

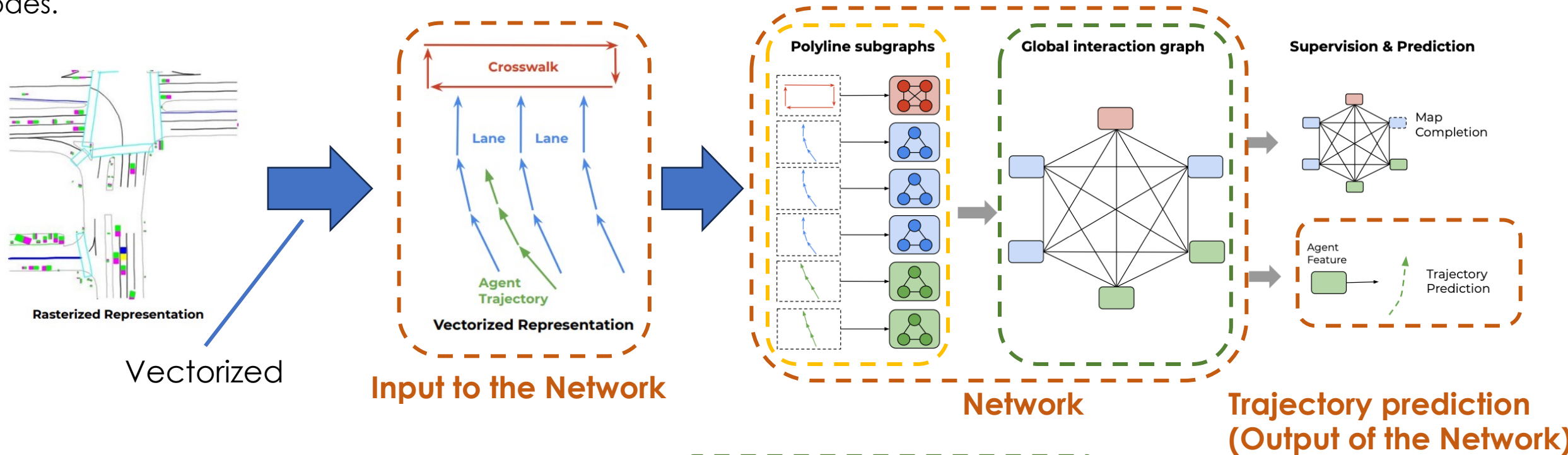
# VectorNet

**VectorNet:** Encoding HD Maps and Agent Dynamics from Vectorized Representation

- Performs **Trajectory prediction** via Neural network with vectorized inputs.
- Published by Waymo in 2020 (Paper link: <https://arxiv.org/abs/2005.04259>)

## Abstract:

To achieve behaviour prediction with multi-agent system, the inputs are vectorized and regarded as a subset of vectors. The subsets of the vectors are then regarded as nodes in the global graph that computes the interactions among the nodes.



- Network consists of ① Polyline subgraphs and ② Global interaction graph

# Overview of VectorNet

## Procedure

### ① Vectorize

Vectorize the inputs (Output from perception)



### ② Polyline Subgraphs Computation

Encode each vectors and aggregate the local information (connectivity)



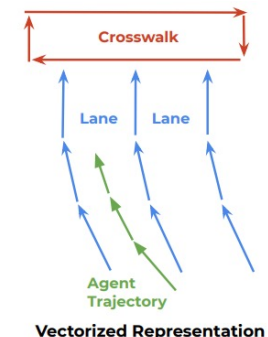
### ③ Global Graph Computation

Group the local graphs and make a node. Use the Attention to compute the relationship among different nodes.



### ④ Trajectory Decoder

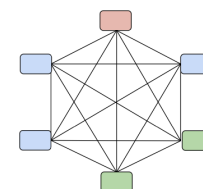
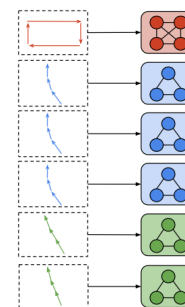
In the final step, make a next time step prediction for each node.



