STINet: Spatio-Temporal-Interactive Network for Pedestrian Detection and Trajectory Prediction

Zhishuai Zhang[1,2*] Jiyang Gao[1] Junhua Mao[1] Yukai Liu[1] Dragomir Anguelov[1] Congcong Li[1]
[1]Waymo LLC [2] Johns Hopkins University
zzhang99@jhu.edu, {jiyanggao, junhuamao, liuyukai, dragomir, congcongli}@waymo.com

Abstract

Detecting pedestrians and predicting future trajectories for them are critical tasks for numerous applications, such as autonomous driving. Previous methods either treat the detection and prediction as separate tasks or simply add a trajectory regression head on top of a detector. In this work, we present a novel end-to-end two-stage network: Spatio-Temporal-Interactive Network (STINet). In addition to 3D geometry modeling of pedestrians, we model the temporal information for each of the pedestrians. To do so, our method predicts both current and past locations in the first stage, so that each pedestrian can be linked across frames and the comprehensive spatio-temporal information can be captured in the second stage. Also, we model the interaction among objects with an interaction graph, to gather the information among the neighboring objects. Comprehensive experiments on the Lyft Dataset and the recently released large-scale Waymo Open Dataset for both object detection and future trajectory prediction validate the effectiveness of the proposed method. For the Waymo Open Dataset, we achieve a bird-eyes-view (BEV) detection AP of 80.73 and trajectory prediction average displacement error (ADE) of 33.67cm for pedestrians, which establish the state-of-the-art for both tasks.

1. Introduction

To drive safely and smoothly, self-driving cars (SDC) not only need to detect where the objects are currently (*i.e.* object detection), but also need to predict where they will go in the future (*i.e.* trajectory prediction). Among the objects, pedestrian is an important and difficult type. The difficulty comes from the complicated properties of pedestrian appearance and behavior, *e.g.* deformable shape and interpersonal relations [7]. In this paper, we tackle the problem of joint pedestrian detection and trajectory prediction from a sequence of point clouds, as illustrated in Figure 1.

∗ Work done during an internship at Waymo.

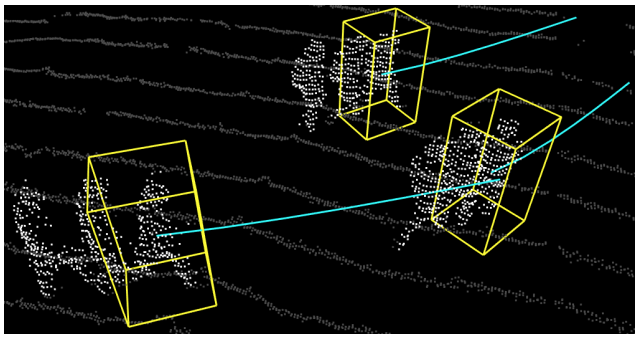

Figure 1. Given a sequence of current and past point clouds, our task is to detect pedestrians in the current frame, and predict the future trajectory of them. In this figure, white points are input point cloud sequence (stacked for visualization), yellow boxes are detected objects, and the cyan lines are predicted future trajectory.

Traditionally, this problem is tackled by dividing the perception pipeline into multiple modules: object detection [6, 13, 15, 16, 20, 21, 29, 30], tracking [18] and trajectory prediction [2, 7, 9]; latter modules take the outputs from the former modules. Although such strategy makes each sub-module easy to design and implement, it sacrifices the potential advantage of joint optimization. Latter modules can lose critical information bottle-necked by the interfaces between sub-modules, *e.g.* a pedestrian's future trajectory depends on many useful geometry features from the raw sensor data, which may be abstracted away in the detection/tracking stage. To this end, researchers recently have proposed several end-to-end neural networks to detect objects and predict trajectories simultaneously. FaF [17] and IntentNet [4] are two of the representative methods, which are designed based on single stage detectors (SSD) [16]; in addition to original anchor classification and regression of SSD, they also regress a future trajectory for each anchor.

We observed that there are two major issues that are critical for joint detection and trajectory prediction, but are not addressed by previous end-to-end methods: 1) Temporal modeling on object level: existence and future trajectory of an object are embedded in both current and past

frames. Current methods simply reuse single-stage detector and fuse the temporal information in the backbone CNN in an object-agnostic manner either via feature concatenation or 3D CNN [4, 17]. Such coarse level fusion can loss fine-grained temporal information for each object, which is critical for both tasks. 2) Interaction modeling among objects: the future trajectory of an object could be influenced by the other objects. *E.g*., a pedestrian walking inside a group may tend to follow others. Existing methods [4, 17] do not explicitly model interactions among objects.

To address the aforementioned issues, we propose an end-to-end Spatio-Temporal-Interactive network (STINet) to model pedestrians' temporal and interactive information jointly. The proposed network takes a sequence of point clouds as input, detects current location and predicts future trajectory for pedestrians. Specifically, there are three subcomponents in STINet : backbone network, proposal generation network, and proposal prediction network. In the backbone net, we adopted a similar structure as PointPillars [13], and applied it on each frame of the point cloud, the output feature maps from multi-frames are then combined. The proposal generation network takes feature maps from the backbone net and generates potential pedestrian instances with both their current and past locations (*i.e*. temporal proposals); such temporal proposals allow us to link the same object across different frames. In the third module (*i.e*. prediction network), we use the temporal proposals to explicitly gather the geometry appearance and temporal dynamics for each object. To reason the interaction among pedestrians, we build a graph layer to gather the information from surrounding pedestrians. After extracting the above spatial-temporal-interactive feature for each proposal, the detection and prediction head uses the feature to regress current detection bounding box and future trajectory.

