1. Model Robustness

Sparsity and noises are common in LiDAR point clouds. To evaluate the robustness of surface representation to sparse and noisy point clouds during training, we conduct several experiments on ScanObjectNN [7], a real-scene object classification dataset. We use overall accuracy (%) as the evalution metric. For extreme settings, we use 16, 32, 64, 128, and 256 points instead. Further, to evaluate model robustness against noises, we add 10% Gaussian noises to each point cloud. That is, we randomly select 10% the points from each point cloud and replace them with Gaussian noise.

Robustness to sparsity. As shown in Table 1, we observe that RealSurf can greatly improve the robustness of its backbone PointNet++ [4] by a large margin ($\sim$10%). At the same time, our Point Densification Module can further enhance its robustness by $\sim$1%. We conjecture this is because our Point Densification Module can densify point clouds implicitly, and thus lead to a more robust model. Overall, Uniform Point Densification Module works better than Gaussian Point Densification Module in this setting.

Robustness to noises. As shown in Table 1, the surface representation can improve model robustness in the 10% noise setting. Compared to baselines, it introduces a performance gain by 7%$\sim$14%. In this challenging case, our Point Densification Module can still improve the robustness of surface representation by 0.1%$\sim$1.5%.

Method	w/o noise					10% noise				
	16	32	64	128	256	16	32	64	128	256
PointNet++	56.32	58.40	64.19	69.40	74.50	51.91	55.31	58.40	66.79	73.59
baseline	60.96	68.88	74.91	78.83	79.91	59.06	66.11	72.24	76.06	79.08
w/ Gaussian	61.35	69.54	75.05	78.04	79.25	**59.61**	66.76	71.89	75.68	78.52
w/ Uniform	**61.83**	**69.88**	**75.64**	**78.94**	**81.58**	59.30	**66.86**	**73.21**	**76.79**	**79.24**

Table 1. Robustness of training on extremely sparse point clouds with or without noises from ScanObjectNN to simulate real-world point clouds. baseline: surface representation without Point Densification Module, Gaussian: our Gaussian Point Densification Module, Uniform: our Uniform Point Densification Module.

2. Class Balancing

Class imbalance can be vital for the training of point-based methods on LiDAR segmentation. To handle this problem, we adopt point cloud mixing inspired by [3]. Given two point clouds $\mathcal{S}_1$ and $\mathcal{S}_2$, the output $\mathcal{S}_{final}$ of the mixed point cloud is as follows:

$$\mathcal{S}_{final} = \left[Augment(\mathcal{S}_1), Augment(\mathcal{S}_2)\right], \tag{1}$$

where $[\cdot, \cdot]$ means the operation of concatenation, and $Augment$ means augmentation, i.e., Random Rotation (z-axis aligned, range: $[-\frac{\pi}{4}, \frac{\pi}{4}]$) $\rightarrow$ Random Flip (prob: 0.5). Different from [3], we do not apply random sub-sampling, random scaling, or random rotation along the other axes.

3. Positive/Negative Sample Balancing

As mentioned in the paper, point-based method usually requires positive/negative sample balancing. To that end, we adopt online hard example mining (OHEM) [5] in our pipeline. If the probability of the predicted class is lower than a threshold, we think we need to keep this sample for learning. In addition, this requires a minimum ratio to keep samples for learning. That is, if the probabilities of the predicted class for most samples are above the threshold, we need to keep that ratio of samples for learning. We set the threshold to 0.7. In order to obtain the hyperparameter of minimal kept ratio, we set it as twice the ratio of foreground points in each dataset. That is, 0.01, 0.005, and 0.001 for SemanticKITTI [1], nuScenes [2], and Waymo [6], respectively.

4. Efficient FPS

The low efficiency of FPS makes point-based methods less competitive compared to voxel-based methods. To alleviate this problem, we propose Sectorized FPS to speed up FPS. Sectorized FPS saves 30%$\sim$40% training time with almost no performance loss. A PyTorch-style Pseudocode of the implementation of Sectorized FPS is shown in Algorithm 1. As shown in Figure 1, we provide an example to show the difference between the results of vanilla FPS and those of Sectorized FPS. To balance the performance and efficiency, we set the hyperparameter of the number of sectors to 12 in all experiments. Besides,

we perform Sectorized FPS only in the first and second stage. For other stages, we perform vanilla FPS instead. We do not apply Sectorized FPS during inference.

