PriorMotion: Generative Class-Agnostic Motion Prediction with Raster-Vector Motion Field Priors

Supplementary Material

1. More Details about Model Designs

We present more detailed model designs here. The detailed model architecture of different modules is shown in Sec. 3.2.2 and Sec. 3.2.3.

1.1. RVpE

Sinusoidal embeddings. This embedding method effectively captures the positional information of each grid cell, providing a rich and continuous representation that is invariant to the absolute position but sensitive to the relative distances between coordinates. The sinusoidal embedding for a coordinate p and dimension j is given by:

$$E(p, 2j) = \sin\left(\frac{p}{T^{2j/d}}\right), \quad E(p, 2j+1) = \cos\left(\frac{p}{T^{2j/d}}\right) \tag{A.1}$$

where T is the scaling factor, typically set to 1000, and d is the total dimension of the embedding.

Motion vector encoding. To effectively capture both the temporal dynamics and the spatial characteristics of the instances, this module is designed to encode the motion behaviors of individual instances, such as vehicles and pedestrians. This module combines a Long Short-Term Memory (LSTM) network with a Multi-Layer Perceptron (MLP) to capture both temporal and spatial features. The LSTM network processes the sequential features of each instance, while the MLP further refines these features to produce a compact and meaningful representation.

Mathematically, the process can be described as (A.2-A.6). Let $\mathbf{x}_t$ be the input feature vector at time step t. The LSTM network updates its hidden state $\mathbf{h}_t$ and cell state $\mathbf{c}_t$ as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{A.2}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{A.3}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{A.4}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \tag{A.5}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{A.6}$$

where σ is the sigmoid activation function, $\odot$ denotes element-wise multiplication, and $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are the weight matrices and bias vectors, respectively.

The output of the LSTM, $\mathbf{h}_t$, is then fed into an 3-layers MLP to further refine the features (A.7)-(A.9).

$$\mathbf{z}_1 = \phi(\mathbf{W}_1 \mathbf{h}_t + \mathbf{b}_1) \tag{A.7}$$

$$\mathbf{z}_2 = \phi(\mathbf{W}_2 \mathbf{z}_1 + \mathbf{b}_2) \tag{A.8}$$

$$\vdots$$

$$\mathbf{y}_t = \mathbf{W}_L \mathbf{z}_{L-1} + \mathbf{b}_L \tag{A.9}$$

where ϕ is the activation function used in the hidden layers, and $\mathbf{y}_t$ is the final refined feature vector.

Attention mechanism in VpE. The Transformer Encoder is responsible for capturing the temporal and spatial dependencies in the embedding instance sequences. The self-attention mechanism is designed to capture the interactions within the input sequence. It uses a hidden dimension of 256, 4 attention heads, and 6 layers. Each layer of the self-attention mechanism includes a multi-head attention module and a feed-forward network with a hidden dimension of 256 and a dropout rate of 0.1. The cross-attention mechanism is designed to capture the interactions between the input sequence and the BEV features. It also uses a hidden dimension of 256 and 4 attention heads. A multi-head attention module and a feed-forward network are used with a hidden dimension of 1024 and a dropout rate of 0.1.

Rasterized prior encoding. The rasterized prior encoding module is designed to encode the grid-based representation of the environment using a self-attention mechanism(shown in Fig.A.1). For temporal self-attention, the input tensor has dimensions $[B, T, H, W, C]$, where B is the batch size, T is the number of time steps, H and W are the height and width of the grid, and C is the number of input channels. A 1x1 convolutional layer reduces the number of input channels to C'. The query (Q), key (K), and value (V) matrices are then projected from this transformed tensor, reshaped into $[B, THW, C']$, and processed through the scaled dot-product attention mechanism(A.10)-(A.17):

$$Q = \text{Conv1x1}(\mathbf{X}) \in \mathbb{R}^{B \times T \times H \times W \times C'}, \tag{A.10}$$

$$K = \text{Conv1x1}(\mathbf{X}) \in \mathbb{R}^{B \times T \times H \times W \times C'}, \tag{A.11}$$

$$V = \text{Conv1x1}(\mathbf{X}) \in \mathbb{R}^{B \times T \times H \times W \times C'}, \tag{A.12}$$