Comprehensive experiments are conducted on Waymo Open Dataset [1] and Lyft Dataset [12] to demonstrate the effectiveness of the STINet. Specifically, it achieves an average precision of 80.73 for bird-eyes-view pedestrian detection, and an average displacement error of 33.67 cm for trajectory prediction on Waymo Open Dataset. It achieves real-time inference speeds and takes only 74.6 ms for inference on a range of 100m by 100m.

The main contributions of our work come in four folds:

- We build an end-to-end network tailored to model pedestrian past, current and future simultaneously.
- We propose to generate temporal proposals with both current and past boxes. This enables learning a comprehensive spatio-temporal representation for pedestrians with their geometry, dynamic movement and history path in an end-to-end manner without explicitly associating object across frames.
- We propose to build a graph among pedestrians to reason the interactions to further improve trajectory prediction quality.
- We establish the state-of-the-art performance for both detection and trajectory prediction on the Lyft Dataset and the recent large-scale challenging Waymo Open Dataset.

2. Related work

2.1. Object detection

Object detection is a fundamental task in computer vision and autonomous driving. Recent approaches can be divided into two folds: single-stage detection [15, 16, 20] and two-stage detection [6, 21]. Single-stage detectors do classification and regression directly on backbone features, while two-stage detectors generate proposals based on backbone features, and extract proposal features for second-stage classification and regression. Single-stage detectors have simpler structure and faster speed, however, they lose the possibility to flexibly deal with complex objects behaviors, *e.g*., explicitly capturing pedestrians moving across frames with different speeds and history paths. In this work, we follow the two-stage detection framework and predict object boxes for both current and past frames as proposals, which are further processed to extract their geometry and movement features.

2.2. Temporal proposals

Temporal proposals have been shown beneficial in action localization in [10, 11]. They showed associating temporal proposals from different video clips can help to leverage the temporal continuity of video frames. [25] proposed to link temporal proposals throughout the video to improve video object detection. In our work, we also exploit temporal proposals and step further to investigate and propose how to build comprehensive spatio-temporal representations of proposals to improve future trajectory prediction. This is a hard task since there are no inputs available for the future. Also we investigate to learn interactions between proposals via a graph. We show that these spatio-temporal features can effectively model objects' dynamics and provide accurate detection and prediction of their future trajectory.

2.3. Relational reasoning

An agent's behavior could be influenced by other agents and it is naturally connected to relational reasoning [3, 23]. Graph neural networks have shown its strong capability in relational modeling in recent years. Wang *et al*. formulated the video as a space-time graph, show the effectiveness on the video classification task [26]. Sun *et al*. designed a relational recurrent network for action detection and anticipation [24]. Yang *et al*. proposed to build an object relationship graph for the task of scene graph generation [28].

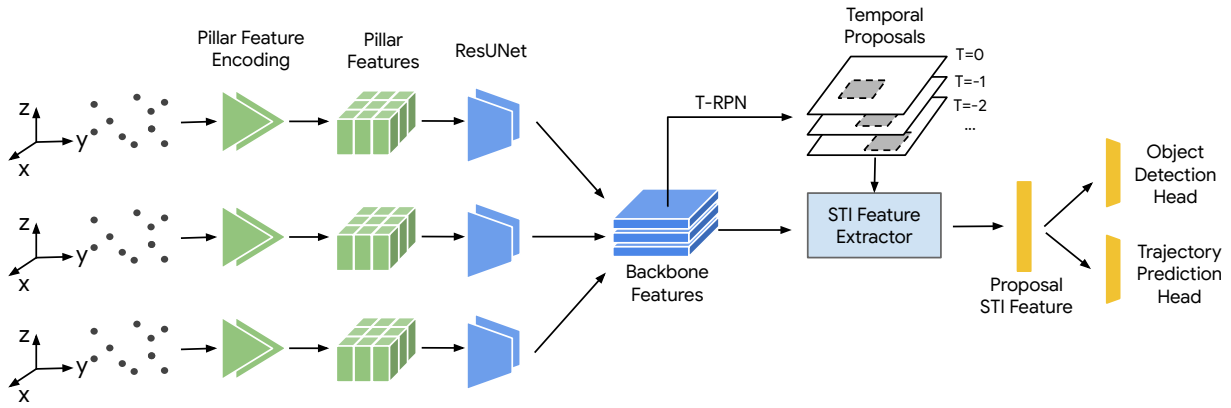

Figure 2. The overview of the proposed method. It takes a sequence of point clouds as input, detects pedestrians and predicts their future trajectories simultaneously. The point clouds are processed by Pillar Feature Encoding [13, 30] to generate Pillar Features. Then each Pillar Feature is fed into a backbone ResUNet [22] to get backbone features. A Temporal Region Proposal Network (T-RPN) takes backbone features and generated temporal proposal with past and current boxes for each object. Spatio-Temporal-Interactive (STI) Feature Extractor learns features for each temporal proposal which are used for final detection and trajectory prediction.

2.4. Trajectory prediction

Predicting the future trajectory of objects is an important task, especially for autonomous driving. Previous research has been conducted based on perception objects as inputs [2, 5, 7, 9, 14]. Recently FaF [17] and IntentNet [4] focused on end-to-end trajectory prediction from raw point clouds as input. However, they simply re-used single-stage detection framework and added new regression heads on it. In our work, we exploit temporal region proposal network and explicitly model Spatio-Temporal-Interaction (STI) representations of pedestrians, and our experiments show that the proposed STI modeling is superior on both detection and trajectory prediction for pedestrians.

