Supplemental Materials for "Another Vertical View: A Hierarchical Network for Heterogeneous Trajectory Prediction via Spectrums"

1 Transformer Details

We employ the Transformer [1] as the backbone to encode trajectory spectrums and the scene context in the two proposed sub-networks. The Transformer used in the E-V^2-Net has two main parts, the Transformer Encoder and the Transformer Decoder, both of which are made up of several attention layers.

Attention Layers. Multi-Head Attention operations are applied in each of the attention layers. Following [1], each layer's multi-head dot product attention with H heads is calculated as:

$$\text{Attention}(\boldsymbol{q},\boldsymbol{k},\boldsymbol{v}) = \text{softmax}\left(\frac{\boldsymbol{q}\boldsymbol{k}^T}{\sqrt{d}}\right)\boldsymbol{v}, \tag{1}$$

$$\begin{aligned}&\text{MultiHead}(\boldsymbol{q},\boldsymbol{k},\boldsymbol{v}) = \\ &\qquad \text{fc}\left(\text{concat}(\{\text{Attention}_i(\boldsymbol{q},\boldsymbol{k},\boldsymbol{v})\}_{i=1}^{H})\right).\end{aligned} \tag{2}$$

Here, fc() denotes one fully connected layer that concatenates all heads' outputs. Query matrix $\boldsymbol{q}$, key matrix $\boldsymbol{k}$, and value matrix $\boldsymbol{v}$, are the three layer inputs. Each attention layer also contains an MLP (denoted as MLP_a) to extract the attention features further. It contains two fully connected layers. ReLU activations are applied in the first layer. Formally, we have output feature $\boldsymbol{f}_o$ of this layer:

$$\boldsymbol{f}_o = \text{ATT}(\boldsymbol{q},\boldsymbol{k},\boldsymbol{v}) = \text{MLP}_a(\text{MultiHead}(\boldsymbol{q},\boldsymbol{k},\boldsymbol{v})). \tag{3}$$

Transformer Encoder. The transformer encoder comprises several encoder layers, and each encoder layer contains an attention layer and an encoder MLP (MLP_e). Residual connections and normalization operations are applied to prevent the network from overfitting. Let $\boldsymbol{h}^{(l+1)}$ denote the output of l-th encoder layer, and $\boldsymbol{h}^{(0)}$ denote the encoder's initial input. For l-th encoder layer, the calculation of the layer output $\boldsymbol{h}^{(l+1)}$ can be written as:

$$\begin{aligned}\boldsymbol{a}^{(l)} &= \text{ATT}(\boldsymbol{h}^{(l)},\boldsymbol{h}^{(l)},\boldsymbol{h}^{(l)}) + \boldsymbol{h}^{(l)},\\ \boldsymbol{a}_n^{(l)} &= \text{Normalization}(\boldsymbol{a}^{(l)}),\\ \boldsymbol{c}^{(l)} &= \text{MLP}_e(\boldsymbol{a}_n^{(l)}) + \boldsymbol{a}_n^{(l)},\\ \boldsymbol{h}^{(l+1)} &= \text{Normalization}(\boldsymbol{c}^{(l)}).\end{aligned} \tag{4}$$

Transformer Decoder. Similar to the Transformer encoder, the Transformer decoder comprises several decoder layers, and each is stacked with two different attention layers. The first attention layer in the Transformer decoder focuses on the essential parts in the Transformer encoder's outputs $\boldsymbol{h}_e$ queried by the decoder's input $\boldsymbol{X}$. The second layer is the same self-attention layer as in the encoder. Similar to Eq. 4, we have the decoder layer's output feature $\boldsymbol{h}^{(l+1)}$:

$$\begin{aligned}\boldsymbol{a}^{(l)} &= \text{ATT}(\boldsymbol{h}^{(l)},\boldsymbol{h}^{(l)},\boldsymbol{h}^{(l)}) + \boldsymbol{h}^{(l)},\\ \boldsymbol{a}_n^{(l)} &= \text{Normalization}(\boldsymbol{a}^{(l)}),\\ \boldsymbol{a}_2^{(l)} &= \text{ATT}(\boldsymbol{h}_e,\boldsymbol{h}^{(l)},\boldsymbol{h}^{(l)}) + \boldsymbol{h}^{(l)},\\ \boldsymbol{a}_{2n}^{(l)} &= \text{Normalization}(\boldsymbol{a}_2^{(l)})\\ \boldsymbol{c}^{(l)} &= \text{MLP}_d(\boldsymbol{a}_{2n}^{(l)}) + \boldsymbol{a}_{2n}^{(l)},\\ \boldsymbol{h}^{(l+1)} &= \text{Normalization}(\boldsymbol{c}^{(l)}).\end{aligned} \tag{5}$$