Vectorized Representation



Polyline subgraphs



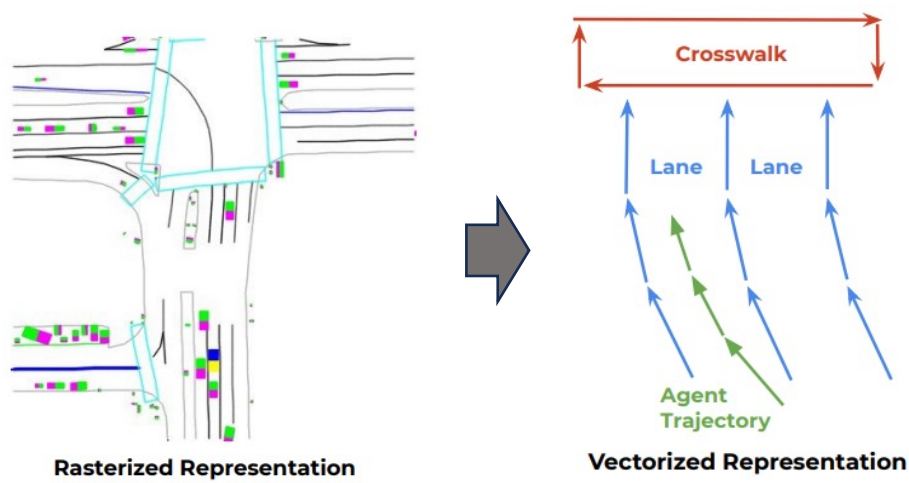
output



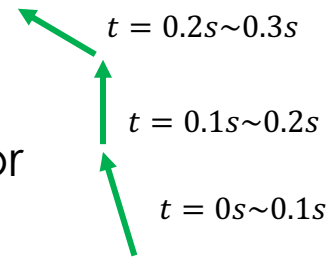
Next time step prediction

# 0. Vectorizer

Before feeding into the VectorNet model, the inputs are all vectorized to make it **graph representation**



- Trajectory:  
Represents 0.1s movement as a vector
- Lanes:  
Make a vector by connecting lane points, which is a result from key point detection.



## Definition of Polyline $\mathcal{P}_j$ and Vector $v_i$

$$v_i = [d_i^s, d_i^e, a_i, j]$$
  
( $i = 0, 1, 2, \dots, p$ )

$[x_s, y_s]$        $[x_e, y_e]$   
xy (start)      xy (end)