3. Proposed method

In this section, we discuss our proposed network in details. The overview of our proposed method is shown in Figure 2, which can be divided into three steps. For each of these steps, we discuss in the following subsections.

3.1. Backbone network

The backbone of our network is illustrated in Figure 3. The input is a sequence of point clouds with t' frames noted as $[\mathrm{PC}_{-(t'-1)}, \mathrm{PC}_{-(t'-2)}, \cdots, \mathrm{PC}_0]$, which corresponds to the lidar sensor input from the past $t'-1$ frames as well as the current frame. All point clouds are calibrated to SDC's pose at the current frame so that the ego-motion is discarded. To build rich pillar features while keeping a feasible memory usage, we generate t pillar features from the t' input frames. Consecutive t'/t point clouds $\mathrm{PC}_{-(j+1)t'/t+1}, \cdots, \mathrm{PC}_{-jt'/t}$ are processed with Voxelization [13, 30] and then concatenated to generate a pseudo image I_j (*i.e.* Pillar Features) with shape $H \times W \times C_{in}$.

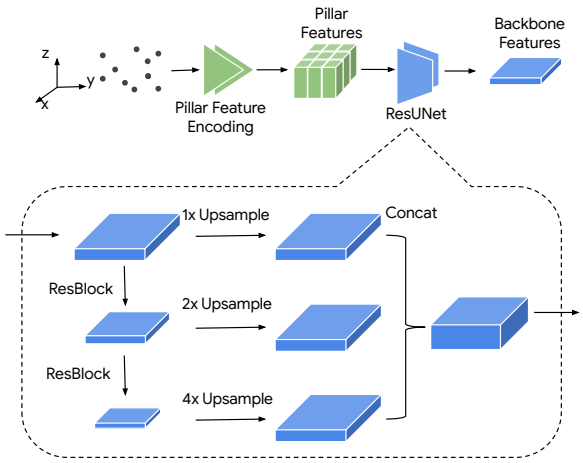

Figure 3. Backbone of proposed network. Upper: overview of the backbone. The input point cloud sequence is fed to Voxelization and Point net to generate pseudo images, which are then processed by ResNet U-Net to generate final backbone feature sequence. Lower: detailed design of ResNet U-Net.

Thus the output of Pillar Feature Encoding is a sequence of t Pillar Features $[I_{-(t-1)}, I_{-(t-2)}, \cdots, I_0]$.

Next we adopt a similar backbone CNN network proposed as in [22], as shown in the lower part of Figure 3. Each of the Pillar Features I_j is first processed by three ResNet-style blocks to generate intermediate features with shape $\mathbb{R}^{H \times W \times C_0}$,$\mathbb{R}^{\frac{1}{2}H \times \frac{1}{2}W \times C_1}$ and $\mathbb{R}^{\frac{1}{4}H \times \frac{1}{4}W \times C_2}$. Then we use deconvolution layers to upsample them to the same spatial shape with I_j. The concatenation of the upsampled features serve as the backbone feature of I_j, noted as B_j.

3.2. Temporal proposal generation

In order to explicitly model objects' current and past knowledge, we propose a temporal region proposal network (T-RPN) to generate object proposals with both current and past boxes. T-RPN takes the backbone feature sequence $[B_{-(t-1)}, B_{-(t-2)}, \cdots, B_0]$ as the input, concatenates them in the channel dimension and applies a 1×1 convolution to generate a temporal-aware feature map. Classification, current frame regression and past frames regression are generated by applying 1×1 convolutional layers over the temporal-aware feature map, to classify and regress the pre-defined anchors.

The temporal region proposal network is supervised by ground-truth objects' current and past locations. For each anchor $\mathbf{a} = (x^a, y^a, w^a, l^a, h^a)$ (x, y, w, l, h correspond to x coordinate of box center, y coordinate of box center, width of box, length of box and heading of box respectively), it is assigned to a ground-truth object with largest IoU of the current frame box $\mathbf{gt} = (x_0^{gt}, y_0^{gt}, w^{gt}, l^{gt}, h_0^{gt})$. Similar to SECOND [27], we compute the regression target in order to learn the difference between the pre-defined anchors and the corresponding ground-truth boxes. For the current frame, we generate a 5-d regression target $\mathbf{d}_0^a = (dx_0^a, dy_0^a, dw^a, dl^a, dh_0^a)$:

$$dx_0^a = (x_0^{gt} - x^a)/\sqrt{(x^a)^2 + (y^a)^2} \tag{1}$$

$$dy_0^a = (y_0^{gt} - y^a)/\sqrt{(x^a)^2 + (y^a)^2} \tag{2}$$

$$dw^a = \log \frac{w^{gt}}{w^a} \tag{3}$$

$$dl^a = \log \frac{l^{gt}}{l^a} \tag{4}$$

$$dh_0^a = \sin \frac{h_0^{gt} - h^a}{2} \tag{5}$$

With similar equations, we also compute $t-1$ past regression targets for anchor $\mathbf{a}$ against the same ground-truth object: $\mathbf{d}_j^a = (dx_j^a, dy_j^a, dh_j^a)$ for $j \in \{-1, -2, \cdots, -(t-1)\}$. Width and length are not considered for the past regression since we assume the object size does not change across different frames. For each anchor $\mathbf{a}$, the classification target s^a is assigned as 1 if the assigned ground-truth object has an IoU greater than th^+ at the current frame. If the IoU is smaller than th^-, classification target is assigned as 0. Otherwise the classification target is -1 and the anchor is ignored for computing loss.