$$Q' = \text{reshape}(Q) \in \mathbb{R}^{B \times THW \times C'}, \tag{A.13}$$

$$K' = \text{reshape}(K) \in \mathbb{R}^{B \times THW \times C'}, \tag{A.14}$$

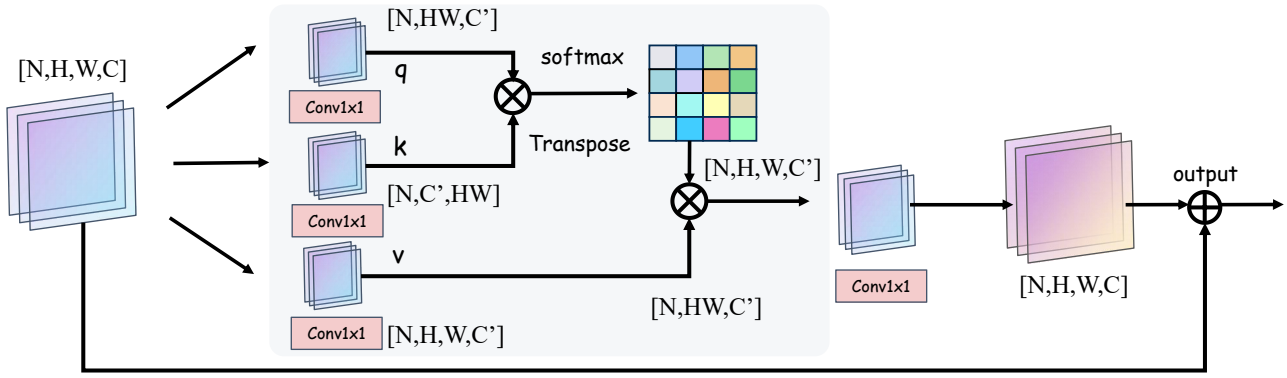

Figure A.1. The details of self-attention mechanism. A 1x1 convolutional layer reduces the number of input channels to C'. The query (Q), key (K), and value (V) matrices are then projected from this transformed tensor, reshaped into $[B, THW, C']$, and processed through the scaled dot-product attention mechanism.

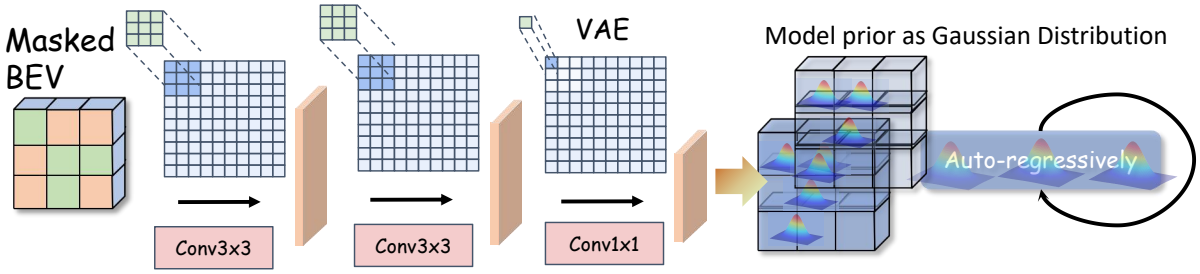

Figure A.2. The details of VAE. The VAE consists of three convolutional layers, each followed by batch normalization and a leaky ReLU[53] activation function.

Table A.1. This table presents the evaluation of different Gaussian sizes (1x1, 16x16, 32x32, and 64x64) for the VAE model in terms of static objects, moving objects with speed $\leq$ 5m/s, moving objects with speed $>$ 5m/s, and overall motion stability. Notably, the 64x64 scale shows the best performance in most categories, especially for high-speed objects and overall motion stability, indicating a potential trade-off between accuracy and efficiency.