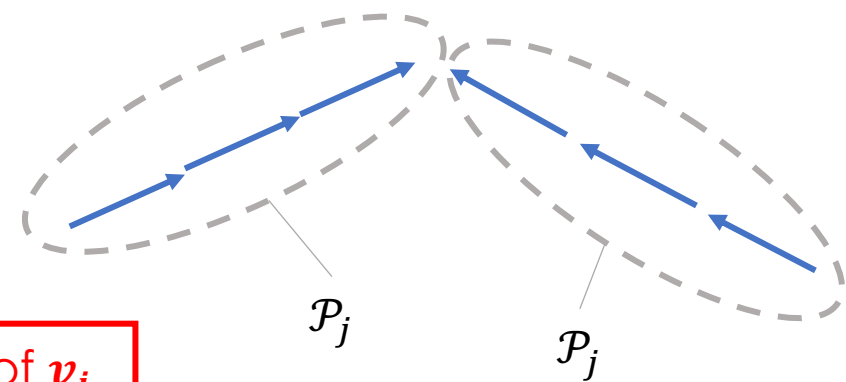


Index

Polyline type

- Object type
- Timestamp
- Road feature type
- Speed limit

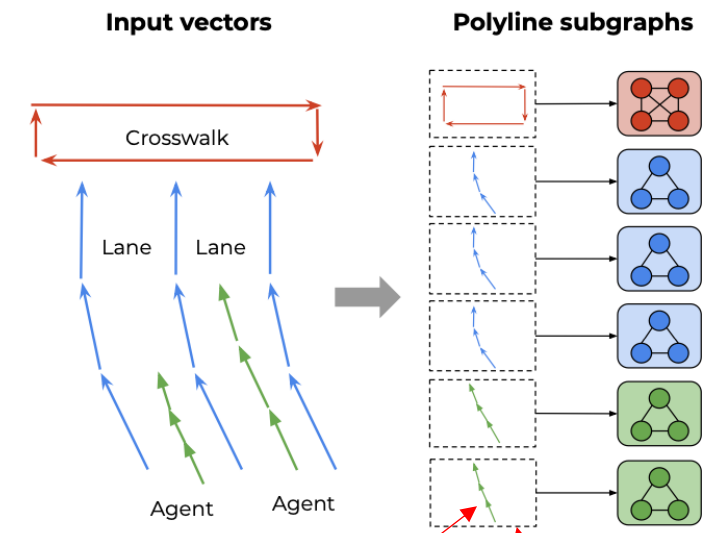
$\mathcal{P}_j$  consists of  $v_i$



Each group is called "Polyline"

# ① Polyline Subgraphs

After the inputs are vectorized and expressed in the graph representation, polyline subgraphs (local graphs) compute the interaction among vectors locally



Overview of Polyline subgraphs

(i is a timestep)

Each vector is represented as

$$v_i = [d_i^s, d_i^e, a_i, j]$$

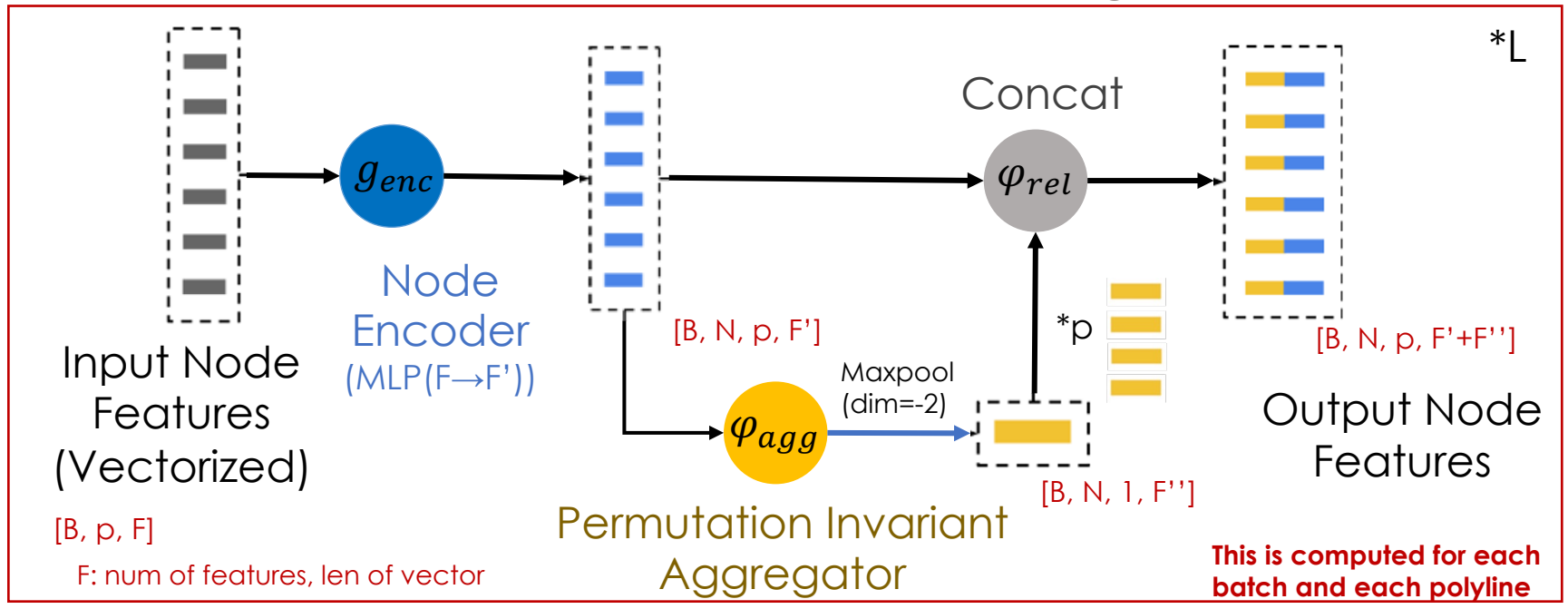
(i = 0,1,2 ..., p)

Each subset of vectors (Polyline)  
is represented as  $\mathcal{P}_j$

In the paper...

the MLP contains a single fully connected layer followed by layer normalization [3] and then ReLU non-linearity.