For each anchor $\mathbf{a}$, T-RPN predicts a classification score $\hat{s}^a$, a current regression vector $\hat{\mathbf{d}}_0^a = (\hat{dx}_0^a, \hat{dy}_0^a, \hat{dw}^a, \hat{dl}^a, \hat{dh}_0^a)$ and $t-1$ past regression vectors $\hat{\mathbf{d}}_j^a = (\hat{dx}_j^a, \hat{dy}_j^a, \hat{dh}_j^a)$ from the aforementioned 1×1 convolutional layers. The objective of T-RPN is the weighted sum of classification loss, current frame regression loss and past frames regression loss as defined in the equations below, where $\mathbb{1}(x)$ is the indicator function and returns 1 if x is true otherwise 0.

$$\mathcal{L}_{\text{T-RPN}} = \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{cur\_reg}}\mathcal{L}_{\text{cur\_reg}} + \lambda_{\text{past\_reg}}\mathcal{L}_{\text{past\_reg}} \tag{6}$$

$$\mathcal{L}_{\text{cls}} = \frac{\sum_{\mathbf{a}} \text{CrossEntropy}(s^a, \hat{s}^a)\mathbb{1}(s^a \geq 0)}{\sum_{\mathbf{a}} \mathbb{1}(s^a \geq 0)} \tag{7}$$

$$\mathcal{L}_{\text{cur\_reg}} = \frac{\sum_{\mathbf{a}} \text{SmoothL1}(\mathbf{d}_0^a, \hat{\mathbf{d}}_0^a)\mathbb{1}(s^a \geq 1)}{\sum_{\mathbf{a}} \mathbb{1}(s^a \geq 1)} \tag{8}$$

$$\mathcal{L}_{\text{past\_reg}} = \sum_{j=1}^{t-1} \frac{\sum_{\mathbf{a}} \text{SmoothL1}(\mathbf{d}_{-j}^a, \hat{\mathbf{d}}_{-j}^a)\mathbb{1}(s^a \geq 1)}{\sum_{\mathbf{a}} \mathbb{1}(s^a \geq 1)} \tag{9}$$

For proposal generation, classification scores and regression vectors are applied on pre-defined anchors to generate temporal proposals, by reversing Equations 1-5. Thus each temporal proposal has a confidence score as well as the regressed boxes for the current and past frames. After that, non-maximum suppression is applied on the current frame boxes of temporal proposals to remove redundancy.

3.3. Proposal prediction

3.3.1 Spatio-temporal-interactive feature extraction

Given backbone features $[B_{-(t-1)}, \cdots, B_0]$ and temporal proposals, spatio-temporal-interactive features are learned for each temporal proposal to capture the comprehensive information for detection and trajectory prediction. Different ways for modeling objects are combined to achieve this.

Local geometry feature: To extract object geometry knowledge, we use the proposal boxes at j-th frame (*i.e.* x_j, y_j, w, l, and h_j) to crop features from B_j, as shown in the lower left part of Figure 4. This is an extension of traditional proposal feature cropping used in Faster-RCNN [21], to gather position-discarded local geometry features from each frame. To simplify the implementation on TPU, we rotate the 5-DoF box (x_j, y_j, w, l, h_j) to the closest standing box $(x_{min,j}, y_{min,j}, x_{max,j}, y_{max,j})$ for ROIAlign [8].

Local dynamic feature: As illustrated in the lower middle part of Figure 4, we use a meta box (drawn in yellow) which covers the whole movement of the pedestrian to crop features for all B_j's. The meta box is the smallest box which contains all current and history proposal boxes. Formally, after transferring all rotated proposal boxes (x_j, y_j, w, l, h_j) to the closest standing boxes $(x_{min,j}, y_{min,j}, x_{max,j}, y_{max,j})$, the meta box is computed with the following equations:

$$x_{min} = \min_j(x_{min,j}); y_{min} = \min_j(y_{min,j})$$

$$x_{max} = \max_j(x_{max,j}); y_{max} = \max_j(y_{max,j})$$

This feature captures the direction, curvature and speed of the object, which are useful for future trajectory prediction.

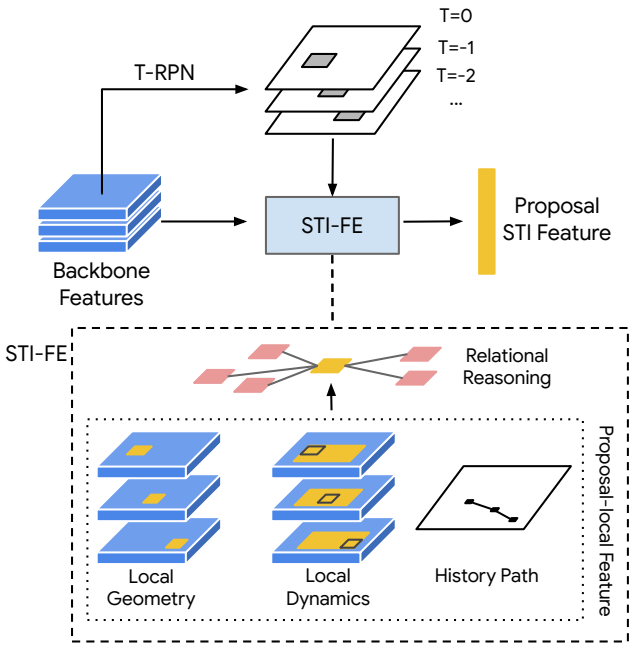

Figure 4. Spatial-Temporal-Interactive Feature Extractor (STI-FE): Local geometry, local dynamic and history path features are extracted given a temporal proposal. For local geometry and local dynamics features, the yellow areas are used for feature extraction. Relational reasoning is performed across proposals' local features to generate interactive features.

