LidarCLIP or: How I Learned to Talk to Point Clouds Supplementary Material

A. Model details

Tab. 1 shows the hyperparameters for the SST encoder [2] used for embedding point clouds in the CLIP space. Window shape refers to the number of voxels in each window. Other hyperparameters are used as is from the original implementation.

The encoded voxel features from SST are further pooled with a multi-head self-attention layer to extract a single feature vector. Specifically, the CLIP embedding is initialized as the mean of all features and then attends to said features. The pooling uses 8 attention heads, learned positional embeddings, and a feature dimension that matches the current CLIP model, meaning 768 for ViT-L/14 and 512 for ViT-B/32.

B. Training details

All LidarCLIP models are trained on the union of the *train* and *raw_large* splits, which are defined in the ONCE development kit [8]. The ViT-L/14 version is trained for 3 epochs, while ablations with ViT-B/32 used only 1 epoch for training. We use the Adam optimizer [7] with a base learning rate of 10^{-5}. The learning rate follows a one-cycle learning rate scheduler with a maximum learning rate of 10^{-3} and cosine annealing, and spends the first 10% of the training increasing the learning rate. The training was performed on four NVIDIA A100s with a batch size of 128, requiring about 27 hours for 3 epochs.

Parameter	Value	Unit
Voxel size	(0.5, 0.5, 6)	m
Window shape	(12, 12, 1)	-
Point cloud range	(0, -20, -2, 40, 20, 4)	m
No. encoder layers	4	-
d model	128	-
d feedforward	256	-

Table 1. Hyperparameters for SST encoder.

C. Baseline details

C.1. PointCLIP

PointCLIP [11] transfers CLIP's knowledge to 3D by applying the pre-trained CLIP image encoder to renderings of point clouds from multiple viewpoints and pooling the feature vectors. Hence, for zero-shot classification and retrieval, PointCLIP does not require any training. We adopt the official PointCLIP implementation[1] for rendering lidar point clouds to 2D depth maps and extracting features. For zero-shot classification, we follow PointCLIP and render objects from six different viewpoints and employ a "point cloud depth map of a [CLASS]." prompting scheme. We also evaluated the prompt ensembling used by LidarCLIP but found it to perform worse. For retrieval, we evaluate using six viewpoints and using only the front-facing one, but find them to perform comparably. For consistency, we use all six viewpoints.

C.2. CLIP2Point

CLIP2Point [5] uses a similar approach to PointCLIP, but applies a different rendering scheme, giving big performance improvements for zero-shot classification on real-world data such as ScanObjectNN. Further, opposite to PointCLIP, they propose to pre-train the CLIP image encoder on depth maps to get further improvements. Specifically, from ShapeNet, they create training samples consisting of one image and two depth maps at different distances. Images are passed through a frozen CLIP image encoder while the depth maps are passed through a CLIP image encoder that does not have frozen weights. The depth encoder is supervised using the NT-Xent loss from SimCLR [1], applied both between the two depth map samples (intra-modality) and between the averaged depth map features and the image features (inter-modality). For complete details, we refer to the original manuscript.

As CLIP2Point shares some similarities to our approach, we also apply their pre-training on ONCE data for a fair comparison. We train two versions, one ViT-B/32 (used in the original manuscript) and one ViT-L/14 (used for supervising LidarCLIP). Similar to the results in [11] we

[1] https://github.com/ZrrSkywalker/PointCLIP

find ViT-B/32 to perform best and hence report those results in the main manuscript. We use the same optimizer settings, learning rate and batch size as in the original manuscript. We train CLIP2Point on the same image-point cloud pairs as LidarCLIP and render two depth maps using the CLIP2Point rendering approach. For zero-shot classification, we render the objects from six viewpoints, while we find rendering only the front-view beneficial for retrieval.