Computation Flow of Polyline Subgraphs



**Subgraph operation** ( $l_{th}$  layer to  $(l + 1)_{th}$  layer) :

$$v_i^{(l+1)} = \varphi_{rel} \left( g_{enc} \left( v_i^{(l)} \right), \varphi_{agg} \left( \{ g_{enc} (v_j^l) \} \right) \right)$$

Relational Operator (Concat))

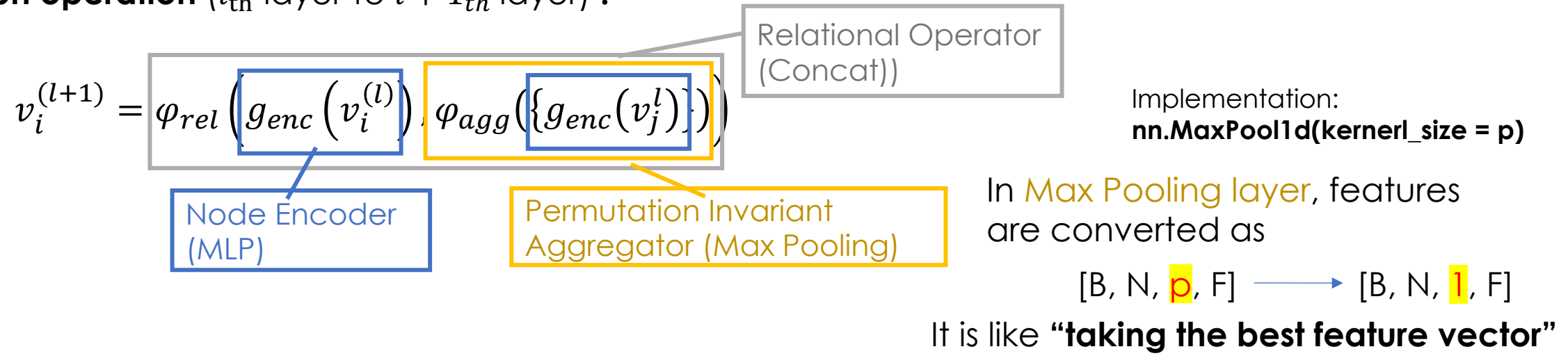
Node Encoder (MLP\*)

Permutation Invariant Aggregator (Max Pooling)

- \*MLP contains
- Linear
  - Layer Norm
  - ReLU

# ① Polyline Subgraphs

**Subgraph operation** ( $l_{th}$  layer to  $l + 1_{th}$  layer) :



## Procedure

For each Polyline  $\mathcal{P}_j$ , Vector  $v_i$  ( $i = 0, 1, 2, \dots, p$ ) which belongs to  $\mathcal{P}_j$  are fed into the layer above  $l$  times.

$$v_i^{(L)} = \varphi(v_i^{L-1})$$

Output from the last layer  $(i = 0, 1, 2, \dots, p)$

The last layer output shape:  
 $[B, N, p, F * 2^{num\_layers}]$

In the end, all the Vector outputs are taken into the Max Pooling layer, which yields 1 dim feature for each Polyline.

$$p_j = \varphi_{agg} \left( \{ v_0^{(L)}, v_1^{(L)}, \dots, v_p^{(L)} \} \right)$$

**Max Pooling**

Make 1 feature for each polyline

$$[B, N, p, F * 2^{num\_layers}] \rightarrow [B, N, 1, F * 2^{num\_layers}]$$

## ② Global Graph

Here, the model generates the features of global interaction among all the polylines. Self-Attention mechanism is adopted to enhance the correlation better.

