MapPrior: Bird's-Eye View Perception with Generative Models
– Supplementary Material –

Abstract

In the supplementary material, we provide more qualitative results of BEV map segmentation and uncertainty quantification, as well as additional analysis on failure cases in Sec. 1. In addition, we provide model architecture details in Sec. 2. The video "MapPrior.mp4" briefly introduces our task, proposed solution, and compares our results with baseline models.

1. Additional Qualitative Results

1.1. Map Segmentation as Generation

We provide additional Qualitative results for BEV map segmentation in Fig. 1 2 3 4. The figures show that our model consistently generates results with better accuracy and realism than the baseline. For instance, the fourth row of Fig. 1 demonstrates our approach preserves a realistic layout of a complicated multi-way intersection, with straight and complete lane boundaries, well-structured cross-walks as well as complete sidewalks, while the baseline's estimate is significantly noisier. In addition, the first row of Fig. 2 demonstrates our approach can complete the vertical layout of the road at the intersection despite limited sensor input. The generated vertical layout also preserves a realistic layout with well-structured lanes, crosswalks, and sidewalks. For another example, the first row in Fig. 4 demonstrates our approach generates correct and regular spacing of the lane boundaries, as well as smoother boundaries of sidewalks and roads than the camera baseline.

In addition, we provide more diversity and uncertainty calibration results in Fig. 5 and 6. Fig. 5 demonstrates that MapPrior can generate multiple diverse results, which can be aggregated into an accurate uncertainty map. Fig. 6 shows that MapPrior's uncertainty map aligns better with the error map in different scenes.

1.2. Failure Cases

Despite the overall improved IoU score in the benchmark, MapPrior can have a lower IoU score than the baseline model in certain cases. For example, in Fig. 7, there is a complicated layout distant from the ego car. In such cases, predictive models tend to make empty predictions. In contrast, MapPrior makes a realistic generation of possible traffic layouts. However, this may lead to significant false positives and a lower IoU score than an empty layout prediction. The better-calibrated confidence score that MapPrior provides should help the downstream planner module be aware of the uncertain areas in this situation.

2. Model Architecture Details

Predictive Stage In the predictive stage, we are using BEVFusion[6], with an unchanged model architecture. BEVFusion uses Swin-T[5] as the image backbone. It then applies FPN to fuse multi-scale camera features to produce a feature map of 1/8 input size. It downsamples camera images to 256×704. For the LiDAR backbone, BEVFusion adopts VoxelNet, using a voxel size of 0.1m. As a final step, BEVFusion uses a convolutional segmentation head to predict the BEV map.

