From Binary to Semantic: Utilizing Large-Scale Binary Occupancy Data for 3D Semantic Occupancy Prediction

Supplementary Material

A. Implementation Details

A.1. 3D Representation

As discussed in Sec. 3.2, we employed a 3D compact representation rather than a BEV representation. The primary limitation of the BEV representation is the absence of the Z dimension, while it retains full resolution along the H and W dimensions (typically $H = 200$ and $W = 200$). To perform binary occupancy prediction from the BEV representation, the channel dimension C must be reshaped (e.g., $C = 512$ is typically reshaped as $Z \times C' = 16 \times 32 = 512$). However, this transformation presents two challenges for our approach.

First, the reduced channel dimension $C' = 32$ is insufficient for the subsequent sparse transformer, potentially creating a bottleneck in the model. Second, the full resolution of the 3D space (typically $H = 200$, $W = 200$, and $Z = 16$) is computationally prohibitive for real-time applications. Consequently, it is necessary to downsample the spatial resolution from $(H, W) = (200, 200)$ to $(H', W') = (100, 100)$, which may introduce an additional bottleneck in the model.

To mitigate these limitations, the proposed model adopts a 3D compact representation with $H^c = 50$, $W^c = 50$, $Z^c = 16$, and $C = 256$, thereby addressing the aforementioned concerns. The representation is then upsampled to a resolution of $H^b = 100$, $W^b = 100$, and $Z^c = 16$ for binary occupancy prediction. For auto-labeling experiments, we employed higher-resolution setting with $H^c = H^b = 256$, $W^c = W^b = 256$, $Z^c = Z^b = 40$, and $C = 64$, as the OpenOccupancy dataset provides a larger volume size ($H = 512$, $W = 512$, $Z = 40$), and the auto-labeling experiments consider only an offboard setting.

A.2. Architecture Details

For both LS-based and BEVFormer-based methods we employed a ResNet50 backbone with a FPN as the neck. The standard transformer in BEVFormer utilizes spatially dense queries and is therefore referred to as a dense transformer in the main text. In contrast, the proposed method incorporates a sparse transformer, which utilizes feature vectors corresponding to occupied regions as queries, as explained in Sec. 3.2. In both transformer architectures, we adopted the following hyperparameters: the number of attention heads is set to 8, the number of reference points for deformable attention is 4 in the self-attention module and 8 in the cross-attention module, and the embedding dimension is 256. For further implementation details, please refer to the provided code.

B. Evaluating Semantic Occupancy Prediction with Varying Numbers of Fine-tuning Scenes

In Section 4.1.1, we presented experimental results using 250 fine-tuning scenes. Figure 9 provides additional results with 500 and 1000 fine-tuning scenes. The proposed method consistently demonstrates a performance advantage across all fine-tuning set sizes.

C. Evaluating Binary Occupancy Prediction During Pre-training

Figure 10 provides the IoU scores of the binary occupancy prediction module within the proposed method as a function of the number of pre-training scenes. The IoU scores are highest upon the completion of pre-training but decrease during fine-tuning, as the number of fine-tuning scenes is substantially smaller than that of pre-training scenes. However, the proposed fine-tuning strategy effectively mitigates this performance degradation by incorporating the pre-training scenes during fine-tuning.

D. Performance Evaluation in Auto-labeling Pipelines

In Sec. 4.2, we evaluated the proposed auto-labeling model using the OpenOccupancy dataset. However, OpenOccupancy is not sufficiently large to evaluate auto-labeling pipelines. Therefore, in this section, we utilize the OpenScene dataset to generate a large-scale semantic occupancy dataset using the proposed pipeline and evaluate its effectiveness by training online methods on the generated dataset.

Table 4 presents a comparison of Sparse-LSS in an offboard setting—where GT binary occupancy is provided as input—versus an onboard setting. Consistent with the results on OpenOccupancy, the offboard model achieves significantly higher performance than its onboard counterparts.