After the local graph computation, each Polyline generates one feature.

$$\{p_0, p_1, \dots, p_P\}$$

### Global graph operation

$$\{p_i^{(l+1)}\} = \text{GNN}\left(\{p_i^{(l)}\}, \mathcal{A}\right)$$

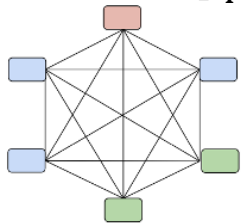
Adjacency matrix

Self-Attention is employed.

$$\text{GNN}(\mathbf{P}) = \text{softmax}(\mathbf{P}_Q \mathbf{P}_K^T) \mathbf{P}_V$$

\* $P_Q, P_K, P_V$  (Query, Key, Value) are generated from linear projection.

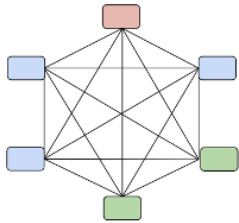
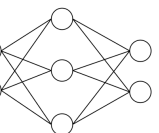
Subset of  $p_i^0$



$\{p_i^{(0)}\}$

**GNN**

\* $t$  layers

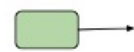


$\{p_i^{(L_t)}\}$

$[B, N, F \cdot 2^{\text{num\_layers}}]$

$[B, F \cdot 2^{\text{num\_layers}}]$

Agent Feature



Trajectory Prediction



Decoder (MLP)