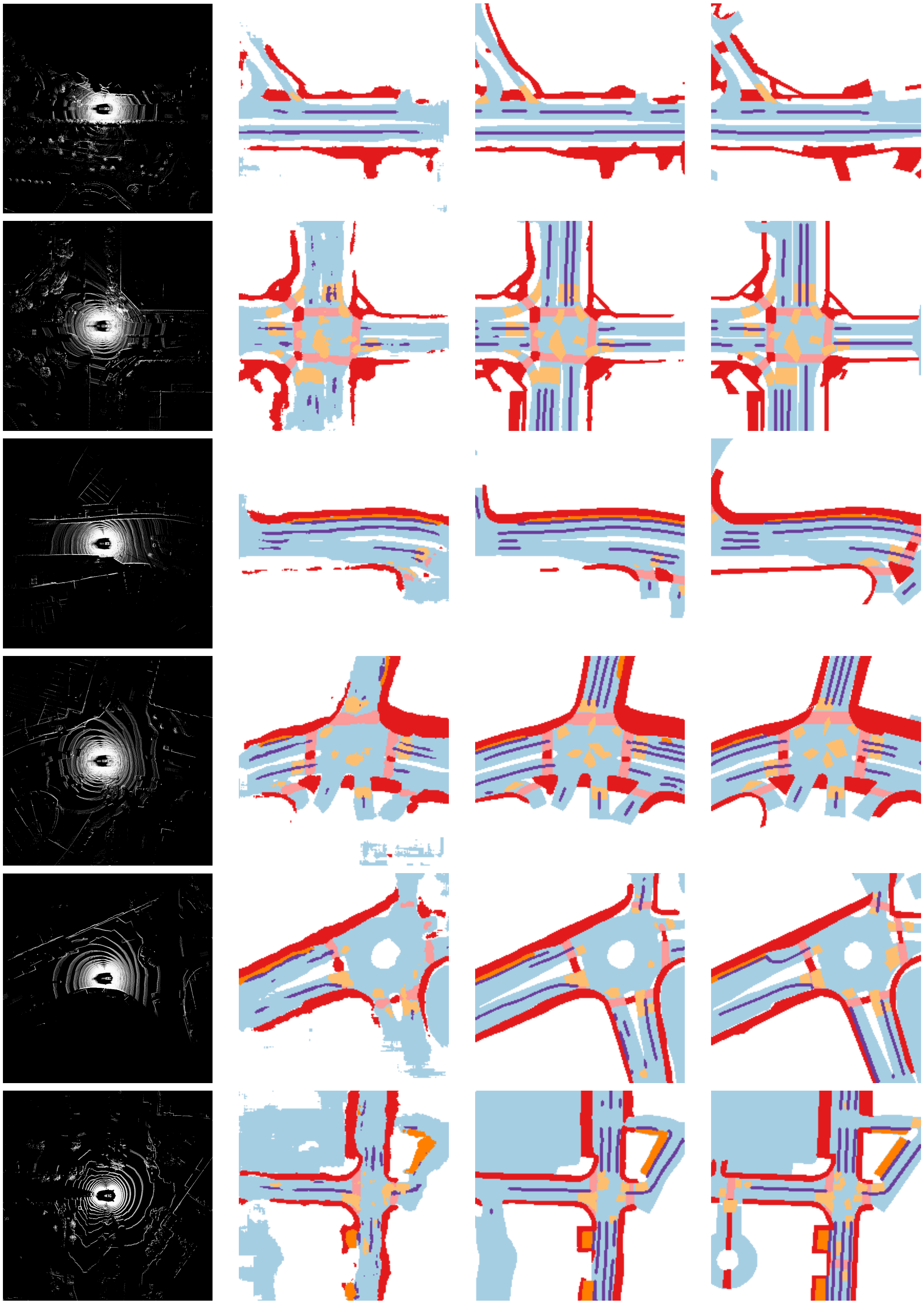
2.1. Auto-encoder

We modified the model architecture from VQGAN[1]. To make sure our input and output resolution of the BEV map is 200×200, and the latent space resolution is 12×12, we adjust the padding strategy in the auto-encoder model.

Encoder The encoder contains an input convolution layer followed by five downsampling blocks. Each downsampling block contains two res-net blocks. After downsampling, the encoder has two middle res-net blocks, one middle attention block, and one output convolution layer. We show detailed model architecture in Tab. 1

Codebook The codebook has 1024 different tokens, each of which is a 256-dimensional feature vector. Empirically we find this size trade-offs well between the generator's capacity and inference speed.

Decoder The decoder has a similar structure as the encoder. It contains an input convolution layer, two res-net blocks, and one attention layer before the upsampling blocks.