C.3. Generative applications

We evaluate our generative applications, i.e., lidar-to-text and lidar-to-image, using roughly 6,000 randomly selected image-lidar pairs from the ONCE validation set. For the lidar-to-image generation, we use CLIP-guided Stable Diffusion[2] [10]. We set the number of inference steps to 250, guidance scale to 0.05, CLIP guidance scale to 1200, number of cutouts to 12. All other parameters are used as is. To ensure photorealistic results (in contrast to art or drawings) we add the prompt "a photorealistic image" when running lidar-to-image generation without captions.

For lidar-to-text we adopt the inference notebook provided by ClipCap[3] [9]. We use the COCO pre-trained model without beam search.

As our lidar-to-image baseline, we use the official pix2pix [6] PyTorch implementation[4]. We train a pix2pix model on 30,000 randomly selected image-lidar pairs from the ONCE train and large raw set. We also randomly select about 6,000 image-lidar pairs for validation during training. Note that these are different from the 6,000 pairs used for evaluation. We create depth images by projecting lidar points in to the image and setting pixel values to normalized depth. Pix2pix hyperparameters set to default values[5] and the training is run for 50 epochs with a batch size of 64.

D. Additional results

D.1. Zero-shot object classification

In Tab. 2 we evaluate additional backbones on the zero-shot classification task. We have included results from the main manuscript (Sec. 4.1.) for convenience. Similarly to the results of [11], ViT-B/32 performs best and is thus the backbone we report on in the main manuscript. We note that pre-training CLIP2Point [5] on ONCE does not improve performance for ViT-B/32. Although it improves ViT-L / 14, we note that the baseline performs worse than random guessing. We believe the pre-training to not be beneficial for zero-shot classification as the training data differ greatly from the inference setup. During training, the scenes cover large areas and vary in crowdedness. At inference, the model instead 'zooms in' on a single object. In contrast, pre-training with the synthetic ShapeNet data does not hurt performance, as it is object-centric.

	Backbone	Fine-tuned	Cls.	Obj.
PointCLIP [11]	ViT-B/32	-	29.1%	25.0%
PointCLIP [11]	ViT-L/14	-	18.1%	3.14%
PointCLIP [11]	RN50	-	28.2%	34.1%
PointCLIP [11]	RN101	-	22.2%	22.8%
CLIP2Point [5]	ViT-B/32	-	31.1%	26.2%
CLIP2Point [5]	ViT-L/14	-	16.0%	8.2%
CLIP2Point [5]	RN50	-	22.6%	28.3%
CLIP2Point [5]	RN101	-	16.9%	17.5%
CLIP2Point [5]	ViT-B/32	ShapeNet	29.8%	28.2%
CLIP2Point [5]	ViT-B/32	ONCE	21.4%	3.2%
CLIP2Point [5]	ViT-L/14	ONCE	32.2%	13.9%
Image	ViT-L/14	-	58.6%	67.1%
LidarCLIP (ours)	ViT-L/14	ONCE	43.6%	62.1%
Joint (ours)	ViT-L/14	(see above)	**60.8%**	**73.3%**

Table 2. Zero-shot classification on ONCE *val*, top-1 accuracy averaged over classes/object instances.

D.2. Retrieval

Here, we provide detailed retrieval results, as the main manuscript only contained averaged numbers. In Tab. 3, we show class-wise performance when querying specifically for objects close to the ego-vehicle. We find that lidar retrieval outperforms image retrieval for most classes, especially on Cyclists. Tab. 4, shows class-wise performance for objects in general. As described in the main manuscript, we hypothesize some classes to be more invariant than others in 3D than 2D. For instance, LidarCLIP sometimes confuses buses to be trucks, resulting in lower truck retrieval score. In Tab. 5, we show retrieval results for different scene conditions. As expected, LidarCLIP does not perform on par with the image encoder for these types of retrieval, but surprisingly still contains enough complementary information for the joint model to achieve the strongest performance.

Separate prompts. Fig. 1 shows additional results for the joint retrieval with separate prompts for the image and lidar encoder. Again, these scenes are close to impossible to retrieve using a single modality.

