Appendix- Supplementary Materials

Cross-modal Self-training: Aligning Images and Pointclouds to learn Classification without Labels

The following section contains supplemental information and encompasses more implementation details, a comparison of datasets, and further analysis of discriminative features learned by the proposed cross-modal self-training. The contents are organized in the order listed below.

- Cross-modal joint pseudo-labels (Appendix 1)
- Additional implementation details (Appendix 2)
- Datasets (Appendix 3)
- Training trends (Appendix 4)
- Analysis of feature embeddings (Appendix 5)

1. Cross-modal joint pseudo-labels

We generate a training sample by pairing a point cloud object with an image showing the object from a random viewpoint. A set of weak augmentations are applied to the batch B of image and point cloud pairs which is then passed through the teacher model to obtain two sets of soft prediction logits from each modality; $q_{b,img}, q_{b,pcl}$ Then, the teacher prediction for each sample that corresponds to the highest confidence between the two modalities is selected to generate a common set of pseudo-labels for both image and point cloud modalities as follows:

$$\hat{r}_b = \{\mathbb{1}\left(\max(q_{b,img}) \geq \max(q_{b,pcl})\right) \hat{q}_{b,img} \\ \cup \mathbb{1}\left(\max(q_{b,img}) < \max(q_{b,pcl})\right) \hat{q}_{b,pcl}\ \},$$

where $\hat{q}_{b,img} = \text{argmax}_c(q_{b,img})$ and $\hat{q}_{b,pcl} = \text{argmax}_c(q_{b,pcl})$. The confidence scores for the combined pseudo-labels are denoted by

$$r_b = \max_b\{\max_c(q_{b,img}), \max_c(q_{b,pcl})\}$$

where $r_b \in R^B$.

2. Implementation details

We use ViTB16 as the image encoder, and a standard transformer [11] with multi-headed self-attention layers and FFN blocks as the point cloud encoder. The [MSK] tokens, projection layers, and decoder layers are randomly initialized, and the classifier is initialized with the text embedding as explained in **??**. Then they are fine-tuned together with the encoders in the EMA teacher-student setting. We use AdamW [6] optimizer with a weight decay of 0.05. We apply a cosine learning rate scheduler and similar to [1, 5], we apply layer-wise learning rate decay of 0.65. The batch size is 512, and the learning rate is scaled linearly with the batch size as (lr= base_lr*batchsize/256). We used 4 V100 GPUs for training.

Image encoder: We initialize image encoder with [7] weights. We use a ViT-B/16 model pretrained by [7] for the image branch, containing 12 transformer blocks with 768 dimensions. The model receives input images of size 224×224. After resizing, random cropping is applied as a weak augmentation on the input to the teacher model to generate pseudo-labels. A set of stronger augmentations; RandomResizedCrop+Flip+RandAug [4] is applied to the input to the student model. We implement a patch-aligned random masking strategy where multiple image patches are randomly masked with a fixed ratio of 30%.

Point cloud encoder: We used Shapenet [2] dataset and its rendered 2D views from [13] to pre-train the point cloud encoder and the projection layers f^P, f^S, and f^I while the image and text encoders are kept frozen. Training is done for 250 epochs with a learning rate of 10^{-3} with AdamW optimizer with a batch size of 64. We divide each point cloud into 64-point patches each containing 32 points. Point centers for clustering local point patches are selected using farthest point sampling (FPS). Then the locations of corresponding point centers are subtracted from the local patches such that they contain only the local geometric information irrespective of its original position. We use a Mini-Pointnet to extract point embeddings of each sub-cloud, followed by a sharpened-pre trained encoder from [14] to convert each sub-cloud into point embeddings. The center locations of each sub-cloud are passed through an MLP to encode a positional embedding and are appended to the point embeddings, before passing to the transformer. The depth of the transformer is set to 12, the feature dimension to 384, and the number of heads to 6. During point encoder pre-training as well as self-training, the encoder is kept frozen. After normalizing, weak augmentations such as rotation perturbation and random scaling in the range of 90% 110% are applied on the input point-cloud to the teacher model to generate pseudo-labels. Stonger augmentations; random cropping, input dropout, rotate, translate, and scaling in the range of 50% 200% are applied to the input to the student model. Random masking is applied to 30% to 40% of the point embeddings.