$$\mathbf{v}_i^{\text{future}} = \varphi_{\text{traj}}(\mathbf{p}_i^{(L_t)})$$

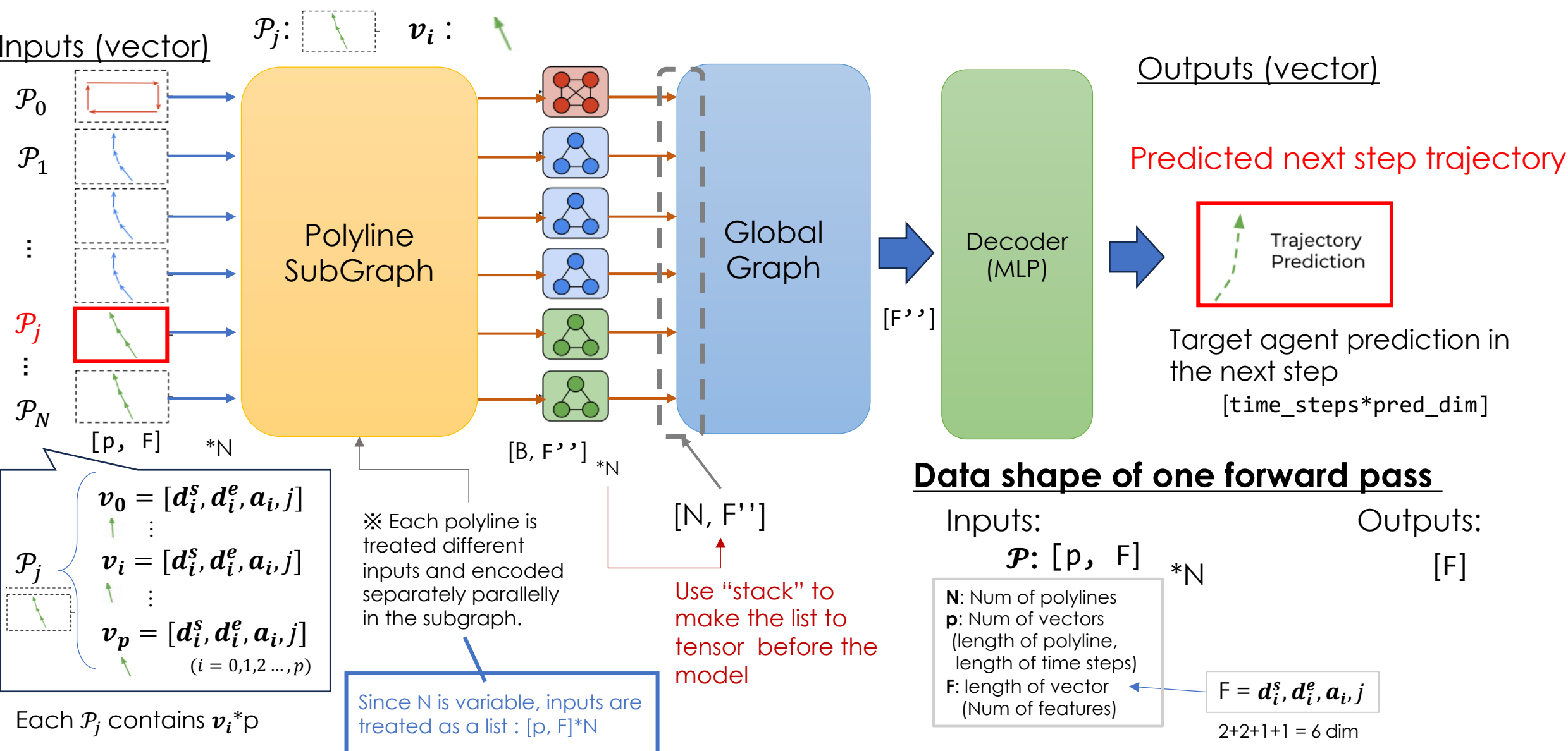
$[B, \text{num\_pred\_features}]$

This could only predict **one target agent!!**

Finally, the output is decoded from the output of GNN. This becomes the future trajectory of target agent. (=Prediction)

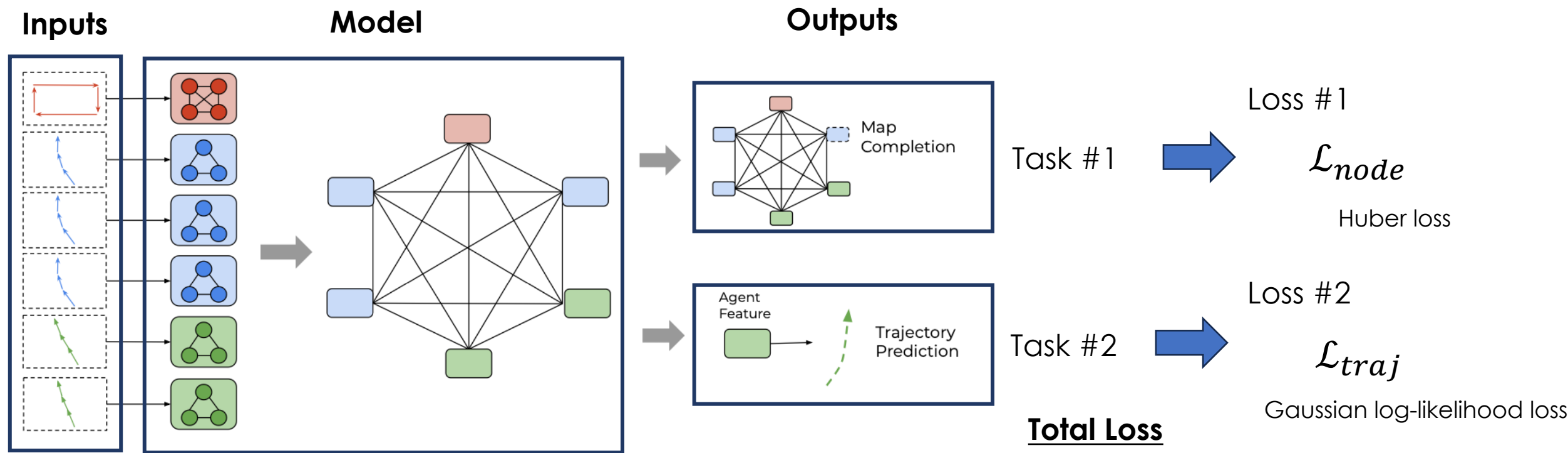
# Input and output

Model forward pass is performed **N\_agents times** for all agents separately. Every time, the coordinates of the inputs are transformed to the target agent centric coordinates.



# How to train, how the loss is computed?

During training, the polyline nodes are randomly masked out and attempt to recover the masked-out feature.



In addition to the “prediction” task, auxiliary **graph completion task** is introduced to encourage the better feature capturing.  
As outputs, the VectorNet returns two outputs; prediction of target agent, map completion task output.

