Supplementary Material for STINet: Spatio-Temporal-Interactive Networks 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

1. Details of network architecture

In this section we provide some details of network architecture omitted in the original paper due to space limit.

1.1. Pillar feature encoding

The backbone network of STINet takes 6 frames of point clouds as input, which are calibrated to SDC's pose at the current (last) frame. Each point has 5 features: x, y, z, intensity and elongation. The x and y coordinates fall in the range of $[0, 150]$, which is splitted into a pillar grid with shape of 480 by 480. Each point is assigned to the closest pillar and its first two features (*i.e.* x and y) are subtracted by the pillar center position. Fully-connected layers with shared weights are applied to all points and map the 5-d feature vector of each point to a 64-d feature vector. After the fully-connected layers, all feature vectors belonging to the same pillar are aggregated by max-pooling. Thus each frame produces a pillar feature map with shape of 480 by 480 by 64. To reduce memory usage, pillar feature maps from consecutive two frames are concatenated, so finally there are three pillar feature maps with shape of 480 by 480 by 128.

1.2. Backbone ResUNet

Pillar feature maps are processed by weight-sharing ResUNets to generate backbone features. The ResUNet has three ResNet blocks, with 2, 2 and 3 ResNet units respectively. The output feature maps of these three ResNet blocks have shapes of $480 \times 480 \times 64$, $240 \times 240 \times 64$ and $120 \times 120 \times 64$ respectively. They are reshaped back to $480 \times 480 \times 64$ with deconvolution, and the concatenation with shape of $480 \times 480 \times 192$ serves as the backbone feature for the corresponding frame.

∗ Work done during an internship at Waymo.

1.3. Temporal region proposal network

Three backbone features are concatenated into a feature map with shape of 480 by 480 by 576, from which a temporal-aware feature map with shape of 480 by 480 by 256 is generated by convolutions. Two anchors with shape of 75cm by 75cm and 150cm by 150cm are set at every position of the temporal-aware feature map. 1×1 convolutions with 2, 10 and 12 neurons are applied on the feature map to predict objectiveness, current frame bounding-box regression and past frame bounding-box regression.

To mitigate the foreground-background imbalance issue, we adopt online hard example mining for computing losses. 16,384 (*i.e.* 3.56% out of $480 \times 480 \times 2$ anchors) anchors with highest foreground classification loss are sampled to compute final losses.

1.4. Proposal classification and regression

For each temporal proposal with the current frame box $\mathbf{p} = (x_0^p, y_0^p, w^p, l^p, h_0^p)$ (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 $\mathbf{gt} = (x_0^{gt}, y_0^{gt}, w_0^{gt}, l_0^{gt}, h_0^{gt})$. For the current frame, we generate a 5-d regression target $\mathbf{d}_0^p = (dx_0^a, dy_0^a, dw^a, dl^a, dh_0^a)$:

$$dx_0^p = (x_0^{gt} - x_0^p)/\sqrt{(x_0^p)^2 + (y_0^p)^2}$$

$$dy_0^p = (y_0^{gt} - y_0^p)/\sqrt{(x_0^p)^2 + (y_0^p)^2}$$

$$dw^p = \log \frac{w^{gt}}{w^p}$$

$$dl^p = \log \frac{l^{gt}}{l^p}$$

$$dh_0^p = \sin \frac{h_0^{gt} - h_0^p}{2}$$

With similar equations, we also compute t_{future} future regression targets for proposal $\mathbf{p}$ against the corresponding

ground-truth object: $\mathbf{d}_j^p = (dx_j^p, dy_j^p, dh_j^p)$ for each $j \in \{1, 2, \cdots, t_{\text{future}}\}$. Also a classification target s^p of 1, 0 or -1 is assigned based on the current frame IoU. The training objective of proposal classification and regression is the weighted sum of classification loss, current frame regression loss and future 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.

$$\mathcal{L}_{\text{P}} = \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{cur\_reg}}\mathcal{L}_{\text{cur\_reg}} + \lambda_{\text{future\_reg}}\mathcal{L}_{\text{future\_reg}}$$

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

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

$$\mathcal{L}_{\text{future\_reg}} = \sum_{j=1}^{t_{\text{future}}} \frac{\sum_{\mathbf{p}} \text{SmoothL1}(\mathbf{d}_j^p, \hat{\mathbf{d}}_j^p)\mathbb{1}(s^p \geq 1)}{\sum_{\mathbf{p}} \mathbb{1}(s^p \geq 1)}$$

2. Details of pedestrian groups

In order to do evaluation breakdown based on pedestrian group size, we design a heuristic rule to discover pedestrian groups and assign each pedestrian a group id as well as the group size.

Algorithm 1: Identify pedestrian groups

Input : List of pedestrians p_i with their locations (x_i, y_i) and speeds (vx_i, vy_i), where $i \in \{1, 2, \cdots, n\}$.

Output: Group id g_i and size s_i for each pedestrian.

$e[i, j] \leftarrow 0$ for $i, j \in \{1, \cdots, n\}$;
for $i \leftarrow 1$ *to* n **do**
  for $j \leftarrow 1$ *to* n **do**
    if $||(x_i - x_j, y_i - y_j)|| \leq 1.0$ ***and*** $||(vx_i - vx_j, vy_i - vy_j)|| \leq 1.0$ ***and*** $0.5 \leq \frac{||(vx_i, vy_i)||}{||(vx_j, vy_j)||} \leq 2.0$ **then**
      $e[i, j] \leftarrow 1$;
    end
  end
end
$\{cc_1, \cdots, cc_m\} \leftarrow$ConnectedComponents($e$);
for $i \leftarrow 1$ *to* m **do**
  for $j \leftarrow$NodesOf(cc_i) **do**
    $g_j \leftarrow i$;
    $s_j \leftarrow$ SizeOf(cc_i);
  end
end

The heuristic rule is designed based on the locations and speeds of pedestrians. Firstly a graph is built in which nodes are pedestrians and two nodes are connected if the corresponding two pedestrians have similar locations and speeds. Then each connected component is consider as a group and each pedestrian is assign the connected component id and size. The detailed algorithm is described in Algorithm 1.