3. Datasets

Shapenet [2] consists of textured CAD models of 55 object categories. We uniformly sample points from each object mesh to create the pointclouds, after which they are nor-

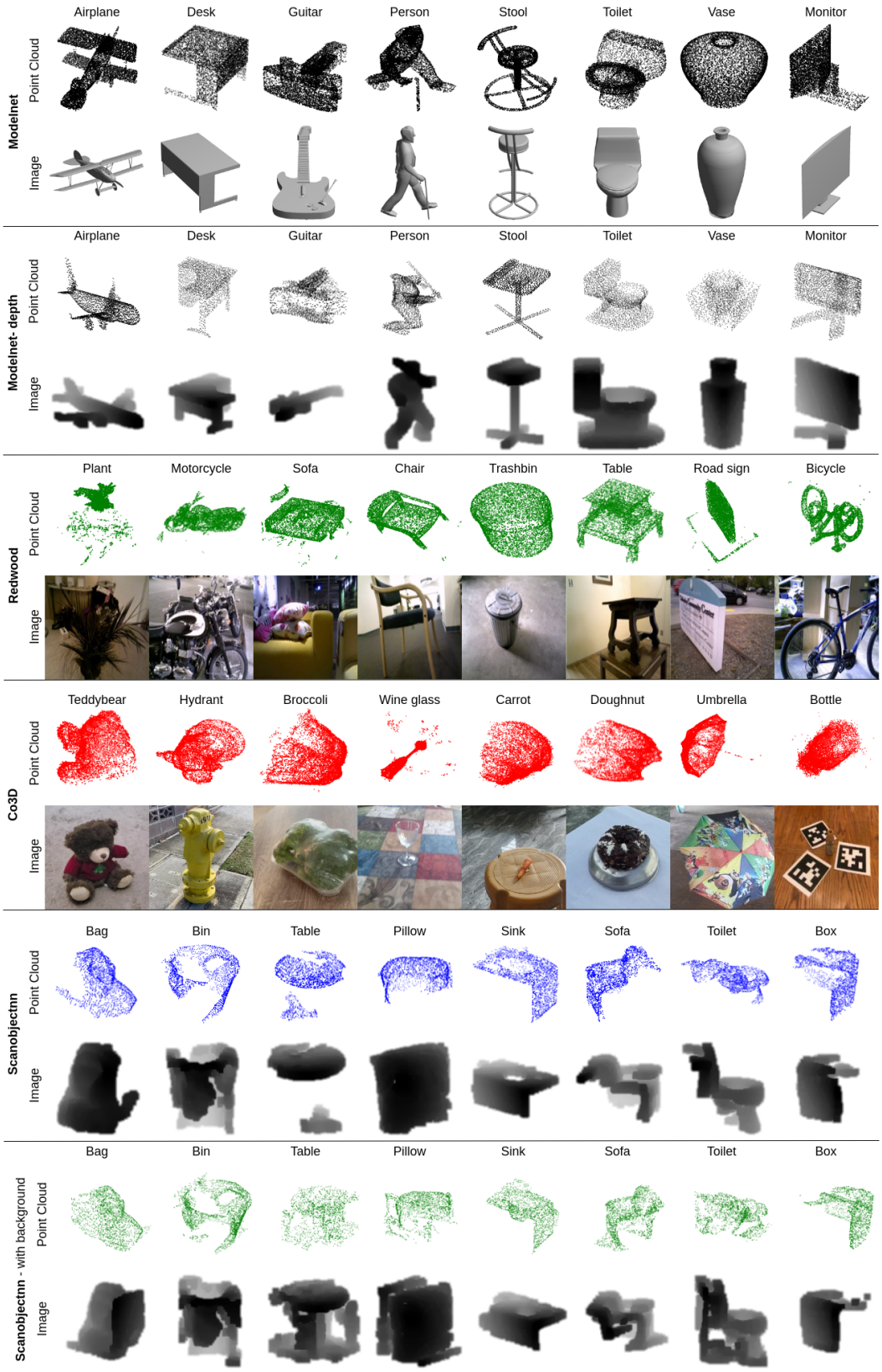

Figure 1. Qualitative comparison of datasets.

malized to fit a unit sphere. Following the work of [13], we render RGB images for each object from different viewpoints. We use this dataset to pretrain the pointcloud branch to provide a better initialization for self-training.