Positional Encoding. Before feeding agents representations or trajectory spectrums into the Transformer, we add the positional coding to inform the relative position of each timestep or frequency portion in the sequential inputs. The position coding $\boldsymbol{f}_e^t$ at step t $(1 \le t \le t_h)$ is obtained by:

$$\begin{aligned}&\boldsymbol{f}_e^t = (f_{e0}^t, ..., f_{ei}^t, ..., f_{ed-1}^t) \in \mathbb{R}^d,\\ &\text{where } f_{ei}^t = \begin{cases}\sin\left(t/10000^{d/i}\right), & i \text{ is even;}\\ \cos\left(t/10000^{d/(i-1)}\right), & i \text{ is odd.}\end{cases}\end{aligned} \tag{6}$$

Then, we have the positional coding matrix $\boldsymbol{f}_e$ that describes t_h steps of sequences:

$$\boldsymbol{f}_e = (\boldsymbol{f}_e^1, \boldsymbol{f}_e^2, ..., \boldsymbol{f}_e^{t_h})^T \in \mathbb{R}^{t_h \times d}. \tag{7}$$

The final Transformer input $\boldsymbol{X}_T$ is the addition of the original sequential input $\boldsymbol{X}$ and the positional coding matrix $\boldsymbol{f}_e$. Formally,

$$\boldsymbol{X}_T = \boldsymbol{X} + \boldsymbol{f}_e \in \mathbb{R}^{t_h \times d}. \tag{8}$$

Layer Configurations. We employ $L = 4$ layers of encoder-decoder structure with $H = 8$ attention heads in each Transformer-based sub-networks. The MLP_e and the MLP_d have the same shape. Both of them consist of two fully connected layers. The first layer has 512 output units with the ReLU activation, and the second layer has 128 but does not use any activations. The output dimensions of fully connected layers used in multi-head attention layers are set to d = 128.

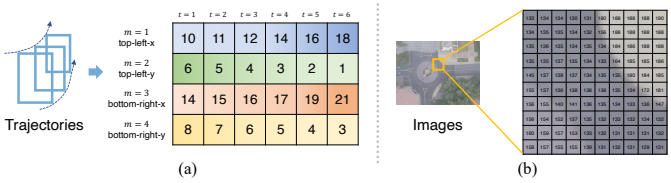

Fig. 1. Matrices views of a trajectory (2D bounding box) and an image.

2 Linear Least Squares Trajectory Prediction

The linear least squares trajectory prediction method aims to minimize the mean square error between the predicted and agents' groundtruth trajectories. When predicting, we perform a separate least squares operation for each dimension of the M-dimensional observed trajectory $\boldsymbol{X}$. Simply, we want to find the $\boldsymbol{x}_m = (b_m, w_m)^T \in \mathbb{R}^2$ $(1 \leq m \leq M)$, such that

$$\hat{\boldsymbol{Y}} = (\hat{\boldsymbol{Y}}_1, \hat{\boldsymbol{Y}}_2, ..., \hat{\boldsymbol{Y}}_m, ..., \hat{\boldsymbol{Y}}_M) \in \mathbb{R}^{t_f \times M},$$
$$\text{where } \hat{\boldsymbol{Y}}_m = \boldsymbol{A}_f \boldsymbol{x}_m = \begin{pmatrix} 1 & t_h+1 \\ 1 & t_h+2 \\ ... & ... \\ 1 & t_h+t_f \end{pmatrix} \begin{pmatrix} b_m \\ w_m \end{pmatrix}. \tag{9}$$