Prompt examples. We provide several examples of prompt templates in Tab. 6. The quality modifiers are `bad, good, clean, dirty, cropped, close-up`. The format modifiers are `photo`. The scene modifiers are `environment, scene, road, street, intersection`. The capture modifiers are `taken, captured`. For object classification we follow

[2] https://github.com/huggingface/diffusers/tree/main/examples/community

[3] https://github.com/rmokady/CLIP_prefix_caption

[4] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

[5] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/blob/9f8f61e5a375c2e01c5187d093ce9c2409f409b0/scripts/train_pix2pix.sh#L2

P@K	10	100	10	100	10	100	10	100	10	100	10	100
Object Cat.	Nearby Car		Nearby Truck		Nearby Bus		Nearby Ped.		Nearby Cyclist		Avg. Nearby	
PointCLIP	0.2	0.34	0.0	0.03	0.0	0.0	0.1	0.02	0.0	0.07	0.06	0.09
CLIP2Point	0.2	0.17	0.0	0.01	0.2	0.06	0.0	0.01	0.0	0.01	0.08	0.05
CLIP2Point†	0.8	0.66	0.0	0.04	0.1	0.26	0.5	0.24	0.4	0.36	0.36	0.31
Image	**1.0**	0.95	0.7	0.83	0.6	0.55	**1.0**	0.66	0.5	0.37	0.76	0.67
LidarCLIP	**1.0**	**0.98**	**0.8**	0.70	**1.0**	**0.90**	0.7	0.71	**0.9**	**0.61**	0.88	0.78
Joint	0.9	**0.98**	**0.8**	**0.92**	**1.0**	0.79	**1.0**	**0.84**	0.8	0.53	**0.90**	**0.81**

Table 3. Retrieval for various object categories, requiring the object to be close to the ego vehicle. We report precision at ranks 10 and 100. †ONCE fine-tuning.

P@K	10	100	10	100	10	100	10	100	10	100	10	100
Object Cat.	Car		Truck		Bus		Pedestrian		Cyclist		Avg.	
PointCLIP	0.6	0.66	0.1	0.13	0.0	0.03	0.4	0.35	0.4	0.29	0.3	0.29
CLIP2Point	0.6	0.60	0.1	0.11	0.1	0.12	0.1	0.20	0.3	0.19	0.24	0.24
CLIP2Point†	**1.0**	0.84	0.0	0.16	0.6	0.64	0.9	0.69	0.4	**0.65**	0.58	0.60
Image	**1.0**	0.96	0.9	**0.94**	0.9	0.94	0.9	0.83	0.5	0.37	0.84	0.81
LidarCLIP	**1.0**	**0.99**	0.8	0.74	**1.0**	**0.97**	0.8	0.82	**1.0**	0.58	0.92	0.82
Joint	0.9	0.97	**1.0**	0.92	**1.0**	**0.97**	**1.0**	**0.90**	0.9	0.42	**0.96**	**0.84**

Table 4. Retrieval for various object categories. We report precision at ranks 10 and 100. Notice the joint classification is superior overall, but there are two categories (bus and cycle), where using only lidar is advantageous. †ONCE fine-tuning.

the prompts proposed by Gu et al. [4], only inserting our automotive class names. We refer to the code for full prompting details[6].

Figure 1. Example of retrieval using separate prompts for image and lidar. We query for images with blur, water spray, glare, corruption, and lack of objects, and for point clouds with nearby trucks, pedestrians, cars, etc. By combining the scores of these separate queries, we can find edge cases that are extremely valuable during the training/validation of a camera-based perception system. These valuable objects are highlighted in red, both in the image and point cloud.

[6] https://github.com/atonderski/lidarclip/blob/main/lidarclip/prompts.py

D.3. Zero-shot scene classification

Here, we provide additional examples of zero-shot classification using LidarCLIP. However, rather than object-level classification, we do zero-shot classification on entire scenes. We compare the performance of image-only, lidar-only, and the joint approach.