Using the trained offboard model, we generated pseudo-labels. However, dense predictions such as semantic occupancy prediction require substantial storage capacity to save the generated labels. Ideally, storing logits for all voxels

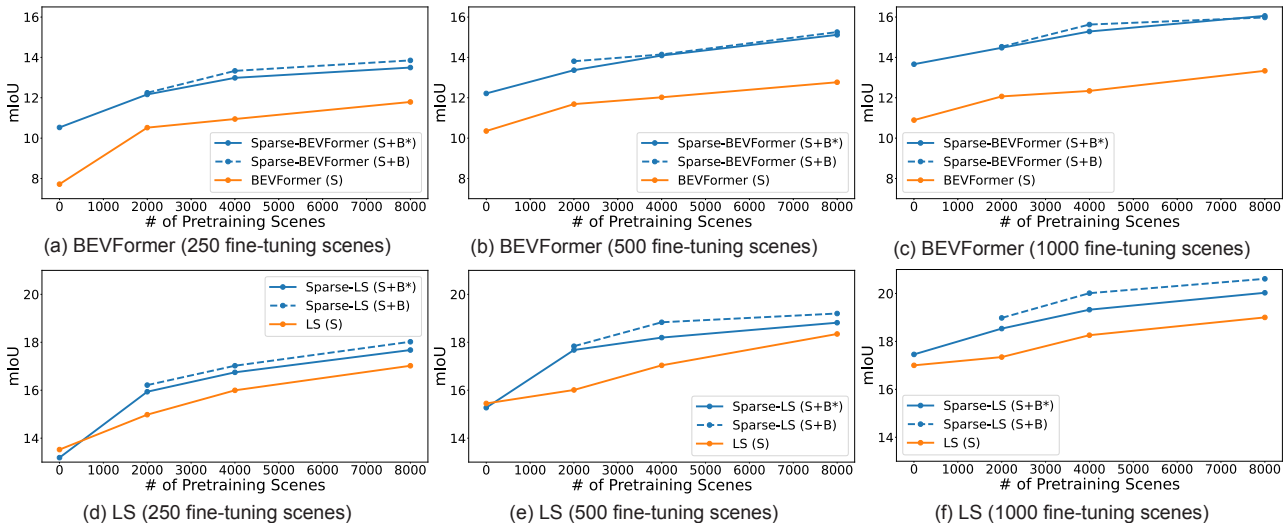

Figure 9. These figures display the mIoU scores as the number of pre-training scenes varies. The top row shows the results of BEVFormer-based models, while the bottom row presents the results of LS-based models.

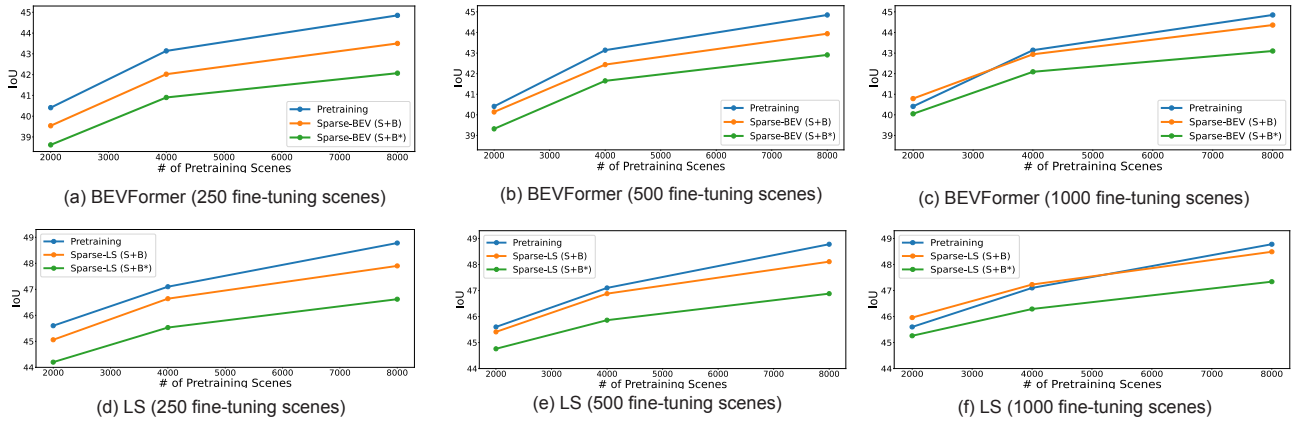

Figure 10. These figures display the IoU scores of binary occupancy prediction module with the proposed method as a function of the number of pre-training scenes.

would maximize performance, but this is impractical due to storage constraints. Therefore, in our experiments, we compared two types of pseudo-labeling approaches. The first is a storage-efficient approach, where only label indices corresponding to the highest predicted probability are stored (denoted as "Top1" in Tab. 5). The second is a balanced approach, where the top two logits are stored (denoted as "Top2" in Tab. 5).