Algorithm 1 PyTorch-Style Pseudocode of Sectorized FPS

```
# xyz: coordinates of a point set
# num_sector: number of sectors
angle = atan2(xyz[..., 0], xyz[..., 1])
sector_range = linspace(angle.min(), angle.max(), num_sector + 1)

# sectorized fps
new_xyz = []
for idx in range(num_sector):
   selected_idx = where((angle >= sector_range[s]) & (angle < sector_range[s + 1]))
   new_sector_xyz = farthest_point_sampling(xyz[selected_idx])
   new_xyz.append(new_sector_xyz)

out = cat(new_xyz, 0)
return out
```

5. Visualization

As shown in Figure 2, 3 and 4, we provide additional visualizations of the predictions by RealSurf compared with the ground-truth labels on the dataset of nuScenes [2], SemanticKITTI [1], and Waymo [6], respectively.

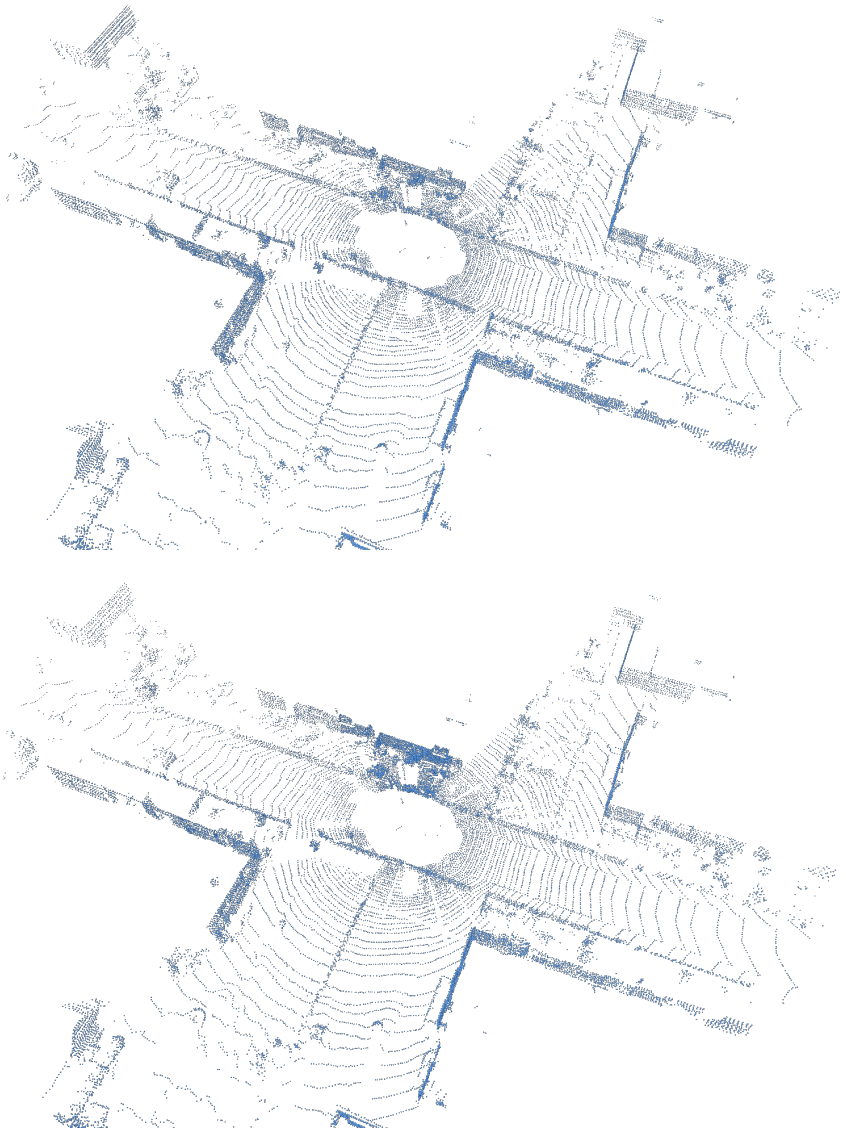

Figure 1. Comparison between vanilla FPS (top) and Sectorized FPS (bottom) in the first stage.