Fig. 2 shows a diverse set of samples from the validation and test set. In many cases, LidarCLIP and image-based CLIP give similar results, highlighting the transfer of knowledge to the lidar domain. In some cases, however, the two models give contradictory classifications. For instance, LidarCLIP misclassifies the cyclist as a pedestrian, potentially due to the upright position and fewer points for the bike than the person. While their disagreement influences the joint method, "cyclist" remains the dominating class.

Another interesting example is the image of the dog. None of the models manage to confidently classify the presence of an animal in the scene. This also highlights a shortcoming with our approach, where the image encoder's capacity may limit what the lidar encoder can learn. This can be circumvented to some extent by using even larger datasets, but a more effective approach could be local supervision. For instance, using CLIP features on a patch level to supervise frustums of voxel features, thus improving the understanding of fine-grained details.

In Fig. 3 we highlight the importance, and problem,

P@K	10	100	10	100	10	100	10	100	10	100	10	100	10	100
Scene Cat.	Night		Day		Sunny		Rainy		Busy		Empty		Avg.	
PointCLIP	0.3	0.29	0.3	0.73	0.6	0.54	0.2	0.53	0.5	0.32	0.2	0.36	0.35	0.46
CLIP2Point	0.2	0.19	0.8	0.78	0.4	0.59	0.5	0.36	0.4	0.28	0.7	0.51	0.50	0.45
CLIP2Point†	1.0	0.75	0.7	0.77	0.5	0.49	1.0	1.00	0.9	0.75	0.7	0.52	0.80	0.71
Image	**1.0**	**1.00**	0.9	0.92	**1.0**	**1.00**	**1.0**	**1.00**	**1.0**	0.91	**0.8**	0.73	0.95	0.93
LidarCLIP	0.2	0.53	**1.0**	0.98	0.5	0.74	0.8	0.98	**1.0**	0.93	0.5	0.69	0.67	0.81
Joint	**1.0**	**1.00**	**1.0**	**0.99**	**1.0**	**1.00**	**1.0**	**1.00**	**1.0**	**0.99**	**0.8**	**0.79**	**0.97**	**0.96**

Table 5. Retrieval of scenes with various global conditions. We report precision at ranks 10 and 100. Notice that the joint classification is superior for all types of conditions. †ONCE fine-tuning.

Retrieval category	**Values**	**Prompt example**
time of day	night/day	a {quality} {format} of a {scene} {capture} at **{value}**
weather	sunny/rainy	a {quality} {format} {capture} in a **{value}** {scene}
busy	-	a {quality} {format} of extremely busy traffic during rush hour
empty	-	a {quality} {format} of a completely empty {scene}
object	car/pedestrian/...	there is a **{value}** in the scene

Table 6. Examples of the prompt templates used for various retrieval categories. Note that multiple prompt templates were used for each category to decrease the impact of the exact choice of prompt.

of including reasonable classes for zero-shot classification. The example scene contains a three-wheeler driving down the street. For the left sub-figure, none of the text prompts contains the word three-wheeler. Consequently, the model is confused between car, truck, cyclist, and pedestrian as none of these are perfect for describing the scene. When including "three-wheeler" as a separate class in the right sub-figure, the model accurately classifies the main subject of the image. To avoid such issues, we would like to create a class for unclassified or unknown objects, such that the model can express that none of the provided prompts is a good fit. Optimally, the model should be able to express what this class is, either by providing a caption or retrieving similar scenes, which can guide a human in naming and including additional classes. We hope that future work, potentially inspired by open-set recognition [3], can study this more closely.

D.4. LidarCLIP for lidar sensing capabilities

In Fig. 4, we show additional examples of retrieved scenes for various colors. Note that images are only shown for reference and that LidarCLIP only observes the point cloud. Similar to results in Sec. 4.2, LidarCLIP has no understanding of distinct colors but can discriminate between dark and bright. For instance, "a gray car" returns dark grey cars, while none of the cars for "a yellow car" are yellow.