History path feature: In order to directly encode objects' past movement, we exploit the location displacement over different frames as the history path feature. To be specific, given a temporal proposal with x_j, y_j as the box centers, the history path feature is MLP($[x_0 - x_{-1}, y_0 - y_{-1}, x_0 - x_{-2}, y_0 - y_{-2}, \cdots, x_0 - x_{-(t-1)}, y_0 - y_{-(t-1)}]$).

To aggregate spatial and temporal knowledge for each proposal, the concatenation of local geometry feature and the local dynamic feature is fed into a ResNet block followed by a global average pooling. The pooled feature is then concatenated with the history path feature, and serves as the proposal-local feature, noted as f_i for the i-th temporal proposal.

As discussed before, the future trajectory of a pedestrian could be influenced by the surrounding pedestrians' behaviors. In order to model such interactions among pedestrians, we design an interaction layer which uses a graph to propagate information among objects, as shown in the middle part of Figure 4. Specifically, we represent each temporal proposal as a graph node i; the embedding of node i is noted as f_i, which is the corresponding proposal-local feature. The edge v_{ij} represents the interaction score between node i and node j. v_{ij} is learned from f_i and f_j, which can be represented as below.

$$v_{ij} = \alpha([\phi_1(f_i); \phi_2(f_j)])$$

where α and ϕ's can be any learnable functions. In our implementation, we use fully-connected layer for α and ϕ's.

Given the interaction scores among all pairs of nodes, we can gather the information for each node from the neighboring nodes. Specifically, the interaction embedding g_i gathered for node i is calculated as follows:

$$g_i = \sum_j \frac{\exp\{v_{ij}\}}{V_i} \gamma([f_i; f_j])$$

where $V_i = \sum_j \exp\{v_{ij}\}$ is the normalization constant, and γ is a mapping function (a fully-connected layer is adopted in our implementation).

3.3.2 Proposal classification and regression

Given proposal-local features f_i for each temporal proposals, two fully-connected layers are applied to do classification and regression respectively for the current frame. To be aligned with our intuitions, the proposal-local feature f_i combined with the interaction feature g_i is used to predict future frame boxes, by one fully-connected layer with $3t$ output channels where t is the number of future frames to predict and 3 stands for x coordinate, y coordinate and heading respectively. During the training, temporal proposals are assigned classification and regression targets with the same strategy discussed in Subsection 3.2 and the objective is the weighted sum of classification loss, current frame regression loss and future frames regression loss similar to Equations 6-9. During inference, each proposal is predicted with a classification score and current/future boxes. Non-maximum suppression is applied on them based on the IoU between their current boxes, to remove redundancy.

4. Experiment

4.1. Experiment settings

Dataset: We conduct experiments on the Waymo Open Dataset (WOD) [1] and the Lyft Dataset (Lyft) [12]. WOD contains lidar data from 5 sensors and labels for 1000 segments. Each segment contains roughly 200 frames and has a length of 20 seconds. Train and validation subsets have 798 and 202 segments respectively. To model the history and predict the future, we take 1 second history frames and 3 second future frames for each example and extract examples from the center 16 seconds (1s∼17s) from each segment. Thus 126,437 train examples and 31,998 validation examples are extracted, and each of them contains history frames of 1 second and future frames of 3 seconds. We sample 6 frames including 5 history frames and the current frame, with $t_{\text{input}} = \{-1.0, -0.8, -0.6, -0.4, -0.2, 0\}$, and the point clouds from those frames are fed into the network as inputs. In order to build richer voxel features while saving computation and memory, every two frames are combined

Model	MF	TS	DE@1 ↓	DE@2 ↓	DE@3 ↓	ADE ↓	HR@1 ↑	HR@2 ↑	HR@3 ↑
IntentNet	✓		21.17$_{\pm 0.02}$	39.74$_{\pm 0.07}$	61.60$_{\pm 0.12}$	36.04$_{\pm 0.12}$	93.18$_{\pm 0.03}$	76.50$_{\pm 0.08}$	61.60$_{\pm 0.12}$
MF-FRCNN	✓	✓	20.87$_{\pm 0.08}$	39.23$_{\pm 0.14}$	60.59$_{\pm 0.22}$	35.57$_{\pm 0.13}$	93.45$_{\pm 0.05}$	76.69$_{\pm 0.18}$	61.57$_{\pm 0.21}$
STINet	✓	✓	**19.63**$_{\pm 0.03}$	**37.07**$_{\pm 0.08}$	**57.60**$_{\pm 0.14}$	**33.67**$_{\pm 0.07}$	**94.36**$_{\pm 0.05}$	**78.91**$_{\pm 0.06}$	**64.43**$_{\pm 0.15}$

Table 1. Trajectory prediction performance for different models on WOD. MF indicates whether the corresponding model takes multiple frames as input. TS indicates whether the model has a two-stage framework. ↑ and ↓ indicate the higher/lower numbers are better for the corresponding metric. DE and ADE are in centimeters. For models implemented by us, we train and evaluate the model for five times and compute the average and standard deviation shown around ± in the table.