Input BEVFusion-L MapPrior-L (Ours) GT

Figure 1: Additional results of BEV map segmentation on nuScenes

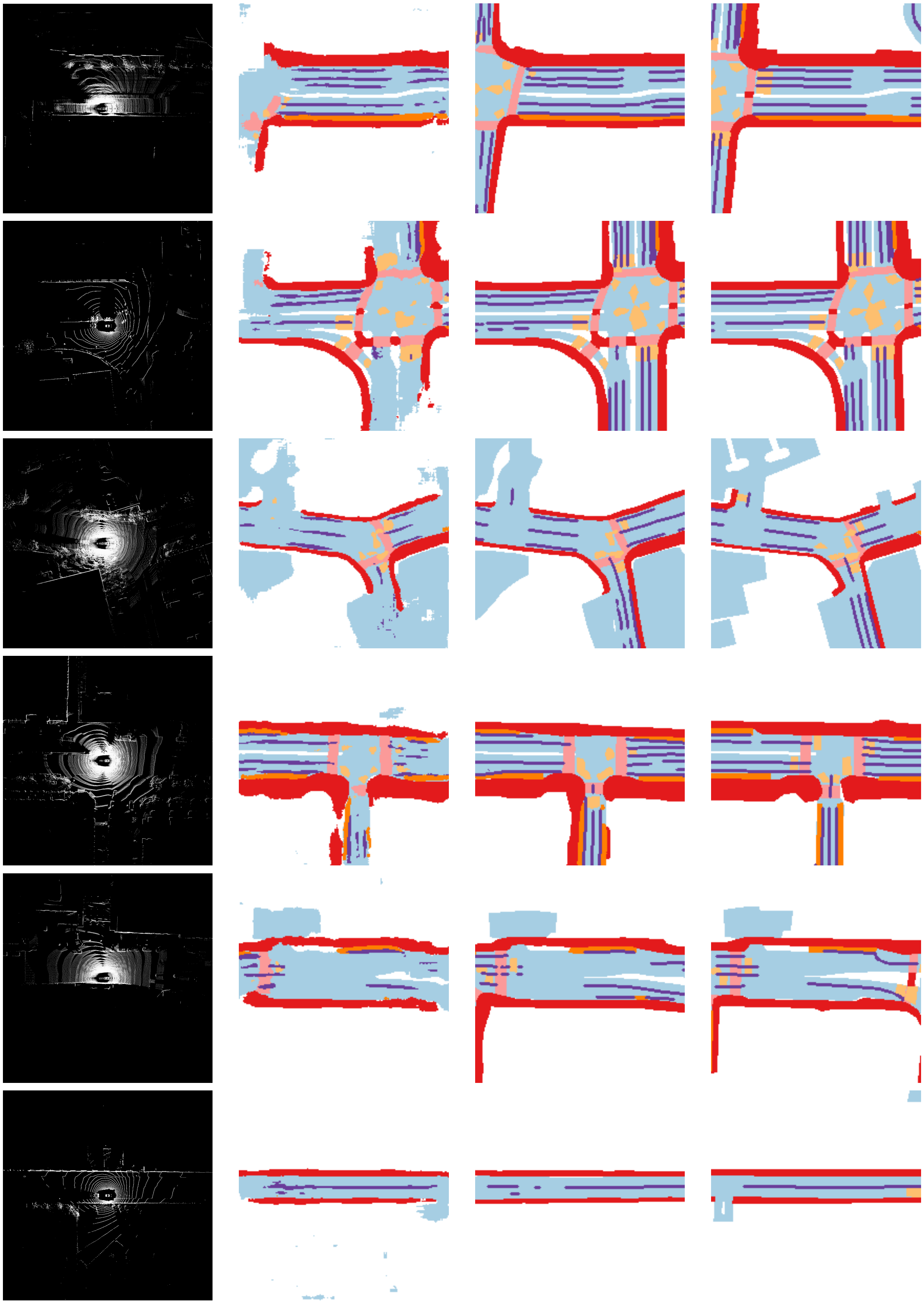


Figure 2: Additional Lidar results of BEV map segmentation on nuScenes, continued