As a baseline, we trained an LS-based model on 500 scenes with GT semantic occupancy labels. From this baseline, we progressively increased the number of scenes with pseudo-labels. As shown in Tab. 5, both IoU and mIoU improve as the number of pseudo-labeled scenes increases. Additionally, as expected, the Top2 strategy outperforms the Top1 strategy, demonstrating the benefits of retaining more predictive information. Notably, when 7,500 pseudo-labeled scenes are used, the model's performance becomes competitive with that of a model trained on 2,000 scenes with GT labels, highlighting the effectiveness of pseudo-labeling in reducing reliance on manually annotated data.

E. Performance Comparison with Longer Epochs

All experiments in Sec. 4.1 were conducted with 24 epochs. We here present experimental results with an extended fine-tuning duration of 64 epochs, during which the mIoU scores fully reach the plateau. As illustrated in Figure 11, our strategy outperforms the baseline by achieving higher plateau mIoU scores.

Method	Offboard	IoU	mIoU	Vechicle	C. Zone Sign	Bicycle	Generic Object	Pedestrian	Traffic Cone	Barrier	Background
Sparse-LS		43.92	19.20	36.88	0.23	8.00	22.92	19.49	10.20	12.94	42.94
Sparse-LS	✓	**73.89**	**33.17**	62.44	6.67	10.11	33.38	37.62	23.08	19.60	72.44

Table 4. Performance comparison between offboard and onboard models on the OpenScene dataset. The onboard model was pretrained on 8000 scenes and fine-tuned on 500 scenes. The offboard model was trained on the same 500 scenes dataset.

Method	# of Scenes w/ GT	# of Scenes w/ Pseudo-labels	Pseudo-labels	IoU	mIoU
LS	500	0	-	37.70	15.45
LS	2000	0	-	41.12	18.76
LS	500	1500	Top1	37.62	17.10
LS	500	1500	Top2	44.62	17.40
LS	500	3500	Top1	38.24	17.55
LS	500	3500	Top2	**46.09**	18.11
LS	500	7500	Top1	38.72	**18.22**

Table 5. Evaluation results of the auto-labeling pipelines using the proposed offboard model. Top1 denotes the auto-labeling approach in which only label indices corresponding to the highest predicted probability are stored, whereas Top2 represents the approach where the top two logits are stored.

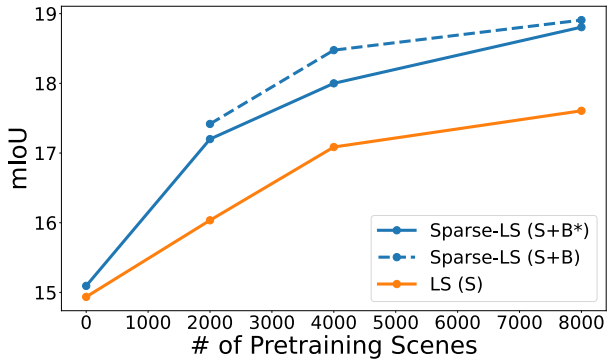

Figure 11. mIoU scores at 64 epoch as a function of the number of pre-training scenes. 250 scenes are used for the fine-tuning.

F. Qualitative Evaluation

Figures 12-16 present representative examples of the outputs generated by the proposed method, which are omitted from Sec. 4.1.4 in the main text due to page limitations.