Model	Gaussian Size	Static		Speed $\leq$ 5m/s		Speed $>$ 5m/s		Motion Stability
		Mean↓	Median↓	Mean↓	Median↓	Mean↓	Median↓	Variance
VAE	1x1	0.0339	0	0.2493	0.0921	0.9141	0.6092	0.01525
	16x16	0.0295	0	0.2317	0.0926	0.8818	0.6148	0.00834
	32x32	0.0288	0	**0.2258**	**0.0859**	0.8542	0.5916	0.00847
	64x64	**0.0285**	**0**	0.2364	0.0893	**0.8064**	**0.5869**	**0.00741**

$$V' = \text{reshape}(V) \in \mathbb{R}^{B\times THW\times C'}, \tag{A.15}$$

$$\mathcal{A} = \text{softmax}\left(\frac{Q'K'^{\top}}{\sqrt{C'}}\right)V' \in \mathbb{R}^{B\times THW\times C'}, \tag{A.16}$$

$$\mathbf{Y} = \text{reshape}(\mathcal{A}) \in \mathbb{R}^{B\times T\times H\times W\times C'}. \tag{A.17}$$

The resulting output tensor $\mathbf{Y}$ has the same spatial-temporal dimensions as the input, $[B, T, H, W, C']$, preserving the original structure while embedding temporal attention.

For spatial self-attention, the input tensor has dimensions $[B, H, W, C]$, where H and W are the height and width of the grid, and C is the number of input channels. A 1x1 convolutional layer similarly reduces the number of input channels to C'. The query (Q), key (K), and value (V) matrices are then projected, reshaped into $[B, HW, C']$, and processed through the scaled dot-product attention mechanism, as shown in (A.18)-(A.24). The output tensor $\mathbf{Y}$ maintains the spatial dimensions of the input, $[B, H, W, C']$, while embedding spatial attention.