D.5. Lidar to image and text

We provide additional examples of generative applications of LidarCLIP in Fig. 6. These are randomly picked scenes with no tuning of the generative process. The latter is especially important for images, where small changes in guidance scale[7] and number of diffusion steps have a massive impact on the quality of the generated images. Furthermore, to isolate the impact of our lidar embedding, we use the same parameters and random seeds for the different scenes. This leads to similar large-scale structures for images with the same seed, which is especially apparent in the rightmost column.

In most of these cases, the generated scene captures at least some key aspect of the embedded scene. In the first row, all generated scenes show an empty road with many road paintings. There is also a tendency to generate red lights. Interestingly, several images show localized blurry artifacts, similar to the raindrops in the source image. The second row shows very little similarity with the embedded scene, only picking up on minor details like umbrellas and dividers. In the third row, the focus is clearly on the bus, which is present in the caption and three out of four generated images. In the fourth row, the storefronts are the main subject, but the generated images do not contain any cars, unlike the caption. In the final scene, we see that the model picks up the highway arch in three out of the four generated images, but the caption hallucinates a red stoplight, which is not present in the source.

[7]The guidance scale is a parameter controlling how much the image generation should be guided by CLIP, which may stand in contrast to photorealism.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 1

[2] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8458–8468, 2022. 1

[3] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(10):3614–3631, 2020. 4

[4] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *International Conference on Learning Representations*, 2022. 3

[5] Tianyu Huang, Bowen Dong, Yunhan Yang, Xiaoshui Huang, Rynson WH Lau, Wanli Ouyang, and Wangmeng Zuo. Clip2point: Transfer clip to point cloud classification with image-depth pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 1, 2

[6] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 2

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 1

[8] Jiageng Mao, Minzhe Niu, Chenhan Jiang, hanxue liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, Jie Yu, Hang Xu, and Chunjing Xu. One million scenes for autonomous driving: ONCE dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 1

[9] Ron Mokady, Amir Hertz, and Amit H Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021. 2

[10] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 2

[11] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by CLIP. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8552–8562, 2022. 1, 2

Figure 2. Qualitative zero-shot classification on the ONCE validation/test set.

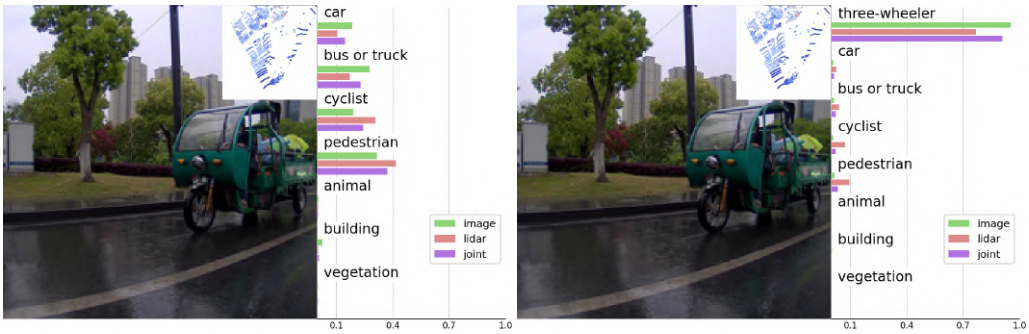

Figure 3. Example of zero-shot classification that demonstrates the importance of picking good class prompts. When excluding the most appropriate class, "three-wheeler", the model is highly confused between the remaining classes.

