VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation

开源代码 repo:
https://github.com/AliceNing/VectorNet_ning
https://github.com/DQSSSSS/VectorNet
https://github.com/xk-huang/yet-another-vectornet

1 核心思想

VectorNet 的核心思想是通过 vectorize 将 map 中的 context information 和 traffic agents 的轨迹进行编码 (避免了使用 HDmap, 减少计算量)，并通过 vector-level 和 Polyline-level 两种层次的图节点信息融合将信息传播到 Target Agent(TA) node 上，从而可以使用 decoder 对 TA 节点的轨迹预测。

2 构建 vector 表示

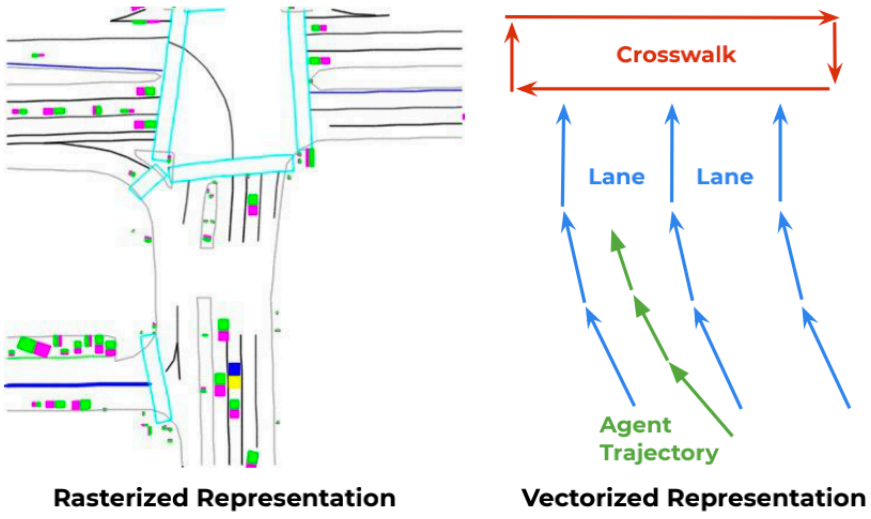

图 1: 地图信息、agent 的轨迹可以用 Polyline 来拟合，而每个 Polyline 可以用一组 vector 来表示

如图1中所示，对于一个 Polyline $\mathcal{P}_i$ 中的向量 v_i 定义如下：

$$\mathrm{v}_i = [\mathrm{d}_i^s, \mathrm{d}_i^e, \mathrm{a}_i, j] \tag{1}$$

其中 $\mathrm{d}^s, \mathrm{d}^e$ 代表着 vector 起点终点的坐标 (x, y) 或 (x, y, z), a 代表着当前 Polyline 的类型 (e.g., 行人轨迹, 十字路口, 车轨迹)，j 是 $\mathcal{P}_j$ 的 id，表示 $\mathrm{v}_i \in \mathrm{P}_j$。

- 对于 map 特征，选取一个起始点和终止点坐标，从样条上均匀采样相同空间距离的关键点，然后依次将相邻的关键点连接成向量。
- 对于 trajectory 特征，选取开始时间点，并将均匀间隔内的轨迹连成向量。

3 vector-level 子图卷积

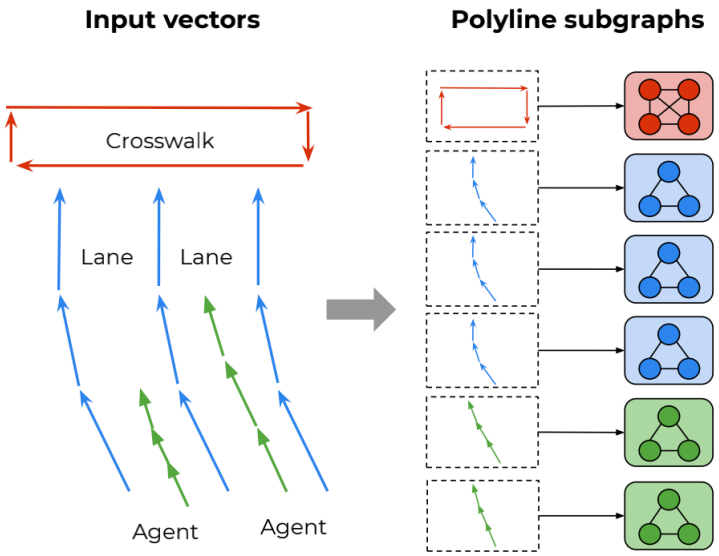

图 2: 有相同空间或语义信息的 vector-level 节点被连接成子图经过图卷积最后形成 Polyline-level 节点

如图2中所示, 对于一个 Polyline $\mathcal{P}_i$ 中的 Node $\{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_P\}$ 建立 vector-level 子图。如图3中所示, 单层的图卷积定义为:

$$\mathrm{v}_i^{(l+1)} = \varphi_{\mathrm{rel}}\left(g_{\mathrm{enc}}\left(\mathrm{v}_i^{(l)}\right), \varphi_{\mathrm{agg}}\left(\left\{g_{\mathrm{enc}}\left(\mathrm{v}_j^{(l)}\right)\right\}\right)\right) \tag{2}$$