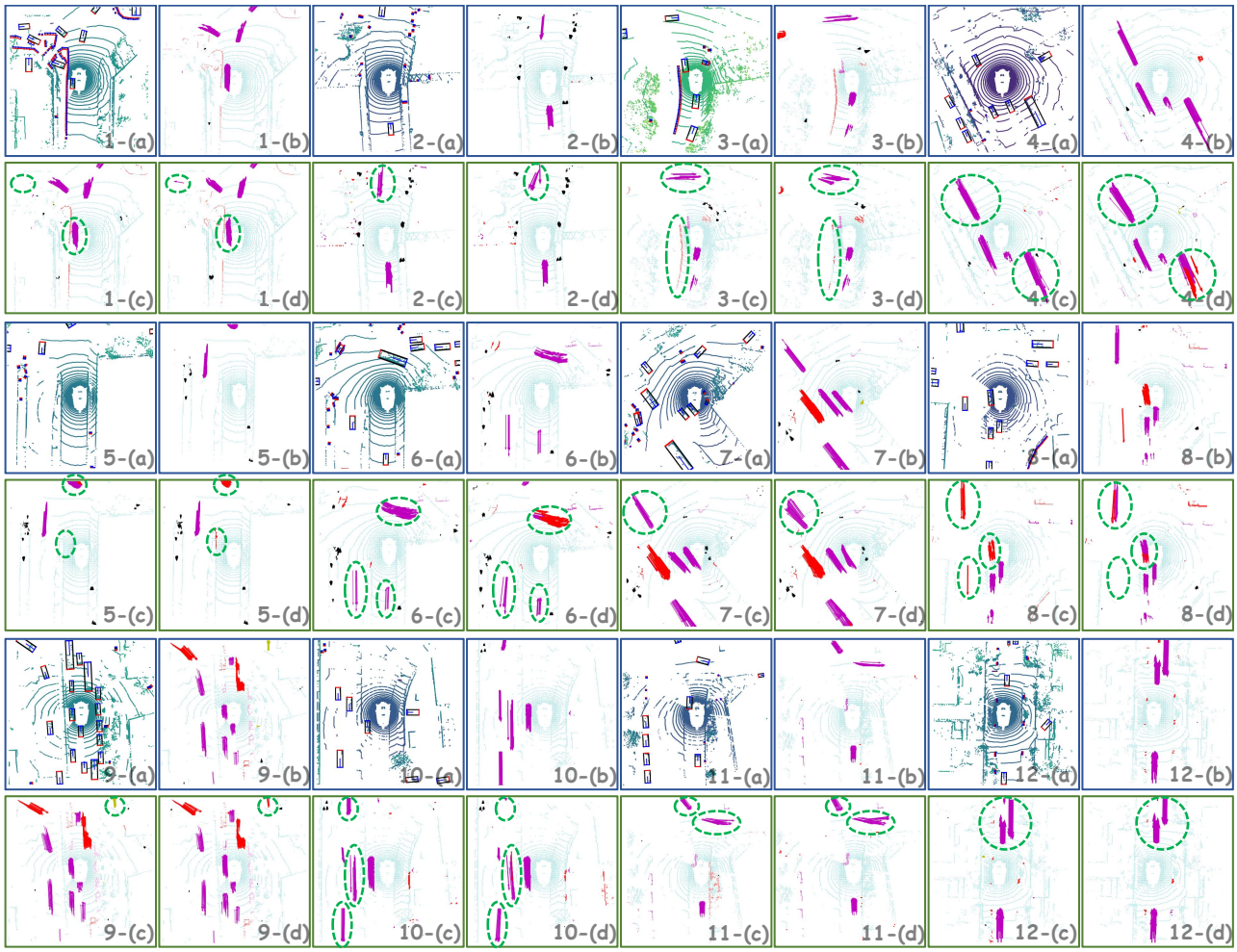

Figure A.3. Comparison of qualitative between results of the proposed PriorMotion and baseline model on **nuScenes**. **Blue border**: (a)object-level ground truth(GT) in BEV; (b)grid-level GT **Green border**: (c)PriorMotion predictions; (d)baseline model predictions. We represent the motions with an arrow attached to each grid. The cell classification result is represented by various colors. Cyan: background; pink: vehicle; black: pedestrian; yellow: bike; red: others.

$$Q = \text{Conv1x1}(\mathbf{X}) \in \mathbb{R}^{B\times H\times W\times C'}, \tag{A.18}$$

$$K = \text{Conv1x1}(\mathbf{X}) \in \mathbb{R}^{B\times H\times W\times C'}, \tag{A.19}$$

$$V = \text{Conv1x1}(\mathbf{X}) \in \mathbb{R}^{B\times H\times W\times C'}, \tag{A.20}$$

$$Q' = \text{reshape}(Q) \in \mathbb{R}^{B\times HW\times C'}, \tag{A.21}$$

$$K' = \text{reshape}(K) \in \mathbb{R}^{B\times HW\times C'}, \tag{A.22}$$

$$V' = \text{reshape}(V) \in \mathbb{R}^{B\times HW\times C'}, \tag{A.23}$$

$$\mathcal{A} = \text{softmax}\left(\frac{Q'K'^{\top}}{\sqrt{C'}}\right)V' \in \mathbb{R}^{B\times HW\times C'}, \tag{A.24}$$

$$\mathbf{Y} = \text{reshape}(\mathcal{A}) \in \mathbb{R}^{B\times H\times W\times C'}. \tag{A.25}$$

1.2. DSpG

VAE network. The Variational Autoencoder (VAE) network is designed to capture the latent space representation of the input data, which is crucial for generating realistic and diverse motion predictions. The VAE consists of three convolutional layers, each followed by batch normalization and a leaky ReLU activation function. The detailed architecture is as shown in Fig.A.2:

Multi-Task Feature Selection Decoder. The SE attention mechanism is defined as:

$$\text{FSD}(\mathbf{F}) = \sigma\left(\mathbf{W} f_{GAP}(\mathbf{F})\right) \cdot \mathbf{F} \tag{A.26}$$

where $\mathbf{F}$ represents the input features, $\mathbf{W}$ denotes a linear transformation matrix, f_{GAP} represents global average pooling, and σ is the sigmoid activation function. This mechanism ensures that the network efficiently allocates its focus to task-relevant information, optimizing performance for diverse tasks.

2. More Details about Loss Function Designs

We present more detailed loss function designs here.

Motion Prediction Loss. To accurately predict the future positions of objects, we employ a weighted smooth L1 loss. This loss ensures that the displacement of each non-empty grid cell is correctly estimated. The motion prediction loss is defined as:

$$L_{\text{mot}} = \frac{1}{N}\sum_{i=1}^{N} w_i \cdot \text{SmoothL1}(x_{\text{mot},i}, x_{\text{gt mot},i}) \quad \text{(A.27)}$$

where $x_{\text{mot},i}$ represents the predicted displacement for the i-th cell, $x_{\text{gt mot},i}$ is the corresponding ground truth, N is the total number of non-empty cells, and w_i balances the representation of different categories by assigning a weight to the i-th cell.