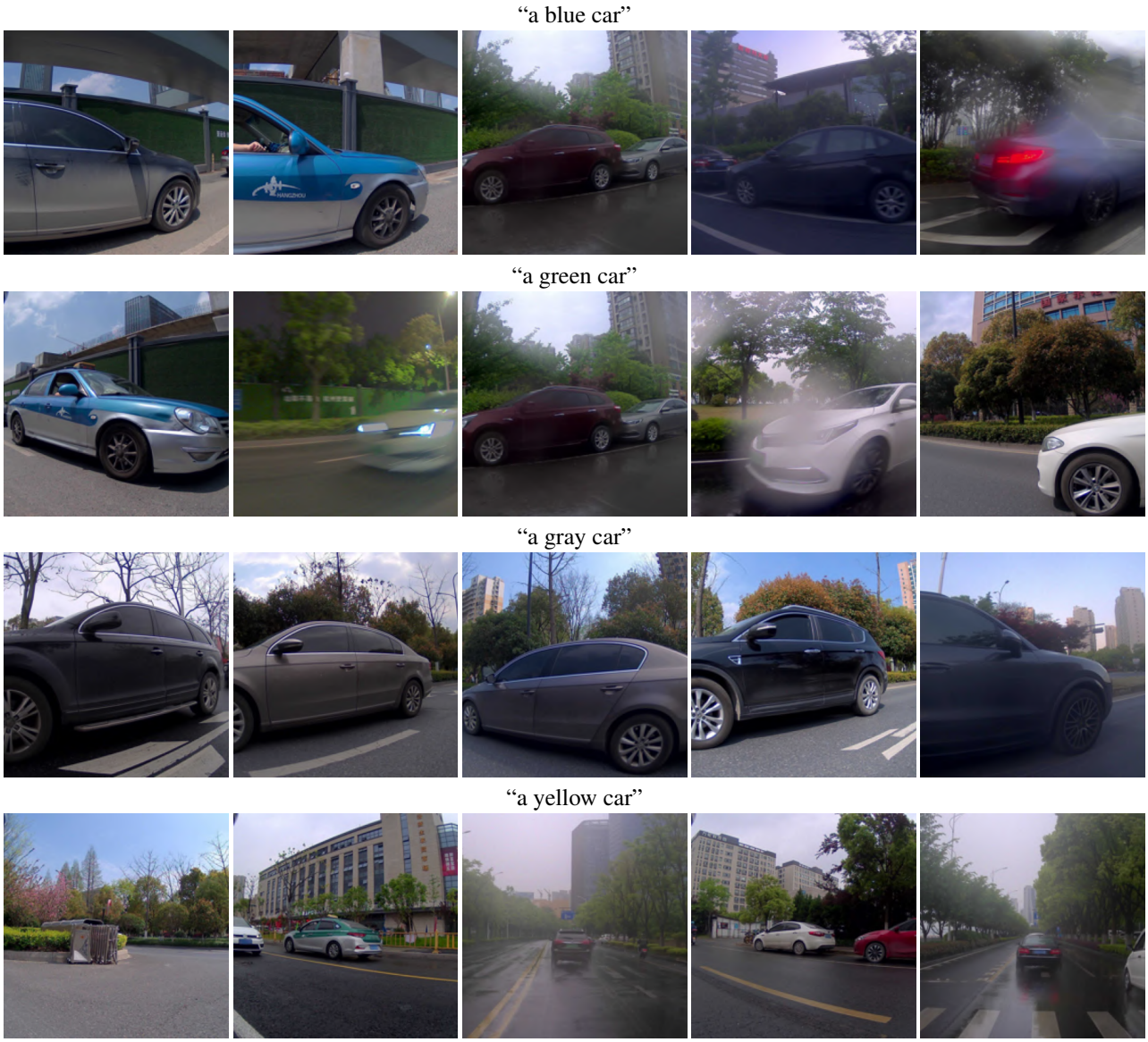

Figure 4. Top-5 retrieved examples from LidarCLIP for different colors. Note that we show images only for visualization, point clouds were used for retrieval.