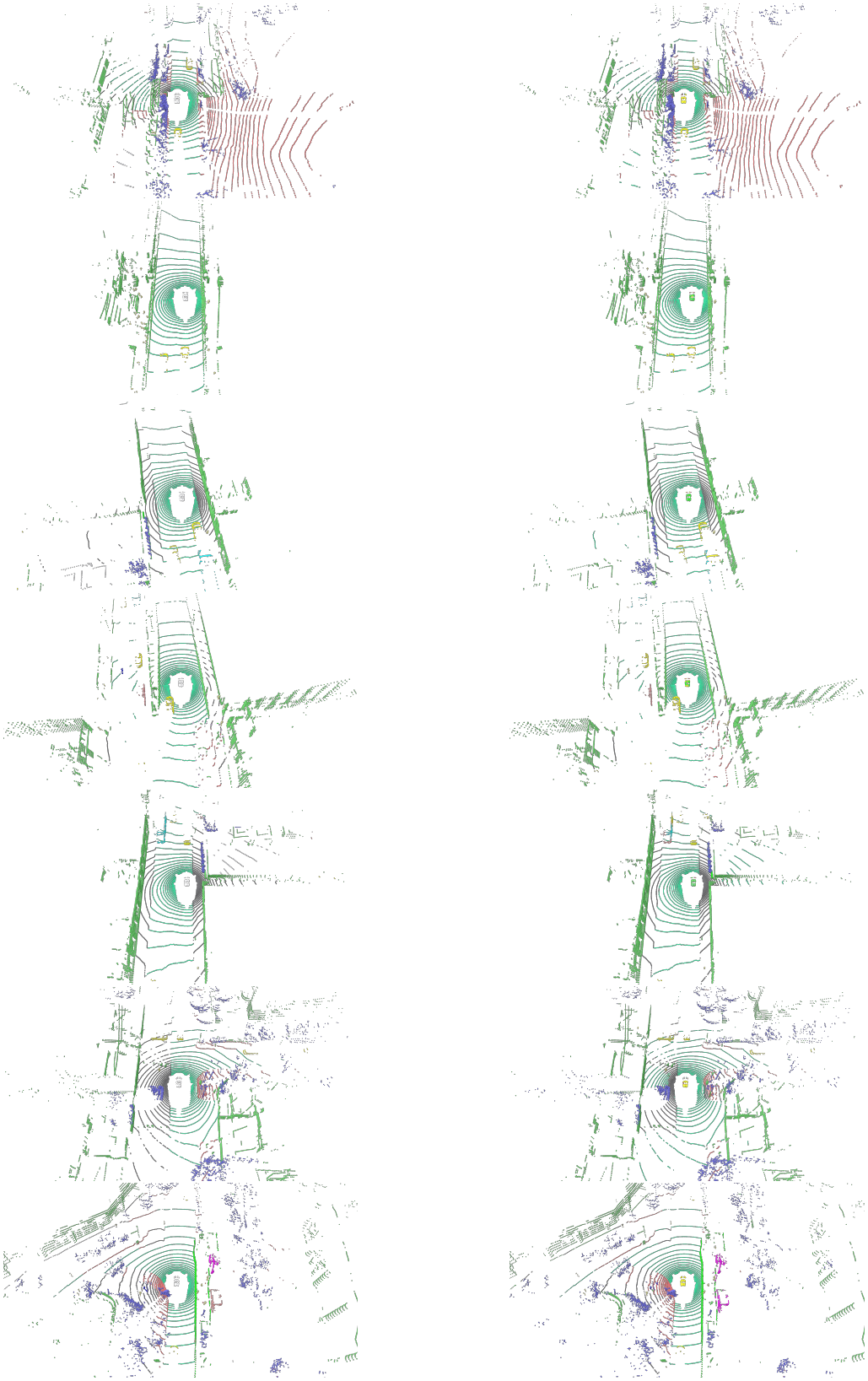

Figure 2. Visualization of ground-truth label (left) and our LiDAR segmentation results (right) on nuScenes [2].