State Estimation Loss. To distinguish between dynamic and static elements in the scene, we use a cross-entropy loss for state estimation. This loss predicts whether each cell is in motion or stationary:

$$L_{\text{state}} = \frac{1}{N}\sum_{i=1}^{N} w_i \cdot \text{CE}(x_{\text{state},i}, x_{\text{gt state},i}) \quad \text{(A.28)}$$

where $x_{\text{state},i}$ is the predicted motion state of the i-th cell, and $x_{\text{gt state},i}$ is the ground truth. The cross-entropy function CE evaluates the prediction error.

Cell Classification Loss. For semantic understanding of each grid cell, a cross-entropy loss is used to classify cells into predefined categories. This classification helps the network interpret the scene at a higher semantic level:

$$L_{\text{cls}} = \frac{1}{N}\sum_{i=1}^{N} w_i \cdot \text{CE}(x_{\text{cls},i}, x_{\text{gt cls},i}) \quad \text{(A.29)}$$

where $x_{\text{cls},i}$ is the predicted class of the i-th cell, and $x_{\text{gt cls},i}$ is the ground truth class label.

3. More Details about Experimental Setting

FMCW LiDAR Benchmark. We conduct experiments on our **private FMCW LiDAR benchmark**, a collected dataset specifically designed for evaluating motion prediction in autonomous driving scenarios. The benchmark features a 128-degree forward-facing FMCW LiDAR sensor, capturing data at 10Hz. It includes 250 scenes, divided into 150 scenes for training, 50 for validation, and 50 for testing. The duration of each scene varies depending on the driving context. Each LiDAR frame is annotated with ground truth bounding boxes, providing high-quality supervision for motion prediction tasks.

Evaluation metrics. To comprehensively evaluate our model's performance, we follow the evaluation protocol established in [47] and divide the non-empty cells into three groups based on their speeds: static ($\mathbf{speed} \leq 0.2$ m/s), slow (0.2 m/s $< \mathbf{speed} \leq 5$ m/s), and fast ($\mathbf{speed} > 5$ m/s). For each group, we report the mean and median prediction error, which is calculated as the L2 distance between the predicted displacements and the ground truth displacements 1 second into the future. The mean prediction error for a group G is given by:

$$\text{Mean Error}_G = \frac{1}{|G|}\sum_{i\in G} \|\hat{\mathbf{d}}_i - \mathbf{d}_i\|_2 \quad \text{(A.30)}$$

where $\hat{\mathbf{d}}_i$ is the predicted displacement and $\mathbf{d}_i$ is the ground truth displacement for cell i.

The median prediction error for a group G is given by:

$$\text{Median Error}_G = \text{median}\left(\left\{\|\hat{\mathbf{d}}_i - \mathbf{d}_i\|_2 \mid i \in G\right\}\right) \quad \text{(A.31)}$$

In addition to the motion prediction error, we also evaluate the performance on auxiliary cell classification tasks. We report the overall accuracy (OA), which is the average accuracy over all non-empty cells:

$$\text{OA} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\hat{y}_i = y_i) \quad \text{(A.32)}$$

where N is the total number of non-empty cells, $\hat{y}_i$ is the predicted class, and y_i is the ground truth class for cell i.

We also report the mean category accuracy (MCA), which is the average accuracy over all five categories:

$$\text{MCA} = \frac{1}{C}\sum_{c=1}^{C} \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c} \quad \text{(A.33)}$$

where C is the number of categories, TP_c is the number of true positives for category c, and FN_c is the number of false negatives for category c.

4. More Qualitative Results on Nuscenes

More qualitative results are shown in Fig. A.3. Our PriorMotion framework is able to accurately predict motion across diverse object categories, and dramatically improve the motion stability and prediction ability at distance region.

5. More Experiment

5.1. Different Latent Feature Size

Comparison of the latent feature size. We experiment with different Gaussian distributions, comparing 1×1, 16×16, 32×32, and 64×64 scales, as shown in Tab. A.1. Our results show that increasing the scales can enhance the

Table A.2. Performance Comparison of VAE and GAN in PriorMotion. The table shows the mean error, median error, and variance for static objects, moving objects with speed $\leq$ 5m/s, moving objects with speed $>$ 5m/s, and overall motion stability.