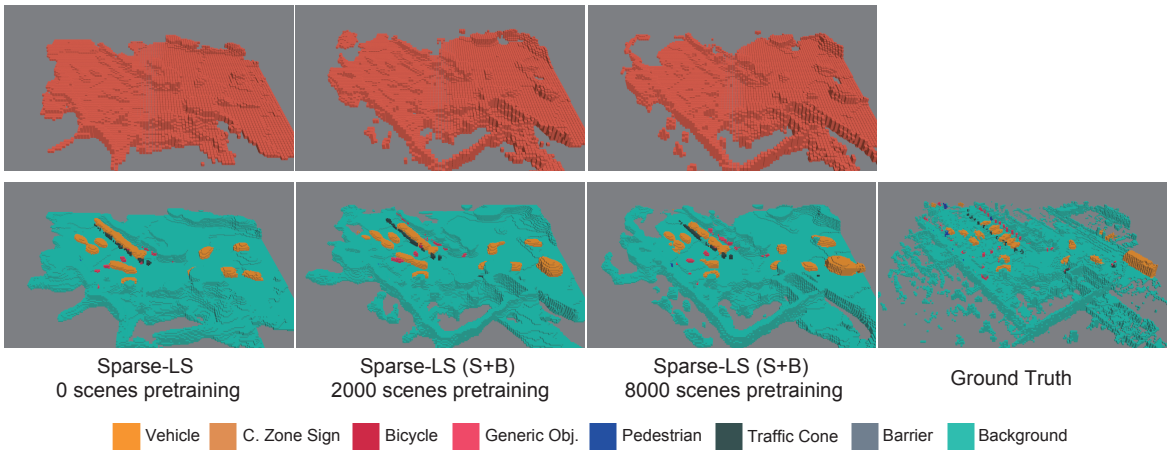

Figure 12. Qualitative comparison across different numbers of pre-training scenes: 0, 2000, and 8000. The top row presents the outputs of the binary occupancy modules, while the bottom row shows the results of semantic occupancy prediction.

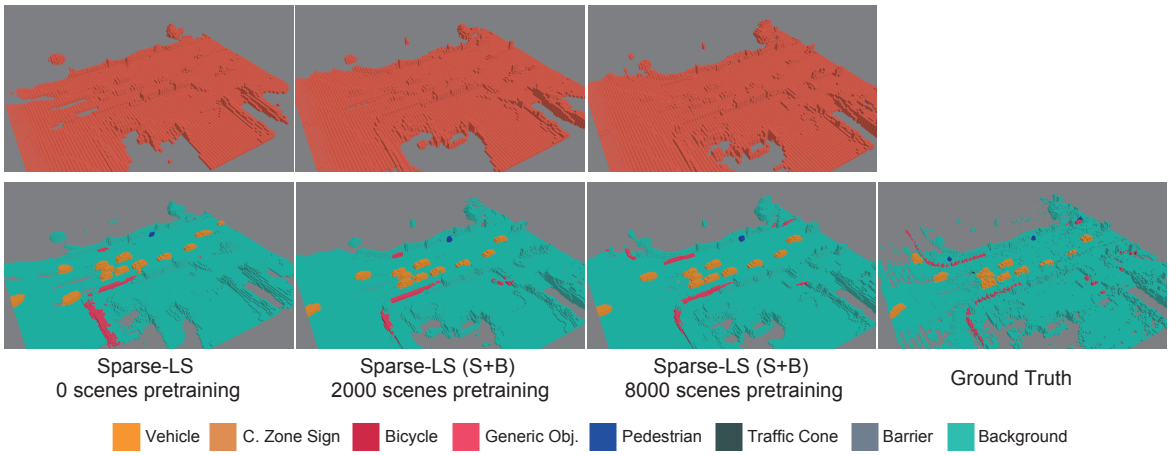

Figure 13. Refer to the caption of Figure 12 for a detailed description.

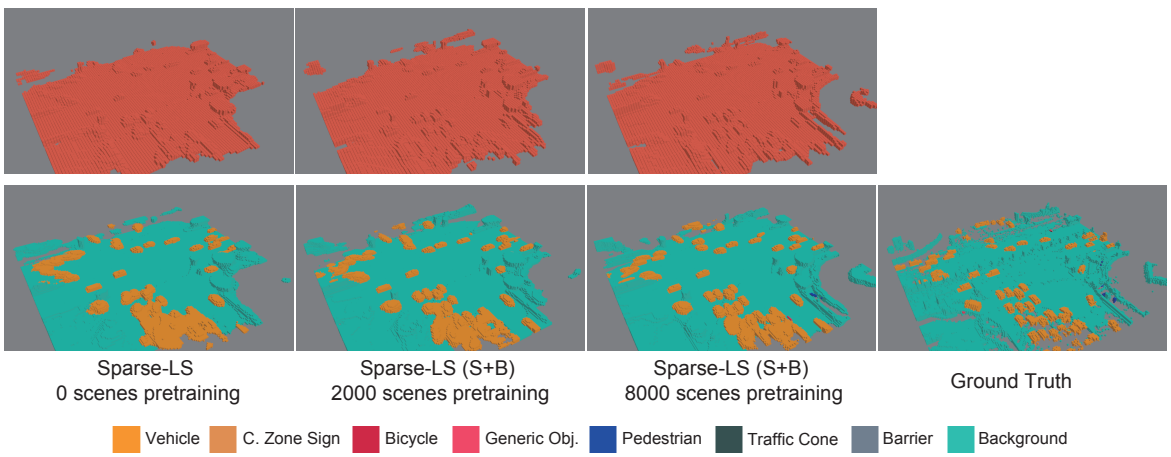

Figure 14. Refer to the caption of Figure 12 for a detailed description.