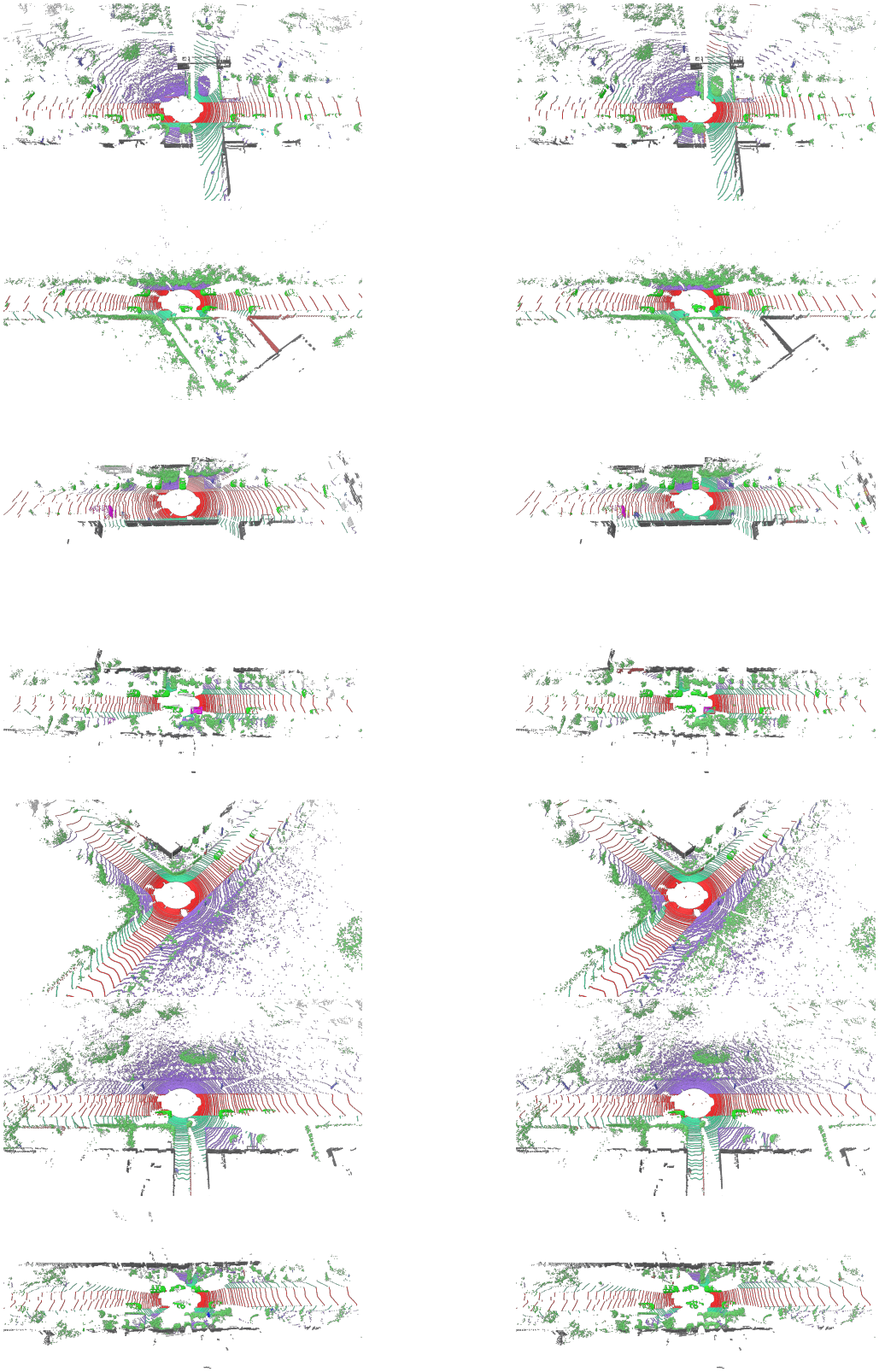

Figure 3. Visualization of ground-truth label (left) and our LiDAR segmentation results (right) on SemanticKITTI [1].