For one of agents' observed M-dimensional trajectory $\boldsymbol{X} \in \mathbb{R}^{t_h \times M}$, we have the trajectory slice on the m-th dimension

$$\boldsymbol{X}_m = (r_{m1}, r_{m2}, ..., r_{mt_h})^T. \tag{10}$$

Suppose we have a coefficient matrix $\boldsymbol{A}_h$, where

$$\boldsymbol{A}_h = \begin{pmatrix} 1 & 1 & 1 & ... & 1 \\ 1 & 2 & 3 & ... & t_h \end{pmatrix}^T. \tag{11}$$

We aim to find a $\boldsymbol{x}_m \in \mathbb{R}^2$, such that the mean square $\|\boldsymbol{A}_h \boldsymbol{x}_m - \boldsymbol{X}_m\|_2^2$ could reach its minimum value. Under this condition, we have

$$\boldsymbol{x}_m = (\boldsymbol{A}_h^T \boldsymbol{A}_h)^{-1} \boldsymbol{A}_h^T \boldsymbol{X}_m. \tag{12}$$

Then, we have the predicted m-th dimension trajectory

$$\hat{\boldsymbol{Y}}_m = \boldsymbol{A}_f \boldsymbol{x}_m. \tag{13}$$

The final M-dimensional predicted trajectory $\hat{\boldsymbol{Y}}$ is obtained by stacking all results. Formally,

$$\hat{\boldsymbol{Y}} = \boldsymbol{A}_f(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_M). \tag{14}$$

3 2D DFT v.s. Bilinear Structure

We apply different transforms on *each dimension* of the trajectory to obtain the corresponding trajectory spectrums either in V^2-Net or the enhanced E-V^2-Net. Moreover, considering that one of our main contributions is to establish connections between trajectories (or spectrums) of different dimensions, a more natural idea might be to apply some 2D transform directly to these trajectories. However, it appears to be less effective from both theoretical analyses and experimental results. In this section, we will discuss the discrepancy between the 2D transform and the proposed bilinear structure in describing the two factors, including the frequency response of the trajectory and the dimension-wise interactions, from different perspectives, taking DFT as an example.

DFT on Different Directions in Trajectories. The 2D DFT can be decomposed into two consecutive 1D DFTs performed in different directions of the target 2D matrix. The M-dimensional trajectory $\boldsymbol{X} \in \mathbb{R}^{N \times M}$ is also a 2D matrix similar to 2D grayscale images. Although the 2D DFT and its variations have achieved impressive results in tasks related to image processing, they might be not directly applied to trajectories. We will analyze this problem specifically by focusing on the different directions of the transforms in the trajectory.

Fig. 1 shows an $M = 4$ 2D bounding box trajectory and an image with the matrix view. As shown in Fig. 1 (b), whether the image is sliced horizontally or vertically, the resulting vector could reflect the change in grayscale values in a particular direction. Therefore, when performing the 2D transform, the first 1D transform will extract the frequency response in a specific direction, while the second 1D transform will fuse it with the frequency response in the vertical direction.

In contrast, different slice directions of the trajectory may lead to different meanings. If the trajectories are sliced according to the time dimension, then four 1D time series will be obtained as shown in Fig. 1 (a). Applying 1D transforms to these four sequences, we can obtain four trajectory spectrums that could describe agents' frequency responses and thus describe their motions from the global plannings and interaction details at different scales. However, if the trajectory is sliced from the dimensional direction, then N (6 in the figure) 4-dimensional vectors will be obtained. These vectors contain information about agents' locations and postures at a particular moment. In addition, the focused dimension-wise interactions are also contained in these vectors. However, it should be noted that we are more interested in the relationships between the data in these vectors, *i.e.*, the "edges" between the different data. If a 1D transform is applied to these vectors, the resulting spectrum may hardly have a clear physical meaning, because the temporal or spatial adjacencies of these points are not reflected in these 4-dimensional vectors.