- Node Encoder $g_{\mathrm{enc}}(\cdot)$ 通过 MLP 将 node 做 transfromation
- Aggregator$\varphi_{\mathrm{agg}}(\cdot)$ 通过 MaxPooling 融合相邻节点信息
- Relational Operator $\varphi_{\mathrm{rel}}(\cdot)$ 将特征拼接

最终 Polyline-level 的特征通过对 L_p 层的特征融合

$$\mathrm{p} = \varphi_{\mathrm{agg}}\left(\left\{\mathrm{v}_i^{(L_p)}\right\}\right) \tag{3}$$

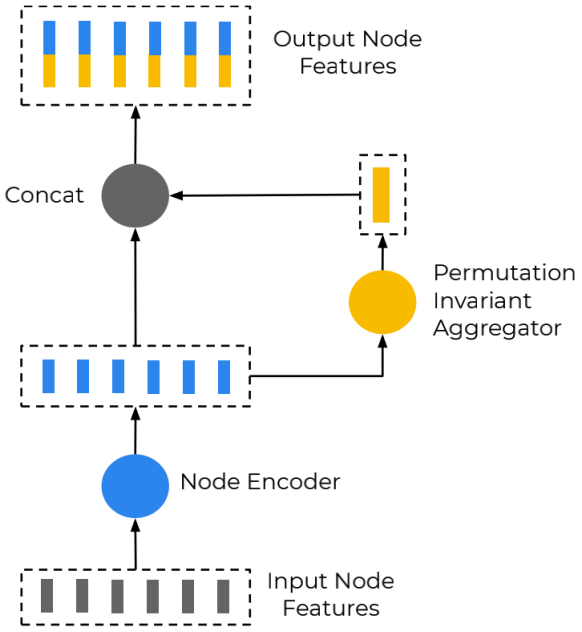

图 3: 单层子图卷积

4 Polyline-level **全局图卷积**

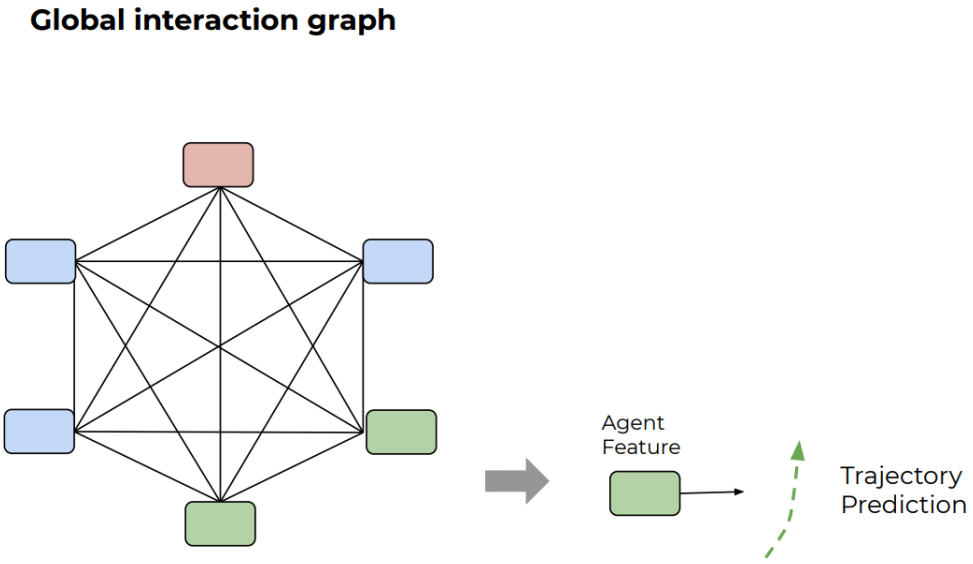

图 4: 使用图卷积对 Poyline 间的关联进行建模，最终对 Polyline 的轨迹进行解码

通过 Eq.(2), Eq.(3)，我们可以学到每个 Polyline 的特征，如图4所示，Polyline-level 图卷积采用全连接图 $\mathcal{A}$ 来对 Polyline 间的高阶关联进行建模：

$$\left\{\mathrm{p}_i^{(l+1)}\right\} = \mathrm{GNN}\left(\left\{\mathrm{p}_i^{(l)}\right\}, \mathcal{A}\right) \tag{4}$$