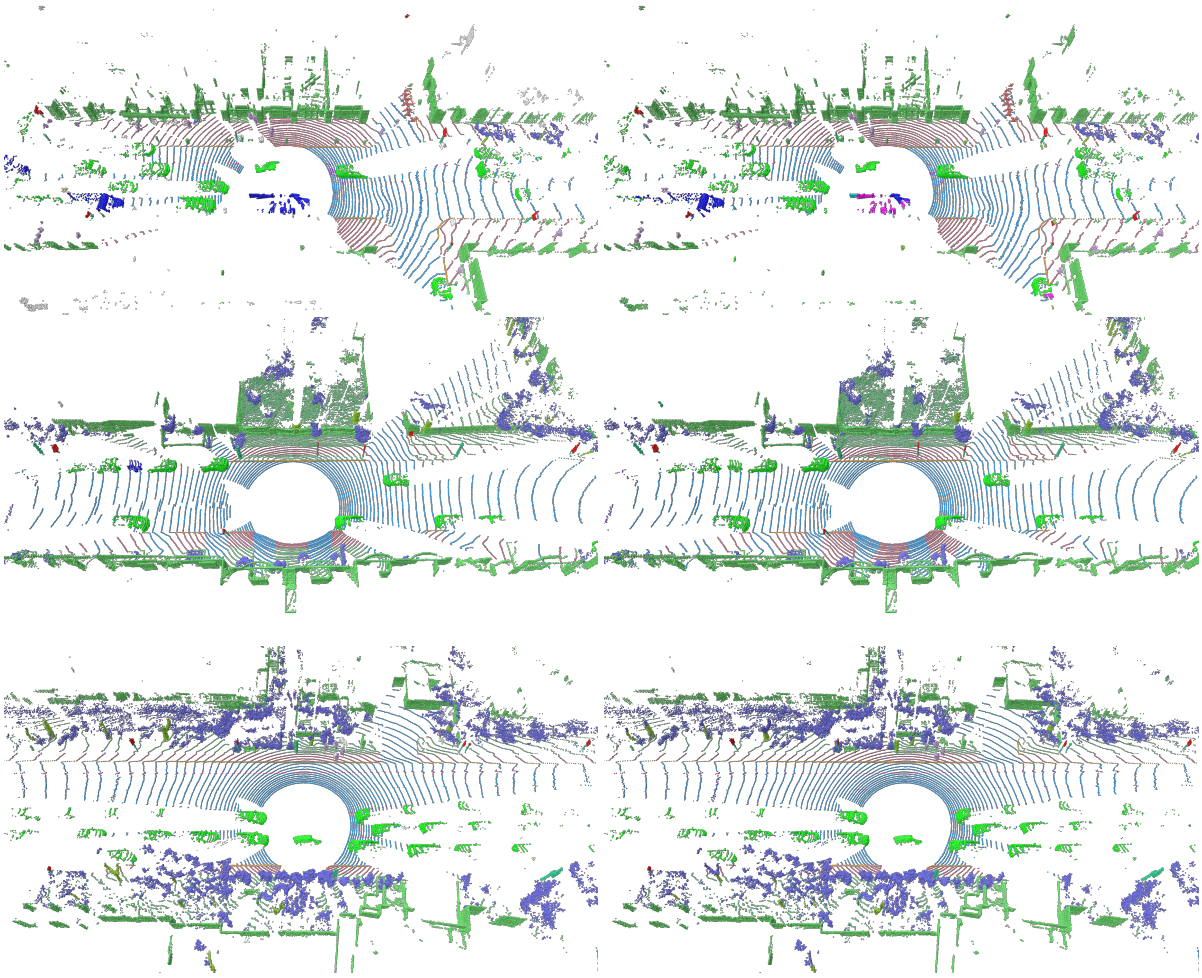

Figure 4. Visualization of ground-truth label (left) and our LiDAR segmentation results (right) on Waymo [6].

References

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9297–9307, 2019. 1, 2, 5

[2] Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom, and Abhinav Valada. Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. *IEEE Robotics and Automation Letters*, 7(2):3795–3802, 2022. 1, 2, 4

[3] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3d: Out-of-context data augmentation for 3d scenes. In *2021 International Conference on 3D Vision (3DV)*, pages 116–125. IEEE, 2021. 1

[4] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 1

[5] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 761–769, 2016. 1

[6] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 1, 2, 6

[7] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1588–1597, 2019. 1