ModelNet40 [12] is a synthetic dataset of 3D CAD models containing 40 categories. The pointclouds are sampled from the object mesh and normalized. We use the work of [9] to get realistic 2D renderings of the CAD models. We pair these renderings with the pointclouds to create Modelnet40 and ModelNet10 (A subset of 10 common classes from [12]). We follow the realistic 2D views generated using [15] to generate the dataset ModelNet40-d (depth).

Redwood [3] is a dataset of real-life high-quality 3D scans and their mesh reconstructions. However, these object meshes also contain points from the floor plane and surrounding objects. Unlike CAD models in [2, 12], Redwood scans are not axis-aligned. For these models to be consistent with such datasets, we run a RANSAC plane detection to identify the floor plane and then rotate the orientation of the object to match the 3D coordinate axes. We then removed the points from surrounding objects/noises by manually cropping every scan. We randomly sample 20 frames from the RGB videos of each object scan and use them in our image branch.

Co3D [8] is a large scale dataset of multiview images capturing common 3D objects. These 2D views have been used to reconstruct 3D point clouds representing each object. Following the work of the original authors, we filter out only the accurate 3D point clouds for our experiments. Being SLAM-based reconstructions, these point cloud objects are also not axis-aligned. We implement a rough correction to the orientations by calculating the principal component directions based on the point densities of the point cloud and rotating the object to align it with the gravity axis in the 3D coordinates. However, this is a very challenging point cloud dataset due to differences in orientation, obscured parts of objects, and surrounding noises. We sample 20 images for each object from the RGB multiview images for the image branch.

Scanobjectnn [10] is a dataset of real scans of 15 object classes. 10 2D views per each object are generated using [15]. We report results on 3 different versions of the dataset; Sc-obj - clean point cloud objects, Sc-obj withbg - Scans of objects with backgrounds, and Sc-obj hardest - Scans with backgrounds and additional random scaling and rotation augmentations.

	Train		**Test**		
	Images	Point clouds	Images	Point clouds	Categories
Shapenet	503316	41943	126204	10517	55
ModelNet	38196	3183	9600	800	40
Redwood	17270	314	4620	84	9
co3d	110536	5519	28192	1406	42
Scanobjectnn	23090	2309	5810	581	15

Table 1. Dataset details

Qualitative comparison: qualitative comparison of the point clouds and accompanying 2D views for the aforementioned datasets are shown in Figure 1

3.1. Modelnet40 dataset splits:

There are some common classes between our pre-train dataset, ShapeNet55, and ModelNet40. Evaluations on these common classes might introduce an unfair comparison of zeroshot performance. ULIP [13] authors introduced three different validation sets for evaluating models and baselines on ModelNet40.

All Set: Includes all the categories in ModelNet40 as shown in Table 2.

Medium Set: Categories whose exact category names exist in pre-training dataset; Shapenet55 have been removed. The resulting categories in this set are shown in Table 3.

Hard Set:Both extract category names and their synonyms in the pre-training dataset have been removed. The final Hard Set is shown in Table 4.

Airplane	Bathtub	Bed	Bench	Bookshelf
Bottle	Bowl	Car	Chair	Cone
Cup	Curtain	Desk	Door	Dresser
Flower Pot	Glass Box	Guitar	Keyboard	Lamp
Laptop	Mantel	Monitor	Night Stand	Person
Piano	Plant	Radio	Range Hood	Sink
Sofa	Stairs	Stool	Table	Tent
Toilet	TV Stand	Vase	Wardrobe	Xbox

Table 2. Modelnet40-All set

Cone	Cup	Curtain	Door	Dresser
Glass Box	Mantel	Monitor	Night Stand	Person
Plant	Radio	Range Hood	Sink	Stairs
Stool	Tent	Toilet	TV Stand	Vase
Wardrobe	Xbox			

Table 3. Modelnet40-Medium set

Cone	Curtain	Door	Dresser	Glass Box
Mantel	Night Stand	Person	Plant	Radio
Range Hood	Sink	Stairs	Tent	Toilet
TV Stand	Xbox			

Table 4. Modelnet40-Hard set

4. Measuring entropy and biasedness of predictions

To calculate the entropy and biasedness of the predictions of our model, we use KL-Divergence to model the relative entropy from U to P where U and P are probability distributions.

$$D_{KL}(P||U) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{U(x)} \right)$$

First we define a uniform probability distribution u, where C is the number of categories.

$$u = \left\{ \frac{1}{C} \right\}_{c=1}^{C}$$

p is the logits produced by the model for a single input in the test set. Then, the entropy of the particular prediction can be calculated as:

$$Pred.\ Entropy\ = \sum_{c \in C} p_c \log \left(\frac{p_c}{u_c} \right) = \sum_{c \in C} p_c \log(C p_c)$$

and averaged over the test set to record the entropy of predictions as training progresses.