For example, suppose we want to apply the 1D DFT on the 4-dimensional (2D bounding box) vector $\boldsymbol{x} = (x_l, y_l, x_r, y_r)^T$. Simply, we have:

$$\begin{aligned} \boldsymbol{\mathcal{X}} = \text{DFT}[\boldsymbol{x}] &= \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{pmatrix} \begin{pmatrix} x_l \\ y_l \\ x_r \\ y_r \end{pmatrix} \\ &= \begin{pmatrix} x_l + y_l + x_r + y_r \\ x_l - x_r - j(y_l - y_r) \\ x_l - y_l + x_r - y_r \\ x_l - x_r + j(y_l - y_r) \end{pmatrix}. \end{aligned} \tag{15}$$

Accordingly, we have its fundamental frequency portion $\boldsymbol{\mathcal{X}}[0] = x_l + y_l + x_r + y_r$ and the high-frequency portion $\boldsymbol{\mathcal{X}}[2] = x_l - y_l + x_r - y_r$. However, since the four positions $\{x_l, y_l, x_r, y_r\}$ do not have specific time-dependent or space-dependent like time-sequences and images, these

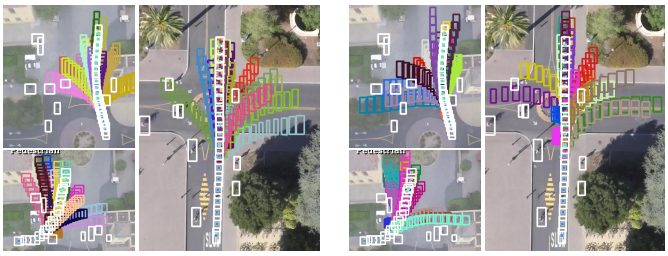

Fig. 2. Visualized comparisons of 2D DFT bilinear structure.

frequency components may hardly reflect the specific frequency response. For example, the fundamental frequencies can represent their average value for a time series, yet the values obtained by directly summing the 4 position coordinates of the 2 points of the 2D bounding box would be uninterpretable. In other words, each element in this 4-dimensional vector is relatively independent, and their connection relationships are more like a *graph* rather than a sequence where an order is required.

Quantitative Analyses. To verify our thoughts, we perform ablation experiments on SDD and nuScenes to compare the effects of 2D DFT and the bilinear structure quantitatively. As shown in TABLE 1, the results of APP2 and APP3 (or APP5 and APP6) show that 2D DFT does not improve quantitative trajectory prediction performance as effectively as bilinear structures. On the contrary, in the prediction of the more complex 3D bounding boxes ($M = 6$), using 2D DFT instead degrades the prediction performance compared to 1D DFT when no bilinear structures are used. These experimental results validate our thoughts of not using 2D transforms but bilinear structures.

Qualitative Analyses. We visualize the prediction results of different models under the effect of 2D DFT and bilinear structure qualitatively. As shown in Fig. 2, the V^2-Net (2D DFT) performs not as well as E-V^2-Net in both the prediction of agent motions and the interactions within the bounding box. In detail, predictions given by V^2-Net (2D DFT) capture fewer path possibilities in the top left prediction scenario. In addition, some predicted trajectories are with less smoothness and naturalness. For example, the prediction in color #DF6091 to the left of the bottom left prediction scene gives a turn with a large angle to observation, which could be not physical-acceptable in the actual scenario. In contrast, predictions given by E-V^2-Net have not shown similar results in this scenario. On the other hand, as shown in the traffic circle prediction scenario on the right, the shape of the bounding box is not well maintained in V^2-Net's predictions, such as the prediction in color #93C3CA to turn right to across the street.

4 A Graph-View of Bilinear Structure

As mentioned above, the modeling of dimension-wise interactions focuses more on the relations of different trajectory dimensions, which could be difficult to represent by the 1D transform. The bilinear structure used in this manuscript learns this connection relation through the outer product, pooling, and fully connected networks. To make it easier to understand, we further explain it from a graph view.

Given an undirected graph $\boldsymbol{G}(t) = (\boldsymbol{V}(t), \boldsymbol{E}(t))$ with a time variable t, where $\boldsymbol{V}(t)$ is the set of vertices, which contains all the position information of one agent at time t, and $\boldsymbol{E}(t)$ is the set of edges, which represents the connection relationships between these vertices at time t. Formally,

$$\begin{aligned}\boldsymbol{V}(t) &= \{f(r_{1t}), f(r_{2t}), ..., f(r_{Mt})\} \\ &= \{\boldsymbol{f}_{1,t}, \boldsymbol{f}_{2,t}, ..., \boldsymbol{f}_{M,t}\}.\end{aligned} \tag{16}$$