Model	MF	TS	BEV AP ↑
PointPillar [29]			68.57
MVF [29]			74.38
StarNet [19]			72.50
IntentNet [4][1]	✓		79.43$_{\pm 0.10}$
MF-FRCNN	✓	✓	79.69$_{\pm 0.19}$
STINet	✓	✓	**80.73**$_{\pm 0.26}$

Table 2. Detection performance for different methods on WOD. MF indicates whether the corresponding model takes multiple frames as input. TS indicates whether the model has a two-stage framework. BEV AP is computed with an IoU threshold of 0.5. ↑ indicates the higher numbers are better for the corresponding metric.

by concatenating the voxelization output features thus we have three pillar features as discussed in Subsection 3.1. For the future prediction, we predict trajectory for 6 future frames with $t_{\text{future}} = \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$. The range is 150m by 150m around the self-driving car, and we use a pillar size of 31.25cm by 31.25cm to generate pillar features of shape 480×480. Lyft contains lidar data from 1 sensor and labels for only 180 segments, with 140 and 40 segments for train and validation respectively. With the same settings, 14,840 and 4,240 examples are extracted for train and validation. Each example has 1-second history and 3-second future. We have $t_{\text{future}} = \{0.6, 1.2, 1.8, 2.4, 3.0\}$ for Lyft due to its 5Hz sampling rate.

Evaluation metric: The evaluation metric for detection is BEV AP (Bird-Eyes-View Average Precision) with the IoU threshold set to 0.5. Objects with fewer than 5 points are considered hard and are excluded during evaluation. For trajectory prediction, we employ the metrics used in [4, 9]. For $t \in t_{\text{future}}$, we compute the DE@t (Displacement Error) and the HR@t (Hit Rate) with a displacement error threshold of 0.5m. We also compute the ADE (Average Displacement Error) which equals to $\frac{1}{|t_{\text{future}}|} \sum_{t \in t_{\text{future}}} \text{DE@}t$.

Implementation: Our models are implemented in TensorFlow and we train the model with Adam optimizer on TPUv3 for 140k and 70k iterations for Waymo Open Dataset and Lyft Dataset respectively. The learning rate is 4×10^{-4} and batch size is 1 per TPU. We use 32 TPU cores together for the training, thus the effective batch size is 32.

[1] IntentNet without intent prediction head implemented by us.

We also implement IntentNet [4] and Faster-RCNN [21] in TensorFlow as the baselines, which are noted as "IntentNet" and "MF-FRCNN". Our implemented IntentNet (1) takes multiple frames as input and share the same backbone net as STINet; (2) removes the intent classification part, and only regresses a future trajectory. MF-FRCNN refers to a Faster-RCNN [21] model with several changes: (1) It uses the same backbone net as STINet, please refer to Section 3.1; (2) for each object proposal, in addition to the bounding box, we also regress future trajectories and headings. Note that the difference between proposals from MF-FRCNN and our method is that MF-FRCNN only predicts the current box of objects, while our method exploits a novel Temporal RPN which also generates the corresponding history boxes associated to each current box.

4.2. Results on Waymo Open Dataset

The main results on Waymo Open Dataset of pedestrian detection and trajectory prediction are summarized in Table 2 and Table 1. For detection we compare our proposed method (in the last row) with the current state-of-the-art detectors [19, 29] and our method surpasses the off-the-shelf baselines by a very large margin, improving the BEV AP from 74.38 to 80.73. To avoid the effects from multi-frame inputs and different implementation details, we also compare with our implementation of IntentNet and multi-frame Faster RCNN [21], which are noted as "IntentNet" and "MF-FRCNN" in Table 2. Our proposed method outperforms all baselines and it confirms the effectiveness of our T-RPN and the STI modeling of proposals.

In Table 1 we compare the trajectory prediction performance between our proposed method, IntentNet and MF-FRCNN. Our proposed method surpasses all competitors by a large margin, and the improvement is larger than the improvement on detection. It aligns with our intuition since T-RPN and STI modeling are designed to better model objects' movement and more useful to forecast their trajectory.

For a detailed comparison of STINet and MF-FRCNN, we evaluate the detection and trajectory prediction by breaking down the objects into five bins based on the future trajectory length in 3s. The five bins are 0∼2.5m, 2.5∼5m, 5∼7.5m, 7.5∼10m and 10m∼ ∞ respectively. We report BEV AP, ADE and the relative improvement in Table 3 and 4. The STINet is consistently better than MF-FRCNN

Model	0∼2.5	2.5∼5	5∼7.5	7.5∼10	10∼ ∞
MF-FRCNN	63.07	90.44	93.27	88.00	77.15
STINet	**64.23**	**91.15**	**94.46**	**88.97**	**80.50**
Δ%	1.8%	0.8%	1.3%	1.1%	4.3%

Table 3. Bird-eyes-view average precision (BEV-AP) breakdown comparison of MF-FRCNN and STINet on WOD. Objects are split into five bins base on the future trajectory length with a bin size of 2.5m. Last row is the relative improvement of STINet.

Model	0∼2.5	2.5∼5	5∼7.5	7.5∼10	10∼ ∞
MF-FRCNN	26.90	37.56	46.39	104.60	173.50
STINet	**26.73**	**35.42**	**41.18**	**89.74**	**137.17**
Δ%	0.6%	6.0%	11.2%	14.2%	20.9%

Table 4. Average displacement error (ADE, in centimeters) breakdown comparison of MF-FRCNN and STINet on WOD. Objects are split into five bins base on the future trajectory length with a bin size of 2.5m. Last row is the relative improvement of STINet.

Model	BEV AP ↑	DE@3 ↓	ADE ↓	HR@3 ↑
MF-FRCNN	33.90	82.61	51.11	49.74
STINet	**37.15**	**76.17**	**46.09**	**50.73**

Table 5. Detection and trajectory prediction performance on Lyft.