## Map completion task:

Recover the masked-out node using decoder.  
Decoder is not used in the inference.

$$\hat{\mathbf{p}}_i = \varphi_{node}(\mathbf{p}_i^{(L_t)})$$

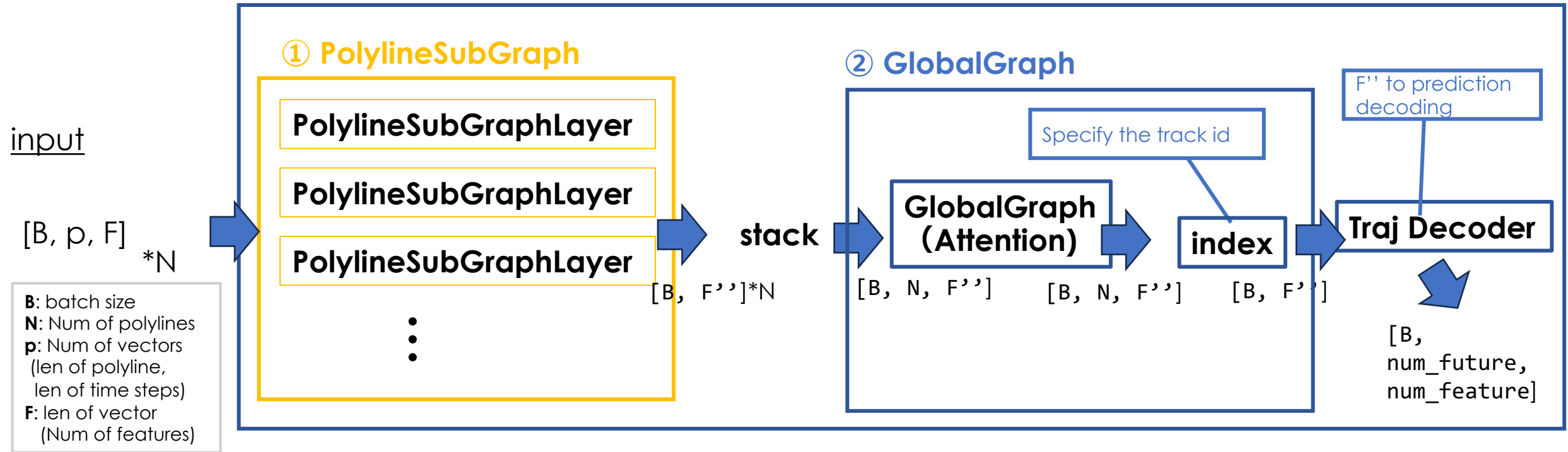
MLP Decoder

Set to 1.0

# Overall architecture and torch module

## Modules Architecture

**VectorNet**(nn.Module)



- Model inherits `nn.Module`.
- The num of input vector ( $p$ ), num of polylines ( $N$ ) are variable. MLP in the Subgraph only cares about the last dimension of the vector ( $=F$ ), because `nn.Linear` does " $(B, \dots, \text{dim\_in}) \rightarrow (B, \dots, \text{dim\_out})$ " this conversion.

# Code Pointers (Global Graph)

## (1) MultiHeadAttention

### Tensor shape

Input:

$[B, \text{num\_polylines}, \text{emb\_dim}]$

Output dim of the subgraph

$[B, \text{num\_polylines}, \text{emb\_dim}]$

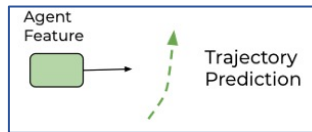
Take an index to specify the track id

Embedding features for each track id

$[B, \text{emb\_dim}]$

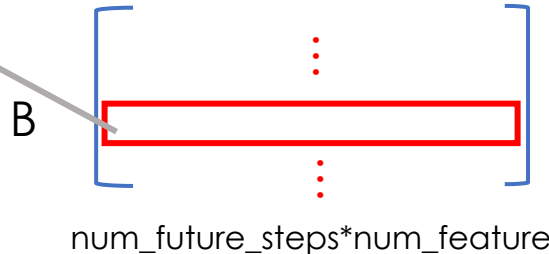
## (2) MLP (Trajectory Decoder)

