $3rd$, $5th$ on the leaderboard according to the metrics, respectively.

Quantitative results. We present some multi-modal prediction trajectories on several hard cases shown in Fig. 5. Despite the noise of the input trajectory, TPCN can generate feasible, reasonable, and smooth trajectories with map constraints. For the multi-modality under junction scenarios, TPCN is able to capture the topology of the road network and give possible trajectories along with the lane's successors or neighborhood.

5. Conclusion

In this paper, we propose our TPCN that serves as a novel and flexible architecture for prediction learning. TPCN models the motion forecasting problem as joint temporal point cloud learning, consisting of both spatial and temporal modules. In the spatial module, TPCN takes the merit of dual-representation learning in point clouds to maintain better locality and geometric relationships. The temporal module utilizes the proposed Multi-interval Learning and Instance Pooling to capture more fine-grained sequential information. Both modules are learned and propagated mutually to obtain better context information for prediction learning. Experiments on the Argoverse motion forecasting benchmark show the effectiveness of our TPCN.

References

[1] S Bai, JZ Kolter, and V Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arxiv 2018. *arXiv preprint arXiv:1803.01271*, 1803. 2

[2] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018. 1, 2

[3] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018. 2

[4] 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, 5, 6

[5] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 6, 8

[6] 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, 5

[7] Nemanja Djuric, Vladan Radosavljevic, Henggang Cui, Thi Nguyen, Fang-Chieh Chou, Tsung-Han Lin, and Jeff Schneider. Short-term motion prediction of traffic actors for autonomous driving using deep convolutional networks. *arXiv preprint arXiv:1808.05819*, 2018. 1

[8] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015. 2

[9] Liangji Fang, Qinhong Jiang, Jianping Shi, and Bolei Zhou. Tpnet: Trajectory proposal network for motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6797–6806, 2020. 6

[10] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020. 1, 2, 5, 6

[11] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018. 4

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2, 4

[13] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015. 2

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4

[15] Adam Houenou, Philippe Bonnifait, Véronique Cherfaoui, and Wen Yao. Vehicle trajectory prediction based on motion model and maneuver recognition. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 4363–4369. IEEE, 2013. 1

[16] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018. 5

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[18] Colin Lea, Michael D Flynn, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks for action segmentation and detection. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 156–165, 2017. 2

[19] Ming Liang, Bin Yang, Rui Hu, Yun Chen, Renjie Liao, Song Feng, and Raquel Urtasun. Learning lane graph representations for motion forecasting. *arXiv preprint arXiv:2007.13732*, 2020. 1, 2, 4, 5, 6, 8

[20] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, pages 965–975, 2019. 4

[21] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting With a Single Convolutional Net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 2

[22] Karttikeya Mangalam, Harshayu Girase, Shreyas Agarwal, Kuan-Hui Lee, Ehsan Adeli, Jitendra Malik, and Adrien Gaidon. It is not the journey but the destination: Endpoint conditioned trajectory prediction. *arXiv preprint arXiv:2004.02025*, 2020. 2

[23] Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil. Multi-head attention for multi-modal joint vehicle motion forecasting. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9638–9644. IEEE, 2020. 8

[24] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020. 2

[25] 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. 1, 2

[26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2

[27] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Advances in Neural Information Processing Systems 30*, volume 30, pages 5099–5108. Curran Associates, Inc., 2017. 2, 3

[28] Jens Schulz, Constantin Hubmann, Julian Löchner, and Darius Burschka. Interaction-aware probabilistic behavior prediction in urban environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3999–4006. IEEE, 2018. 1

[29] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020. 3

[30] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013. 2

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[32] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. *arXiv preprint arXiv:2007.16100*, 2020. 3

[33] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019. 2, 3

[34] Guotao Xie, Hongbo Gao, Lijun Qian, Bin Huang, Keqiang Li, and Jianqiang Wang. Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models. *IEEE Transactions on Industrial Electronics*, 65(7):5999–6008, 2017. 1

[35] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 4

[36] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1631–1640, 2020. 2, 3

[37] 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 Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 2

[38] 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. 2, 8

[39] Julius Ziegler, Philipp Bender, Markus Schreiber, Henning Lategahn, Tobias Strauss, Christoph Stiller, Thao Dang, Uwe Franke, Nils Appenrodt, Christoph G Keller, et al. Making bertha drive—an autonomous journey on a historic route. *IEEE Intelligent transportation systems magazine*, 6(2):8–20, 2014. 1