LG	LD	BEV AP ↑	DE@3 ↓	ADE ↓	HR@3 ↑
✓		80.38	64.15	37.67	58.46
	✓	79.69	59.71	34.96	62.22
✓	✓	**80.53**	**58.95**	**34.49**	**62.99**

Table 6. Ablation studies on local geometry and local dynamic features (noted as LG and LD in the table respectively). All entries are trained without History Path and Interactive features.

L+G	Path	DE@3 ↓	ADE ↓	HR@3 ↑
✓		58.95	34.49	62.99
✓	✓	**58.04**	**33.92**	**63.87**
†	✓	67.80	39.86	52.25

Table 7. Ablation studies on history path feature. † indicates the corresponding feature is used only for detection and ignored while prediction the trajectory.

for both tasks. For trajectory prediction on objects moving more than 5m, the relative improvements are significant and consistently more than 10%. It confirms that the proposed method can leverage the details of history information and provide much better trajectory predictions, especially for pedestrians with a larger movement.

4.3. Results on Lyft Dataset

The detection and trajectory prediction results on the Lyft Dataset are summarized in Table 5. The performances on both tasks are improved largely and the results confirm the effectiveness of proposed method a small-scale dataset.

4.4. Ablation studies

In this section we conduct ablation experiments to analyze the contribution of each component and compare our model with potential alternative methods on the Waymo Open Dataset. The results are summarized below. For clarity, we only show DE@3, ADE and HR@3 for trajectory prediction. The other metrics have the same tendency.

Breakdown	I	DE@3 ↓	ADE ↓	HR@3 ↑
All		58.04	33.92	63.87
	✓	**57.60**	**33.67**	**64.43**
Group		49.67	30.85	64.87
	✓	**48.89**	**30.40**	**65.55**

Table 8. Ablation studies on interaction features. 'I' indicates whether the proposal interaction modeling is adopted. "All" and "Group" correspond to evaluation on all pedestrians and pedestrians belonging to a group with at least 5 pedestrians respectively.

Effect of local geometry and local dynamic features: We conduct experiments to analyze the effect of local geometry and local dynamic features, summarized in Table 6. The local geometry feature is good at detection and the local dynamic feature is good at trajectory prediction. Geometry feature itself does not work well for trajectory prediction since it ignores dynamics for better detection. By combining both of the features, the benefits in detection and trajectory prediction can be obtained simultaneously.

Effect of history path: Although objects' geometry and movement are already represented by local geometry dynamic features, taking history path as an extra feature can give another performance gain by improving the DE@3 from 58.95 to 58.04 and the HR@3 from 62.99 to 63.87 (as shown in the first two row of Table 7). This suggests the history path, as the easiest and most direct representation of objects' movement, can still help based on the rich representations. However history path itself is far from enough to give accurate trajectory prediction, suggested by the poor performance in the last row of Table 7.

Effect of proposal interaction modeling: To demonstrate the effectiveness of the proposed pedestrian interaction modeling, we measure the performance for all pedestrians as well as pedestrians in a group. Specifically, we design a heuristic rule (based on locations and speeds) to discover pedestrian groups and assign each pedestrian a group label on the evaluation set. The details about the grouping algorithm can be found in supplementary. We evaluate the trajectory prediction performance on all pedestrians and the pedestrians belonging to a group with at least 5 pedestrians, shown in Table 8. The interaction modeling improves trajectory prediction performance on "all pedestrians" and achieve a larger boost for pedestrians that belong to groups (DE@3 improved from 49.67 to 48.89 by 1.6%).

4.5. Model inference speed

We measure the inference speed of our proposed model as well as baseline models on context range of 100m by

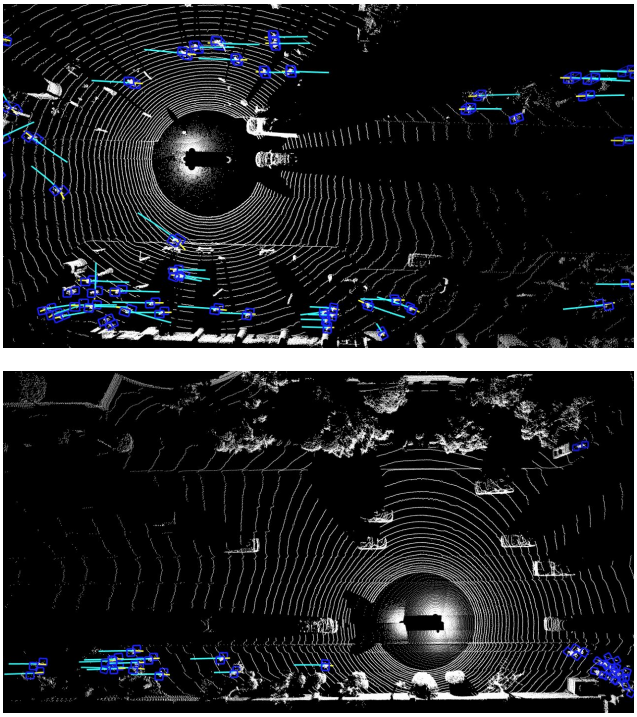

Figure 5. Qualitative examples of STINet. The blue box are detected pedestrians. The cyan and yellow lines are predicted future and history trajectories of STINet respectively.