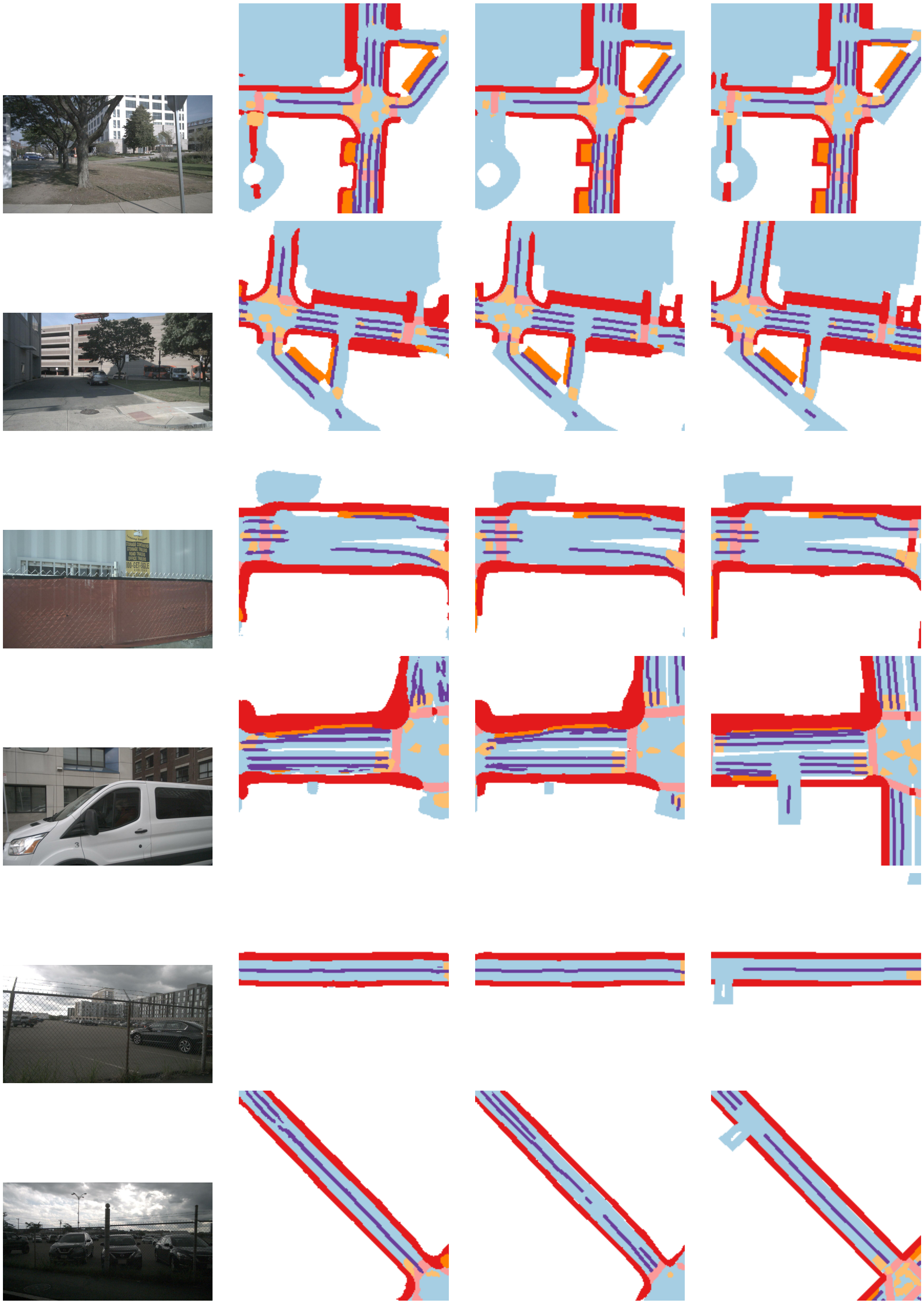


Figure 3: Additional Camera results of BEV map segmentation on nuScenes

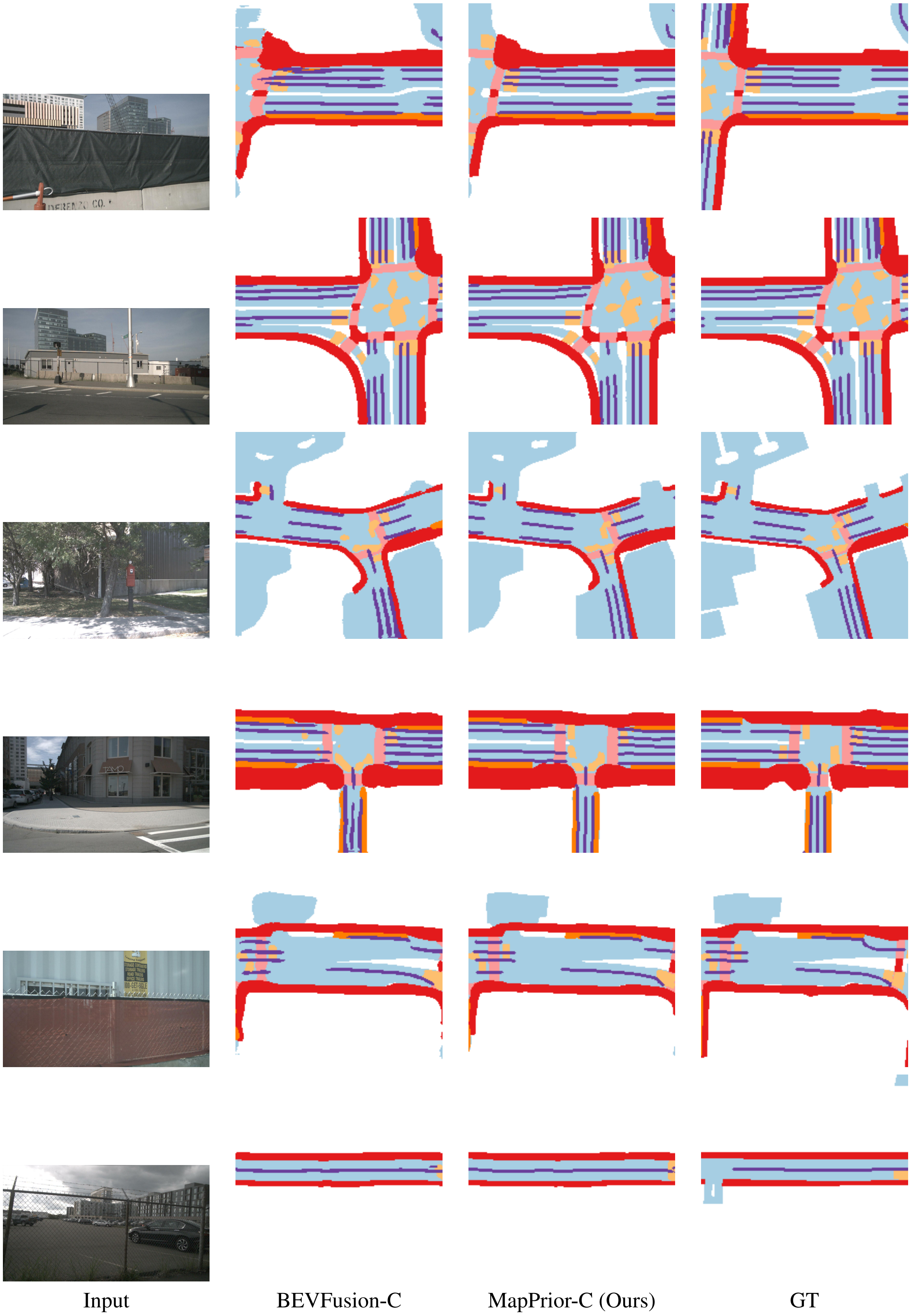


Figure 4: Additional Camera results of BEV map segmentation on nuScenes, continued

Table 1: Architecture for Encoder

layers		parameters
input	Conv2d	in_ch:6, out_ch: 128, kernel: 3x3, stride: 1, pad: 1
downsample_block_1	ResnetBlock	in_ch:128, out_ch: 128
	ResnetBlock	in_ch:128, out_ch: 128
	Downsample(Conv2d)	in_ch:128, out_ch: 128,kernel:3x3, stride:2, padding=((0,1,0,1),val=0)
downsample_block_2	ResnetBlock	in_ch:128, out_ch: 128
	ResnetBlock	in_ch:128, out_ch: 128
	Downsample(Conv2d)	in_ch:128, out_ch: 128,kernel:3x3, stride:2, padding=((0,1,0,1),val=0)
downsample_block_3	ResnetBlock	in_ch:128, out_ch: 256
	ResnetBlock	in_ch:256, out_ch: 256
	Downsample(Conv2d)	in_ch:256, out_ch: 256,kernel:3x3, stride:2, padding=((0,1,0,1),val=0)
downsample_block_4	ResnetBlock	in_ch:256, out_ch: 256
	ResnetBlock	in_ch:256, out_ch: 256
	Downsample(Conv2d)	in_ch:256, out_ch: 256,kernel:3x3, stride:2, padding=((0,1,0,1),val=0)
downsample_block_5	ResnetBlock	in_ch:256, out_ch: 512
	ResnetBlock	in_ch:256, out_ch: 256
middle	ResnetBlock	in_ch:512, out_ch: 512
	AttnBlock	in_ch:512
	ResnetBlock	in_ch:512, out_ch: 512
end	Normalize	GroupNorm,num_groups=32, num_channels=512
	Activation	x*sigmoid(x)
	Conv2d	in_ch:512, out_ch: 256, kernel: 3x3, stride: 1, pad: 1