where

$$\boldsymbol{f}_{m,t} = f(r_{mt}) \in \mathbb{R}^d \tag{17}$$

indicates an embedding function to map these vertices into the high-dimension feature space. To establish and learn the connections between these vertices, we define the trainable adjacency matrix as:

$$\boldsymbol{A}(t) = \begin{pmatrix} \boldsymbol{W}_{1,1}(t) & \cdots & \boldsymbol{W}_{1,M}(t) \\ \vdots & \ddots & \vdots \\ \boldsymbol{W}_{M,1}(t) & \cdots & \boldsymbol{W}_{M,M}(t) \end{pmatrix}. \tag{18}$$

Here, each matrix $\boldsymbol{W}_{i,j} \in \mathbb{R}^{d\times d}$ are made trainable. They are used to describe the relation between node i and node j (*i.e.*, $\boldsymbol{f}_{i,t}$ and $\boldsymbol{f}_{j,t}$).

We converge the information on the node edges by graph convolution. Formally,

$$\boldsymbol{f}'_{m,t} = \sigma\left(\boldsymbol{W}'_m \text{Flatten}\left(\sum_{j=1}^{M} \boldsymbol{W}_{m,j}(t)\boldsymbol{f}_{m,t} \otimes \boldsymbol{f}_{j,t}\right)\right), \tag{19}$$

where σ represents a non-linear activation, and the $\boldsymbol{W}'_m$ is another trainable weight matrix. Finally, we have the refined vertices

$$\boldsymbol{V}'(t) = \{\boldsymbol{f}'_{1,t}, \boldsymbol{f}'_{2,t}, ..., \boldsymbol{f}'_{M,t}\}. \tag{20}$$

It is worth noting that the above Eq. 19 and the bilinear structure introduced in the manuscript describe the same network structure, despite their difference in representation. To reduce unnecessary misunderstandings, we do not describe the network inference process through this graph form in the manuscript, although the use of a graph may make it easier to understand the motivation for the use of the bilinear model. In addition, all the operations above are performed on time series. If we use trajectory spectrums to replace the Eq. 17, and take the frequency variable n to instead the time variable t, we have

$$\boldsymbol{\mathcal{V}}(n) = \{\boldsymbol{f}_{m,n}\}_{m=1}^{\mathcal{M}}, \quad \text{where } \boldsymbol{f}_{m,n} = f(s_{n,m}). \tag{21}$$

Accordingly, we have the refined vertices' spectrum representations:

$$\boldsymbol{\mathcal{V}}'(t) = \{\boldsymbol{f}'_{1,n}, \boldsymbol{f}'_{2,n}, ..., \boldsymbol{f}'_{\mathcal{M},n}\}. \tag{22}$$

Then, the outer product matrix $\boldsymbol{R}[n,:,:]$ has become the adjacency matrix of the graph $\boldsymbol{G}(n) = (\boldsymbol{\mathcal{V}}(n), \boldsymbol{\mathcal{E}}(n))$ on the frequency node $n \in [1, \mathcal{N}_h]$.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008. 1

TABLE 1
Validation of 2D DFT and bilinear structures with *best-of-20* on SDD (2D bounding box) the nuScenes (3D bounding box).

No.	Model	Type	T	N_{key}	BS	Dataset	ADE ↓	FDE ↓	cADE ↓	cFDE ↓	AIoU ↑	FIoU ↑
APP1	V^2-Net	bb	DFT	3	×	SDD (2D bounding box)	6.78	10.73	6.60	10.54	0.717	0.601
APP2	V^2-Net	bb	2D DFT	3	×		6.74	10.84	6.58	10.68	0.723	0.602
APP3	E-V^2-Net	bb	DFT	3	✓		6.62	10.57	6.47	10.41	0.725	0.604
APP4	V^2-Net	3dbb	DFT	2	×	nuScenes (3D bounding box)	0.229	0.335	0.203	0.293	0.747	0.666
APP5	V^2-Net	3dbb	2D DFT	2	×		0.234	0.341	0.209	0.301	0.739	0.656
APP6	E-V^2-Net	3dbb	DFT	2	✓		0.210	0.300	0.184	0.260	0.762	0.688