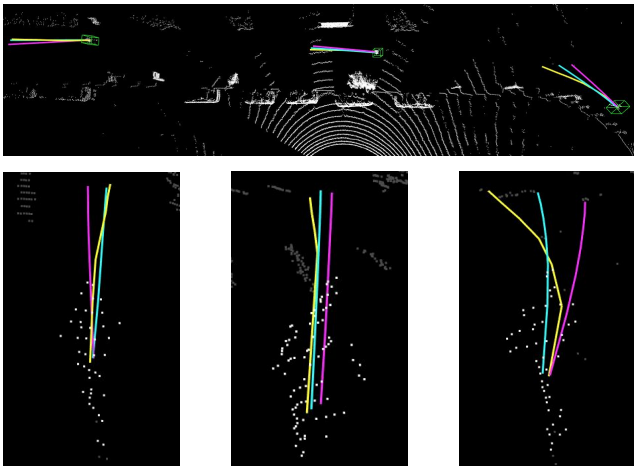

Figure 6. Comparison between MF-FRCNN and STINet. The yellow line is the ground-truth future trajectory for pedestrians. The pink and cyan lines are the predicted future trajectory from MF-FRCNN and STINet respectively. It is clear that our proposed method gives a much better prediction compared with the baseline, for all three pedestrians. Upper: the overview of three pedestrians. Lower: zoom-in visualization for three pedestrians.

100m as well as 150m by 150m. All models are implemented in TensorFlow and the inference is executed on a single nVIDIA Tesla V100 GPU. For the context range of 100m by 100m, IntentNet, MF-FRCNN and STINet have inference time of 60.9, 69.4 and 74.6ms respectively. Both two-stage models (MF-FRCNN and STINet) are slower than the single-stage model, and STINet is slightly slower than MF-FRCNN. However, all three models can achieve a real-time inference speed higher than 10Hz. For the maximum range of Waymo Open Dataset, *i.e.*, 150m by 150m, three models have inference time of 122.9, 132.1 and 144.7ms respectively.

4.6. Qualitative results

The visualization for the predictions of STINet is shown in Figure 5. The blue boxes are the detected pedestrians. The cyan and yellow lines are the predicted future and history trajectory for each detected pedestrian respectively. We show two scenarios where the SDC is stationary in the upper sub-figure and the SDC is moving fast in the lower sub-figure. It demonstrates that our model detects and predicts very accurately in both cases.

Figure 6 shows a detailed comparison between STINet and MF-FRCNN against the ground-truth for trajectory prediction. Green boxes are the ground-truth boxes. Yellow, pink and cyan lines are the ground-truth future trajectory as well as the predicted future trajectories from MF-FRCNN and STINet respectively. For the left two pedestrians who are walking in a straight line, both MF-FRCNN and STINet predict future trajectory reasonably well but the MF-FRCNN still has a small error compared with the ground-truth; for the right-most pedestrian who is making a slight left turn, MF-FRCNN fails to capture the details of its movement and gives an unsatisfactory prediction, while STINet gives a much better trajectory prediction.

5. Conclusion

In this paper, we propose STINet to perform joint detection and trajectory prediction with raw lidar point clouds as the input. We propose to build temporal proposals with pedestrians' both current and past boxes and learn a rich representation for each temporal proposal, with local geometry, dynamic movement, history path and interaction features. We show that by explicitly modeling the spatio-temporal-interaction features, both detection and trajectory prediction quality can be drastically improved compared with single-stage and two-stage baselines. This also makes us to re-think the importance of introducing second-stage and proposals, especially for the joint detection and trajectory prediction task. Comprehensive experiments and comparisons with baselines and state-of-the-arts confirm the effectiveness of our proposed method, and our method significantly improves the prediction quality while still achieves the real-time inference speed which makes our model practical to be used in real-world applications. Combining camera/map data and utilizing longer history with LSTMs could be investigated to further improve the prediction and we will explore them in future work.

References

[1] Waymo open dataset: An autonomous driving dataset, 2019. 2, 5

[2] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016. 1, 3

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. 2

[4] 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. 1, 2, 3, 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. 3

[6] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 2

[7] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2255–2264, 2018. 1, 3

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 4

[9] Joey Hong, Benjamin Sapp, and James Philbin. Rules of the road: Predicting driving behavior with a convolutional model of semantic interactions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8454–8462, 2019. 1, 3, 6

[10] Rui Hou, Chen Chen, and Mubarak Shah. Tube convolutional neural network (t-cnn) for action detection in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5822–5831, 2017. 2

[11] Vicky Kalogeiton, Philippe Weinzaepfel, Vittorio Ferrari, and Cordelia Schmid. Action tubelet detector for spatio-temporal action localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4405–4413, 2017. 2

[12] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. Lyft level 5 av dataset 2019. urlhttps://level5.lyft.com/dataset/, 2019. 2, 5

[13] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 1, 2, 3

[14] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B Choy, Philip HS Torr, and Manmohan Chandraker. Desire: Distant future prediction in dynamic scenes with interacting agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 336–345, 2017. 3

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1, 2

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 2

[17] 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. 1, 2, 3

[18] Anton Milan, S Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017. 1

[19] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, et al. Starnet: Targeted computation for object detection in point clouds. *arXiv preprint arXiv:1908.11069*, 2019. 6

[20] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2, 4, 6

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3

[23] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017. 2

[24] Chen Sun, Abhinav Shrivastava, Carl Vondrick, Rahul Sukthankar, Kevin Murphy, and Cordelia Schmid. Relational action forecasting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 273–283, 2019. 2

[25] Peng Tang, Chunyu Wang, Xinggang Wang, Wenyu Liu, Wenjun Zeng, and Jingdong Wang. Object detection in videos by high quality object linking. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 2

[26] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 399–417, 2018. 2

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

[28] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 670–685, 2018. 2

[29] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. *arXiv preprint arXiv:1910.06528*, 2019. 1, 6

[30] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 1, 3