It then includes five upsampling blocks, each containing three res-net blocks. Finally, the decoder applies an output convolution layer. We show the detailed model architecture in Tab. 2

Discriminator Following VQGAN, we define a 3-layer PatchGAN discriminator as in Pix2Pix [3]. The discrimination takes the reconstructed output or the ground truth BEV map as input and outputs a prediction map to discriminate the reconstructed map from ground truth. It includes one input convolution layer using leaky ReLU as activation. It then includes three convolution layers with batch norm as normalization and leaky ReLU as activation. It finally contains an output convolution layer to output a one-channel prediction map.

2.2. Latent Space Transformer

The latent space transformer consists of a GPT transformer and a BEV feature extractor. The GPT transformer contains token embedding, feature embedding, and positional embedding. The token embedding is a learnable feature vector for each code in the codebook, and the positional embedding is a learnable feature vector for every position. The feature embedding is the output of the BEV feature extractor. The final embedding is the token embedding concatenated with feature embedding and then encoded with a positional embedding. The final embedding is then fed into 24 attention blocks, and each consists of one vanilla multi-head masked self-attention layer with 16 heads, three mlp layers, and layer norm. As in GPT, a causal mask ensures that attention is only applied to the past in the input sequence.

We present the detailed parameter of GPT in Tab. 3. The BEV feature extractor shares a similar architecture as the encoder. We present the detailed architecture in Tab. 4.

3. Additional Quantitative Results

3.1. Expected calibration error clarification

We calculated the expected calibration error (ECE) following the method proposed in ”Verified Uncertainty Calibration” [4] (Eq. 1). In addition, to provide a comprehensive understanding, we also compute the original ECE score following Guo et al. [2]. Under this metric, our method, MapPrior-L still significantly outperforms the strongest baseline, with scores of 0.021 for MapPrior-L and 0.066 for BEVFusion-L.

References

[1] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 1

[2] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 8

[3] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 8

[4] Ananya Kumar, Percy Liang, and Tengyu Ma. Verified uncertainty calibration. In *NeurIPS*, 2019. 8

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*, 2021. 1

[6] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird’s-eye view representation. In *ICRA*, 2023. 1