To calculate the biasness of the model towards predicting certain classes, we define a vector j:

$$j = \frac{1}{S} \{a_1, a_2, a_3 ..., a_C\}$$

where S is the size of the test set, and a_c is the number of samples predicted as category c by the model. Then, the prediction bias of the model in one test cycle is calculated as:

$$Pred.\ Bias\ = \sum_{c \in C} j_c \log \left(\frac{j_c}{u_c} \right) = \sum_{c \in C} j_c \log(C j_c)$$

5. Analysis of feature embeddings

Quality of feature embeddings: We visualize the TSNE embeddings of the 3D and 2D features extracted from respective modalities before and after cross-modal self-training for ModelNet10 in Figure 2 and Scanobjectnn 3. In both cases, features from both image and point cloud modalities show improved class separability after self-training.

Furthermore, in Figure 4 and Figure 5 we visualize the TSNE embeddings against the original class labels, as well as the predictions returned by our model. It is evident that the feature-level discrimination as well as class-level classification results are significantly improved over zeroshot by using our proposed method.

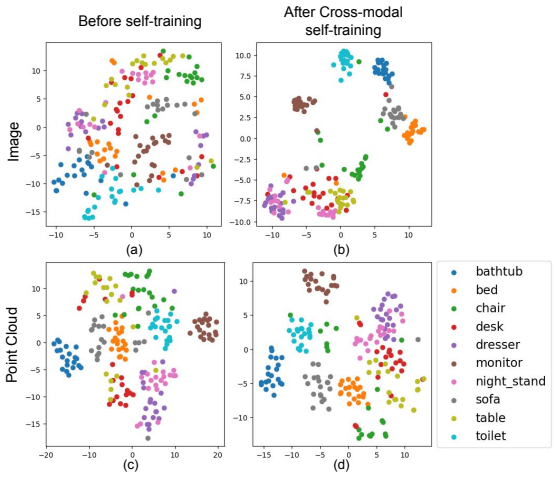

Figure 2. TSNE-feature embeddings for Modelnet10 before and after Cross-modal self-training.

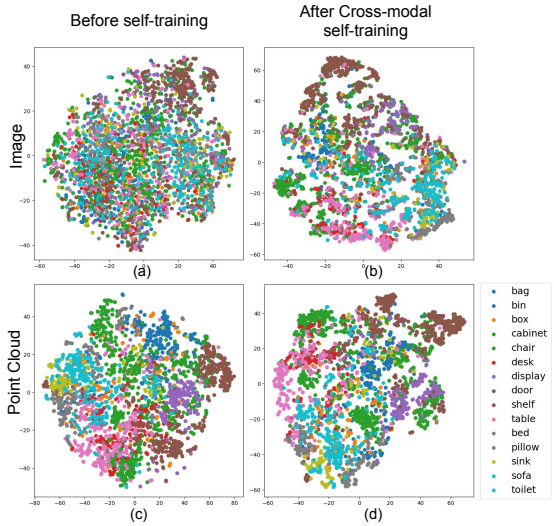

Figure 3. TSNE-feature embeddings for Scanobjectnn before and after Cross-modal self-training..