$[B, \text{num\_future\_steps} * \text{num\_feature}]$



Each row corresponds to each track id's feature

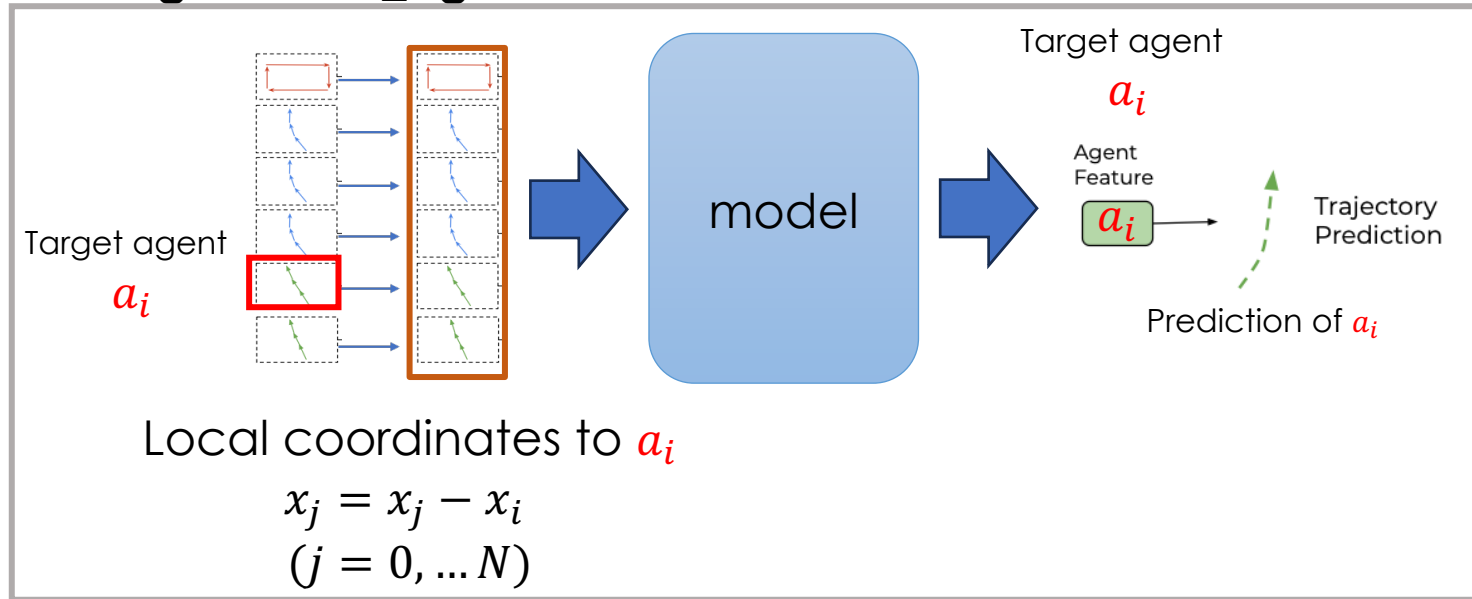
B: corresponds to length of track ids list



# Limitation of VectorNet

In the VectorNet, it executes **N\_agent times forward pass** to the model to predict the agent trajectory in the next step.

for **agent** in **all\_agents**;



For each agent, the inputs are transformed into agent-centric local coordinates including maps (lanes) information and fed into the model. The output will be the target agent prediction trajectory for the next time step.

**This process is repeated for all agents (N\_agent times) to predict the next step of the trajectory**, which lacks the efficiency. (GNN in VectorNet was not designed for multi-target decoding.)

## How to improve?

As a next step, Waymo has introduced global coordinate system shared across the scene.

### ◆ SceneTransformer (Waymo 2022):

- Uses **one global coordinate system**, shared across the scene.
- Each agent is represented as a token in a transformer.
- All agents are predicted **simultaneously** from the same forward pass.
- Map and agent features are shared → much more efficient.

### ◆ Wayformer (2023):

- Similar idea: uses **scene-level transformer architecture**.
- Single pass → joint multi-agent prediction with interaction modeling.

### ◆ UniAD (2023):

- Processes the entire scene in BEV, shared for all tasks.
- Predicts for **all agents** from a single shared BEV feature map.