Table 2: Architecture for Decoder

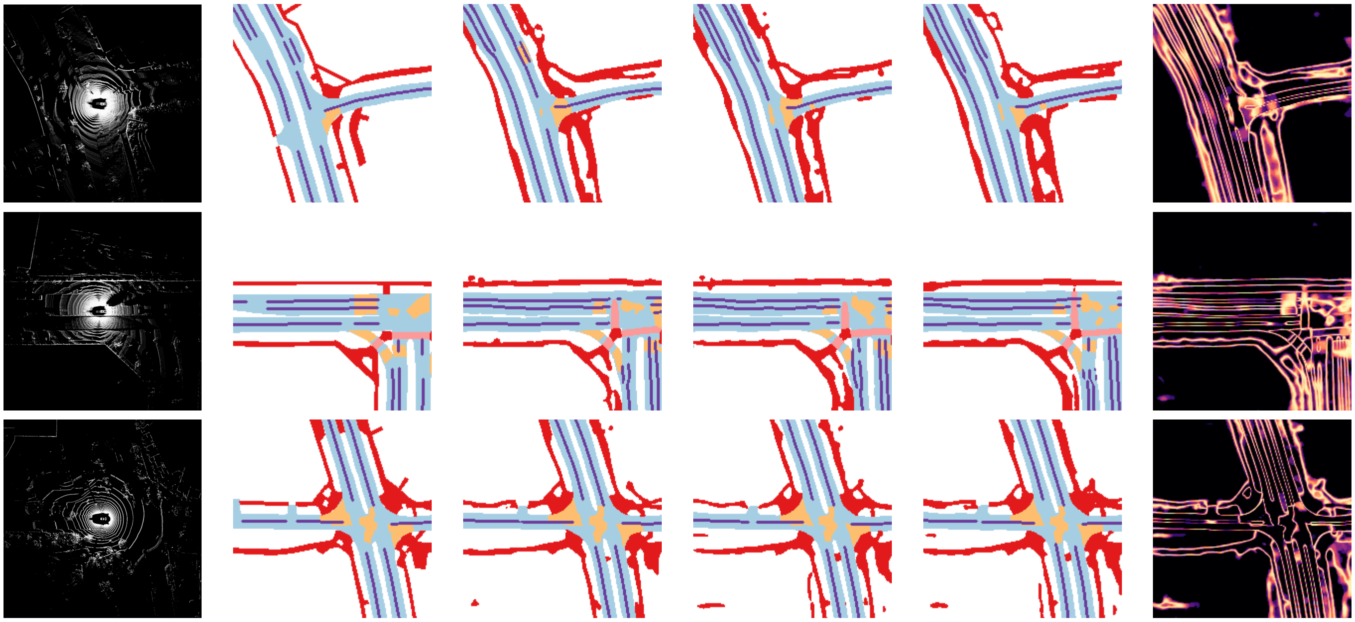
layers		parameters
input	Conv2d	in_ch:256, out_ch: 512, kernel: 3x3, stride: 1, pad: 1
middle	ResnetBlock	in_ch:512, out_ch: 512
	AttnBlock	in_ch:512
	ResnetBlock	in_ch:512, out_ch: 512
upsample_block_1	ResnetBlock	in_ch:512, out_ch: 256
	ResnetBlock	in_ch:256, out_ch: 256
	ResnetBlock	in_ch:256, out_ch: 256
upsample_block_2	ResnetBlock	in_ch:256, out_ch: 256
	ResnetBlock	in_ch:256, out_ch: 256
	ResnetBlock	in_ch:256, out_ch: 256
	Upsample(nearest_interpolate)	scale_factor=2.0
	Conv2d	in_ch:256, out_ch: 256,kernel=3x3,stride=1,padding=1)
upsample_block_3	ResnetBlock	in_ch:256, out_ch: 256
	ResnetBlock	in_ch:256, out_ch: 256
	ResnetBlock	in_ch:256, out_ch: 256
	Upsample(nearest_interpolate)	scale_factor=2.0
	Conv2d	in_ch:256, out_ch: 256,kernel=3x3,stride=1,padding=1)
upsample_block_4	ResnetBlock	in_ch:256, out_ch: 128
	ResnetBlock	in_ch:128, out_ch: 128
	ResnetBlock	in_ch:128, out_ch: 128
	Upsample(nearest_interpolate)	scale_factor=2.0
	Conv2d	in_ch:128, out_ch: 128,kernel=3x3,stride=1,padding=1)
upsample_block_5	ResnetBlock	in_ch:128, out_ch: 128
	ResnetBlock	in_ch:128, out_ch: 128
	ResnetBlock	in_ch:128, out_ch: 128
	ConvTranspose2d	in_ch:128, out_ch: 128, kennel:3x3,stride = 2
end	Normalize	GroupNorm,num_groups=32, num_channels=512
	Conv2d	in_ch:128, out_ch: 6, kernel: 3x3, stride: 1, pad: 1

Table 3: Architecture for Transformer

parameters	value
vocab_size	1024
block_size(context_size)	512
n_layer	24
n_head	16
n_embd	1024

Table 4: Architecture for BEV Feature Extarctor

layers		parameters
input	Conv2d	in_ch:80/256, out_ch: 128, kernel: 3x3, stride: 1, pad: 1
downsample_block_1	ResnetBlock	in_ch:128, out_ch: 128
	ResnetBlock	in_ch:128, out_ch: 128
	Downsample(Conv2d)	in_ch:128, out_ch: 128,kernel:3x3, stride:2, padding=((0,1,0,1),val=0)
downsample_block_2	ResnetBlock	in_ch:128, out_ch: 128
	ResnetBlock	in_ch:128, out_ch: 128
	Downsample(Conv2d)	in_ch:128, out_ch: 128,kernel:3x3, stride:2, padding=((0,1,0,1),val=0)
downsample_block_3	ResnetBlock	in_ch:128, out_ch: 256
	ResnetBlock	in_ch:256, out_ch: 256
	Downsample(Conv2d)	in_ch:256, out_ch: 256,kernel:3x3, stride:2, padding=((0,1,0,1),val=0)
downsample_block_4	ResnetBlock	in_ch:256, out_ch: 512
	ResnetBlock	in_ch:256, out_ch: 256
middle	ResnetBlock	in_ch:512, out_ch: 512
	AttnBlock	in_ch:512
	ResnetBlock	in_ch:512, out_ch: 512
end	Normalize	GroupNorm,num_groups=32, num_channels=512
	Activation	x*sigmoid(x)
	Conv2d	in_ch:512, out_ch: 256, kernel: 3x3, stride: 1, pad: 1