References

[1] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers, 2022. 1

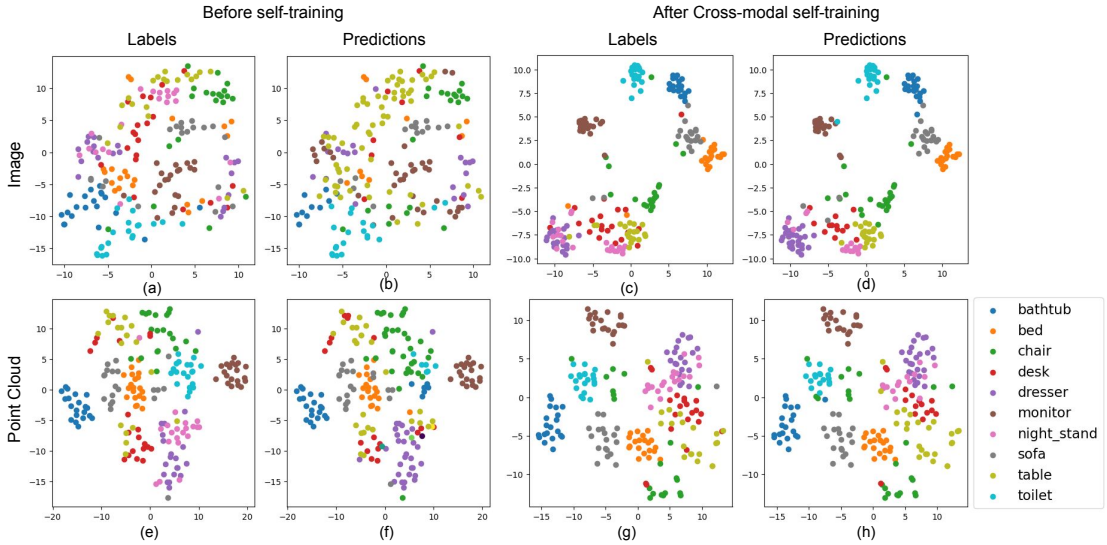

Figure 4. TSNE-feature embeddings for Modelnet10 before and after Cross-modal self-training with the original class labels and the predictions returned by our classifier.

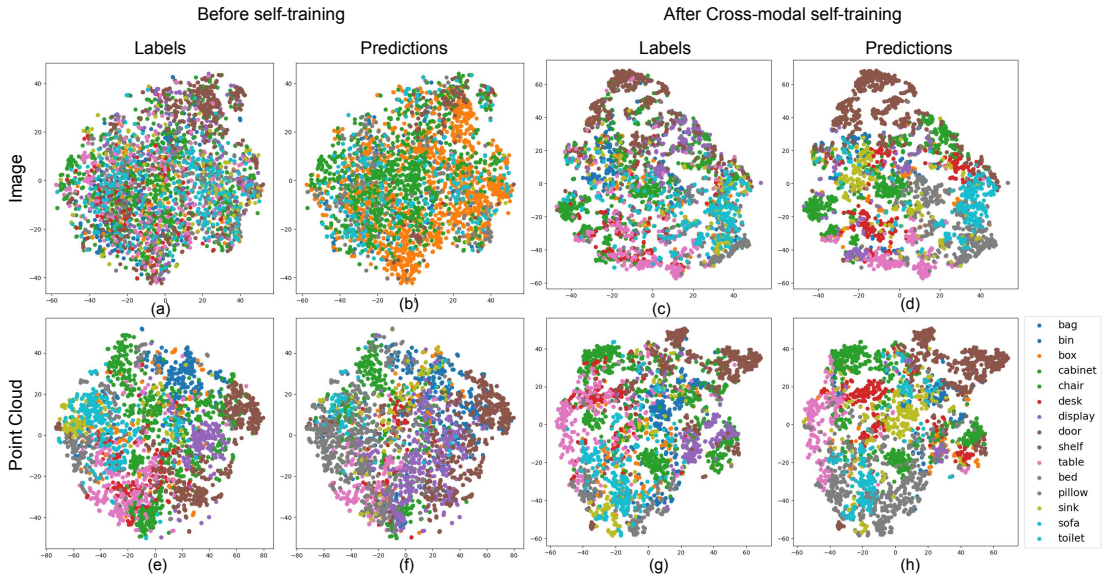

Figure 5. TSNE-feature embeddings for Scanobjectnn before and after Cross-modal self-training with the original class labels and the predictions returned by our classifier.

[2] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. *arXiv:1512.03012*, 2015. 1, 3

[3] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans, 2016. 3

[4] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019. 1

[5] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021. 1

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 1

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual

models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8748–8763, 2021. 1

[8] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction, 2021. 3

[9] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition, 2015. 3

[10] 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 *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1588–1597, 2019. 3

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 1

[12] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3

[13] Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding, 2023. 1, 3

[14] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling, 2022. 1

[15] Xiangyang Zhu, Renrui Zhang, Bowei He, Ziyao Zeng, Shanghang Zhang, and Peng Gao. Pointclip v2: Adapting clip for powerful 3d open-world learning, 2022. 3