Model	Static		Speed $\leq$ 5m/s		Speed $>$ 5m/s		Motion Stability
	Mean↓	Median↓	Mean↓	Median↓	Mean↓	Median↓	Variance
VAE	0.0281	0	0.2493	0.0921	**0.9141**	**0.6092**	**0.01525**
GAN	**0.0262**	**0**	**0.2377**	**0.0796**	0.9596	0.6174	0.01986

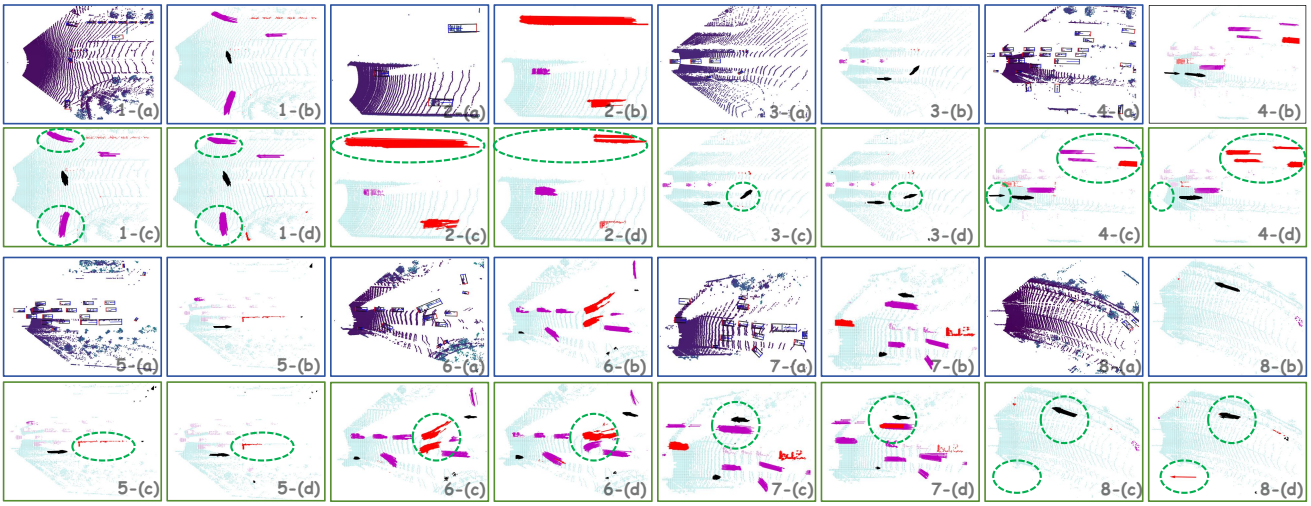

Figure A.4. Comparison of qualitative between results of the proposed PriorMotion and baseline model on **FMCW LiDAR**. **Blue border**: (a)object-level ground truth(GT) in BEV; (b)grid-level GT **Green border**: PriorMotion predictions (c); (d)baseline model predictions. We represent the motions with an arrow attached to each grid. The cell classification result is represented by various colors. Cyan: background; pink: vehicle; black: pedestrian; yellow: bike; red: others.

performance of motion prediction. This improvement is attributed to the ability to model different regions with finer granularity, which captures more detailed spatial and temporal patterns. However, larger scales also introduce greater computational overhead, which may impact the efficiency of the model. Therefore, there is a trade-off between the performance gain and the increased computational cost.

5.2. Different Generative Model

Comparison of generative models. We also compare the effectiveness of different generative models in PriorMotion. Specifically, we evaluate the use of Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). Both models show improvements, but our generative model, which is based on a VAE, achieves the best performance. Notably, GANs, due to their dual-model architecture, have a more complex and larger overall structure, making them more challenging to train. The performance comparison of the two generative models is shown in Tab. A.2.

5.3. Performance on the FMCW LiDAR Dataset

Main Results. We compare our proposed PriorMotion framework with several state-of-the-art (SOTA) approaches on the motion prediction task using the FMCW LiDAR dataset. As shown in Tab. A.3, PriorMotion demonstrates significant performance improvements across various metrics, particularly in challenging scenarios.