其中 $\left\{\mathrm{p}_i^{(l)}\right\}$ 表示 Polyline-level 全局图中的节点集合, VectorNet 利用了 self-attention 的思想来建立图网络:

$$\mathrm{GNN}(\mathrm{P}) = \mathrm{softmax}\left(\mathrm{P}_Q \mathrm{P}_K^T\right) \mathrm{P}_V$$

此处 $\mathcal{A}$ 为全连接矩阵，P 为节点特征矩阵 $\mathrm{P}_Q, \mathrm{P}_K$ 以及 P_V 为 P 的线性映射。

在经过 L_t 层 GNN 后，我们可以对 Target Agent(i.e. Target Polyline-level feature node) 的轨迹进行解码:

$$\mathrm{v}_i^{\text{future}} = \varphi_{\text{traj}}\left(\mathrm{p}_i^{(L_t)}\right) \tag{5}$$

$\varphi_{\text{traj}}(\cdot)$ 是解码器，论文中采用的 MLP，而 RNN，Multi-Path 等方法也可以采用。

5 Self-attention 和 GCN 的联系

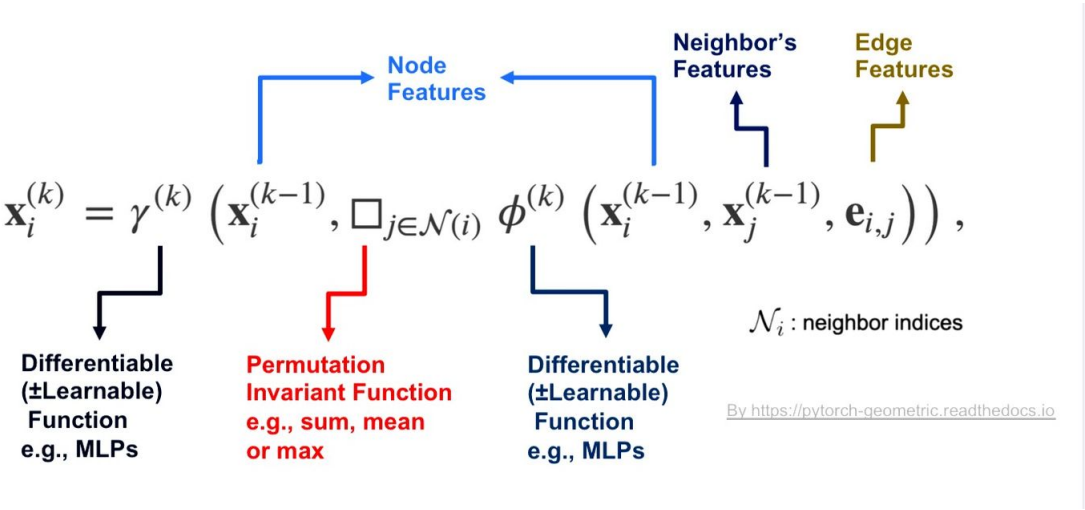

图 5: Message Passing 框架

大部分 GCN 和 Self-attention 都属于 Message Passing。GCN 中的 Message 从当前节点的邻居节点传播来，Self-attention 的 Message 从 Query 的 Key-Value 传播来。

5.1. GCN 的 Message Passing 解释

节点 i 在第 k 层 GCN 卷积的过程:

1. 把节点 i 的每一个邻居 j 与该节点的特征经过函数变换后形成一条 Message（对应图5公式里函数 ϕ ）；

2. 经过一个 Permutation Invariant (置换不变性)函数把该节点领域的所有 Message 聚合在一起（对应函数□）；
3. 再经过函数 γ 把聚合的领域信息和节点特征做一次函数变化，得到该节点在第 k 层图卷积后的特征 x_i；

5.2. Self-attention **的** Message Passing **解释**

Query i 计算 Self-attention 的过程：

1. Query i 的特征 x_i 会和每一个 Key j 的特征计算一个相似度 e_{ij};

$$e_{ij} = \frac{\left(x_i W^Q\right)\left(x_j W^K\right)^T}{\sqrt{d_z}}$$

2. 得到 Query i 与所有 Key 的相似度后经过 SoftMax 得到 Attention coefficient α_{ij};

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}}$$

3. 通过 Attention coefficient 加权 Value j 计算出 Query i 最后的输出 z_j;

$$z_i = \sum_{j=1}^{n} \alpha_{ij}\left(x_j W^V\right)$$

Self-attention 与 Message Passing 可对应为：

- 每个 Key-Value j 可以看作是 Query i 的邻居;
- 相似度和注意力系数的计算和最后 3. 中 Value j 与注意力系数 (α_{ij}) 相乘的操作可以对应为 Message Passing 中第一步构成 Message 的过程;
- 最后 Self-attention 的求和运算对应 Message Passing 中第二步的 Permutation Invariant 函数, 也就是说这里聚合领域信息的过程是通过 Query 对 Key-Value 聚合而来;
- Transformer 中的 Skip connection + MLP 对应 Message Passing 中的 γ 函数。