Input GT Our Sample 1 Our Sample 2 Our Sample 3 Uncertainty Map

Figure 5: Additional Qualitative results of diversity on NuScenes.

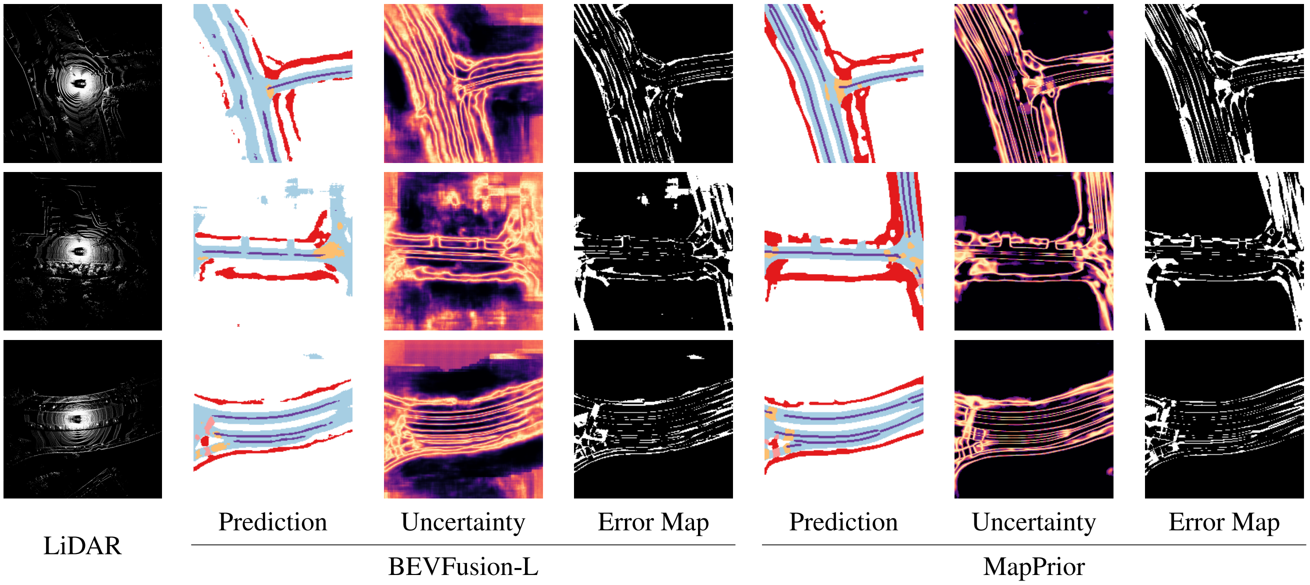


Figure 6: Additional results on uncertainty and error map comparison

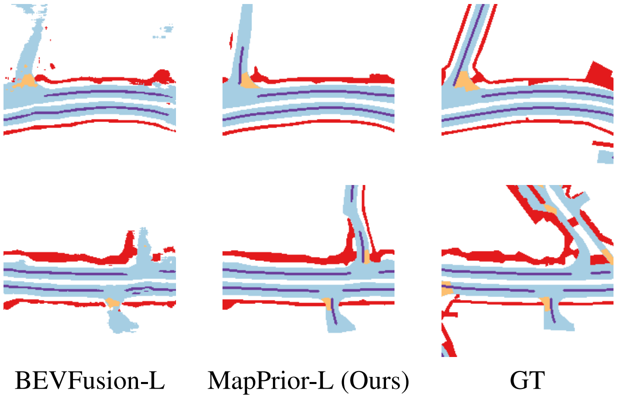


Figure 7: Failure Cases