Our framework achieves consistent improvements with different backbones, including STPN and STI. Specifically, with the STI backbone, PriorMotion achieves a 9.63% reduction in the mean prediction error for static objects and a 15.98% reduction for objects moving faster than 5m/s, compared to the BE-STI baseline. Furthermore, PriorMotion achieves the best performance in speed stability, reducing the error to 0.0607.

In addition to motion prediction, PriorMotion outperforms competing approaches in the cell classification task. It achieves the highest Mean Class Accuracy (MCA) of 75.2% when paired with the STI backbone, demonstrating its ability to effectively classify dynamic objects such as bicycles and pedestrians in complex traffic scenarios. These results highlight the robustness and versatility of our framework across different tasks and backbones.

Prediction performance at distant region. We evaluate the performance of PriorMotion across various distance

Table A.3. Comparison with State-of-the-Art Results on FMCW LiDAR benchmark.

Method	**Backbone**	Static	Speed≤5m/s	Speed > 5m/s	Motion Stability↓	Cell Classfication	
		Mean↓	Mean↓	Mean↓		MCA↑	OA↑
MotionNet	STPN	**0.0644**	0.5036	1.0654	0.1992	74.2	97.5
STPN /w (Ours)	STPN	0.0653	0.4344	0.8897	0.0722	74.6	97.2
STI	STI	0.0645	0.4457	0.9278	0.1017	74.3	**97.5**
STI /w (Ours)	STI	**0.0641**	**0.4028**	**0.7792**	**0.0607**	**75.2**	97.3

Table A.4. Comparison with SOTA methods on FMCW LiDAR benchmark on long-distance speed error metrics.

Method	Backbone	Static		Speed ≤ 5m/s		Speed > 5m/s	
		Mean↓	Median↓	Mean↓	Median↓	Mean↓	Median↓
MotionNet	STPN	0.05223	0	0.4914	0.3459	1.1371	0.5322
STPN /w (Ours)	STPN	0.06164	0	0.4830	0.2218	1.0904	**0.4207**
STI	STI	0.06302	0	0.5221	0.2395	1.1034	0.4671
STI /w (Ours)	STI	0.06448	**0**	**0.4678**	**0.2144**	**0.9135**	0.4590

ranges, with a particular emphasis on long distances ([20m, 64m]). Our method not only demonstrates significant improvements in reducing prediction errors at these longer ranges on nuScenes[4], as evidenced in Tab. 5, but also establishes its superiority on our proprietary FMCW LiDAR benchmark (shown in Tab. A.4). On this benchmark, PriorMotion consistently outperforms the baseline methods in long-distance motion prediction, underscoring the effectiveness of our approach.

For objects moving faster than 5m/s, PriorMotion reduces the mean error to 0.9135 and the median error to 0.4590 with the STI backbone, outperforming all baseline methods. This demonstrates its enhanced ability to handle challenging scenarios where point cloud data becomes sparse. Similarly, for slower-moving objects (speed $\leq$ 5m/s), PriorMotion achieves a mean error of 0.4678 and a median error of 0.2144, significantly improving motion prediction accuracy.

Traditional methods primarily optimize encoder designs but struggle with the sparsity of distant point clouds. In contrast, our generative model leverages prior knowledge from the motion field, effectively compensating for data sparsity at long ranges. These results highlight the robustness of PriorMotion in predicting distant object trajectories, ensuring reliable performance even in challenging environments.

Qualitative Results. As illustrated in Fig. A.4, **PriorMotion** demonstrates superior performance, particularly for fast-moving objects. Notably, scenarios such as turning at intersections show significant improvements. Our framework substantially reduces displacement prediction errors across moving objects. Additionally, our method proves especially promising when handling sparse point clouds, particularly in distant regions.

In real-world autonomous driving systems, accurate and reliable motion prediction is crucial for safe navigation. For instance, PriorMotion enhances the system's ability to anticipate the movements of vehicles, cyclists, and pedestrians at complex intersections, thereby improving decision-making processes. This capability is essential for avoiding collisions and ensuring smooth traffic flow. Moreover, by effectively managing sparse point clouds in distant regions, our methods contributes to more robust perception in challenging environments, such as highways or rural roads where sensor data may be limited. These advancements pave the way for safer and more efficient autonomous driving technologies.