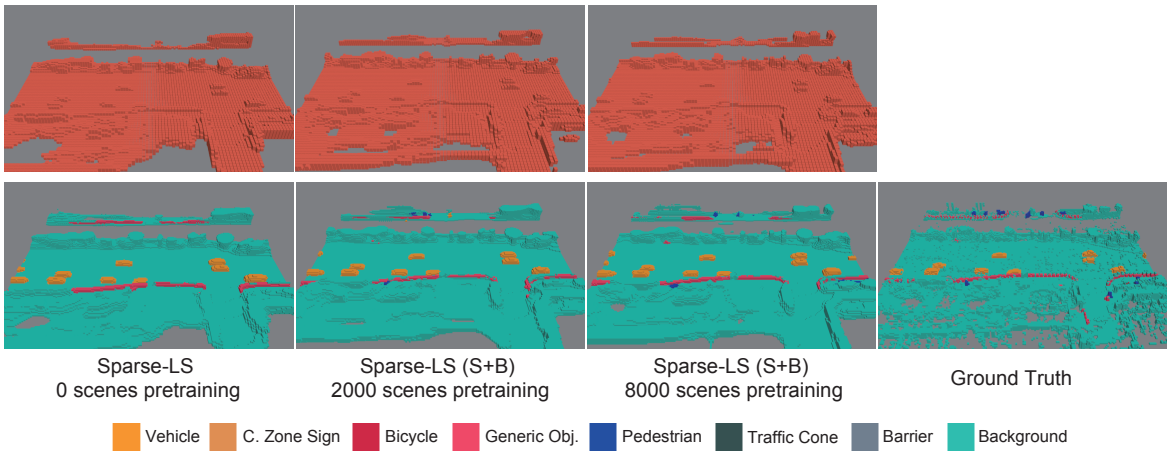

Figure 15. Refer to the caption of Figure 12 for a detailed description.

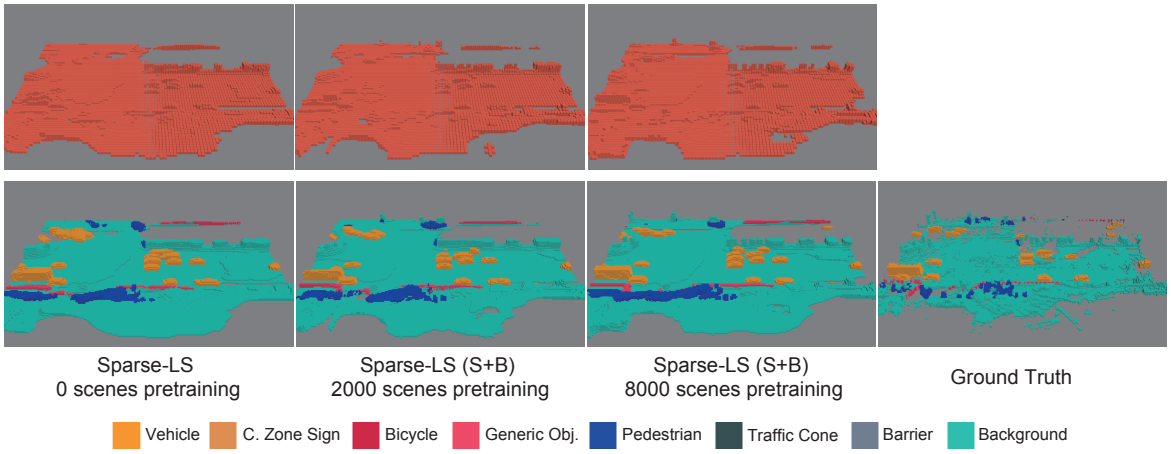

Figure 16. Refer to the caption of Figure 12 for a detailed description.