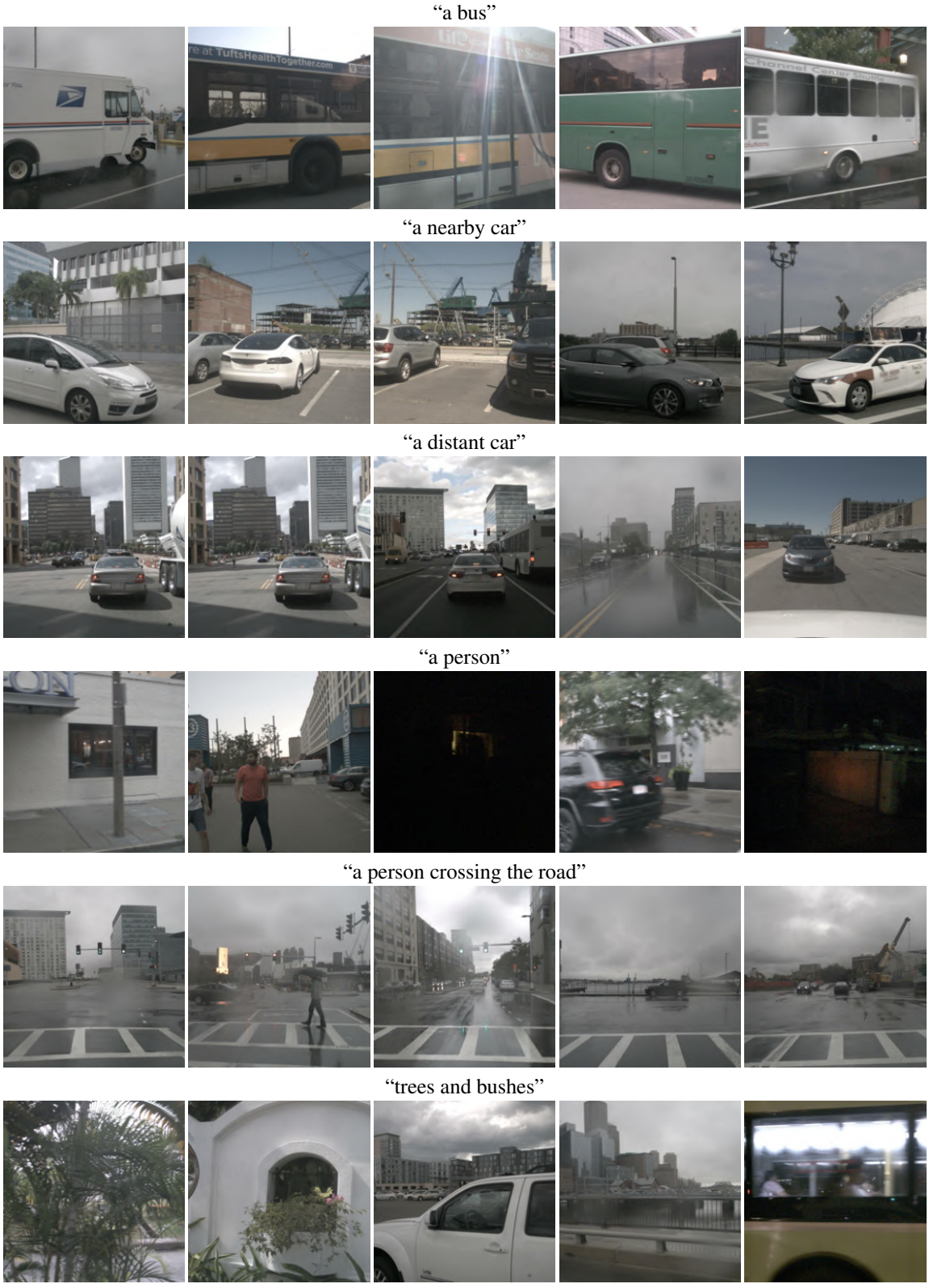

Figure 5. Top-5 retrieved examples when transferring LidarCLIP to nuScenes. LidarCLIP generalizes decently for distinct, large, objects, and even maintains some concept of distances. However, smaller objects and less uniform objects, like pedestrians and vegetation, do not transfer well.

Figure 6. Example of generative application of LidarCLIP. A point cloud is embedded into the clip space (left, image only for reference) and used to generate text (top) and images (right). All four images are only generated with guidance from the lidar embedding, the caption was not used for guidance.