ABO: Dataset and Benchmarks for Real-World 3D Object Understanding (Supplemental Material)

1. Dataset Properties

Metadata Visualization We visualize additional products according to their metadata attributes in Figure 1. For each attribute (unit count and weight), each row depicts products that fall into that label (for categorical attributes) or bin (for continuous-valued attributes). For continuous-valued attributes, products within the same row are ordered from lowest to highest.

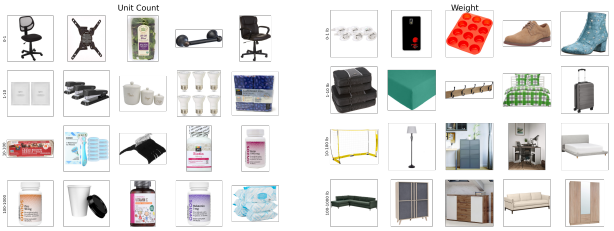

Figure 1. **Metadata visualization.** Catalog image samples for "unit count" and "weight" metadata attributes. For each product we show its *main* image.

2. Dataset Organization

In this work we used ABO to derive benchmarks for different tasks such as 3D reconstruction, material estimation, and multi-view retrieval. Table 1 outlines how each data subset in ABO is used at both train and test time. "No-BG Renders" refers to the white-background rendered dataset of ABO objects described in Section 4.1 of the main text, while "BG Renders" refers to the Material Estimation Dataset described in Section 3 of the main text.

		3D Recon.	Material Est.	Retrieval
Train	No-BG Renders			
	BG Renders		✓	✓
	Catalog Images			✓
Test	No-BG Renders	✓		
	BG Renders		✓	✓
	Catalog Images		✓	✓

Table 1. **ABO data subsets used in each benchmarking experiment at both train and test time.** "BG" and "No-BG" Renders refer to renderings from 3D models in ABO with and without backgrounds.

3. 6DOF Pose Optimization

Instance Masks We use instance masks generated both from MaskRCNN [6] trained on LVIS [5] as well as PointRend [7] trained on COCO [9]. Both model checkpoints were obtained from the Detectron2 repository[1]. We kept predicted masks from all categories with a confidence score greater than 0.1.

Pose Optimization For each instance mask, we initialize 24 different runs with random rotations and optimize $\mathbf{R}$ and $\mathbf{T}$ for $1,000$ steps using the Adam optimizer with a learning rate of 1e-2. We parameterize the rotation matrix using the symmetric orthogonalization procedure of [8]. At the end of the 24 runs, we pick the pose that has the lowest loss and validate its correctness via a human check.

4. Single-View 3D Reconstruction Evaluation

View-Space Evaluations Since view-space predictions can be scaled and z-translated simultaneously while preserving the projection onto the camera, we must solve the depth-ambiguity to align the predicted and ground truth (GT) meshes for benchmarking. We use known camera extrinsics to transform the GT mesh into view-space, and align it with the predicted mesh by solving for a Chamfer-distance minimizing depth. In practice, we normalize average vertex depths for each mesh independently and then search through 51 candidate depths. We compare the predicted and GT meshes after alignment following the evaluation protocol in [2, 3] and report Chamfer distance and Absolute Normal Consistency. Since Chamfer varies with the scale of the mesh, we follow [2, 3] and scale the meshes such that the longest edge of the GT mesh bounding box is of length 10.

Canonical-Space Evaluations To evaluate shapes pre-

[1] https://github.com/facebookresearch/detectron2

	Chamfer (↓)	Abs. Normal Consistency (↑)
3D R2N2 [1]	1.97	0.55
OccNets [11]	1.19	0.70
GenRe [13]	1.61	0.66
Mesh R-CNN [4]	0.82	0.62

Table 2. **3D reconstruction on ABO test split.** Chamfer distance and absolute normal consistency averaged across all categories

dicted in canonical space, we must first align them with the GT shapes. Relying on cross-category semantic alignment of models in both ShapeNet and ABO, we use a single (manually-set) rotation-alignment for the entire data. We then solve for relative translation and scale, which remain inherently ambiguous, to minimize the Chamfer distance between the two meshes. In practice, we search over a 3^4 grid of candidate scale/translation after mean-centering (to vertex centroid) and re-scaling (via standard deviation of vertex distances to centroid) the two meshes independently. Note that R2N2 [1] predicts a voxel grid so we convert it to a mesh for the purposes of benchmarking. Marching Cubes [10] is one such way to achieve this, however, we follow the more efficient and easily batched protocol of [3] that replaces every occupied voxel with a cube, merges vertices, and removes internal faces.

Additional Qualitative Examples We show additional reconstructions of ABO objects from the best performing single-view 3D reconstruction method, Mesh R-CNN, and the method that claims to be category-agnostic, GenRe in Figure 2. We find that both methods generally fail to reconstruct objects with thin structures, but that the failure occurs in different ways for each method. GenRe simply does not reconstruct them, whereas Mesh R-CNN produces a reconstruction that qualitatively appears more like the 3D convex hull of the object.

Quantitative Evaluation on Test Split As the focus of this work was to measure the domain gap of ShapeNet-trained 3D reconstruction methods to ABO, we evaluated reconstruction performance for all 3D models in ABO that came from ShapeNet categories. To facilitate future research that may want to *train* 3D reconstruction methods using ABO and compare to ShapeNet trained methods, we generate a train/val/test split (80%/10%/10%) that we release along with the dataset. We also report the performance (averaged across all categories) of each method on this test split in Table 2.

Figure 2. **Additional qualitative 3D reconstruction results for GenRe and Mesh R-CNN.** The first four rows display ABO objects that overlap with common ShapeNet categories, such as cabinet, chair and table. The bottom four rows display reconstructions for ABO objects from categories not represented in ShapeNet.

5. Material Estimation

Dataset Curation We use the Material Estimation Dataset outlined in the main text, omitting object with transparencies, resulting in 7,679 models. We split the 3D models into a non-overlapping train/test set of 6,897 and 782 models, respectively. To test generalization to new lighting conditions, we reserve 10 out of 108 HDRI environment maps for the test set only.

Network Details A visualization of the single-view material estimation network (SV-net) and multi-view network (MV-net) can be found in Figure 3 (middle and bottom, respectively). The MV-net uses camera poses to establish pixel-level correspondences. Given a pixel p in one viewpoint with image coordinate x and depth z, the image coordinate x' of the its corresponding pixel in another viewpoint can be computed as $x' = KRK^{-1}x + Kt/z$, where K is the camera intrinsic matrix, R and t are the rotation and translation between the two viewpoints. Some pixels in one view can be occluded and hence not visible in other views. These can be determined by using a depth-based occlusion test. We fill these pixels using values from the reference view.

Full Texture Map Reconstruction Using per-view predicted material maps, we can also generate a full textured PBR model. This is achieved by back-projecting the predicted material maps to the UV domain and using a learned network to aggregate and smooth the predictions. Figure 4 shows the full UV map reconstruction pipeline.

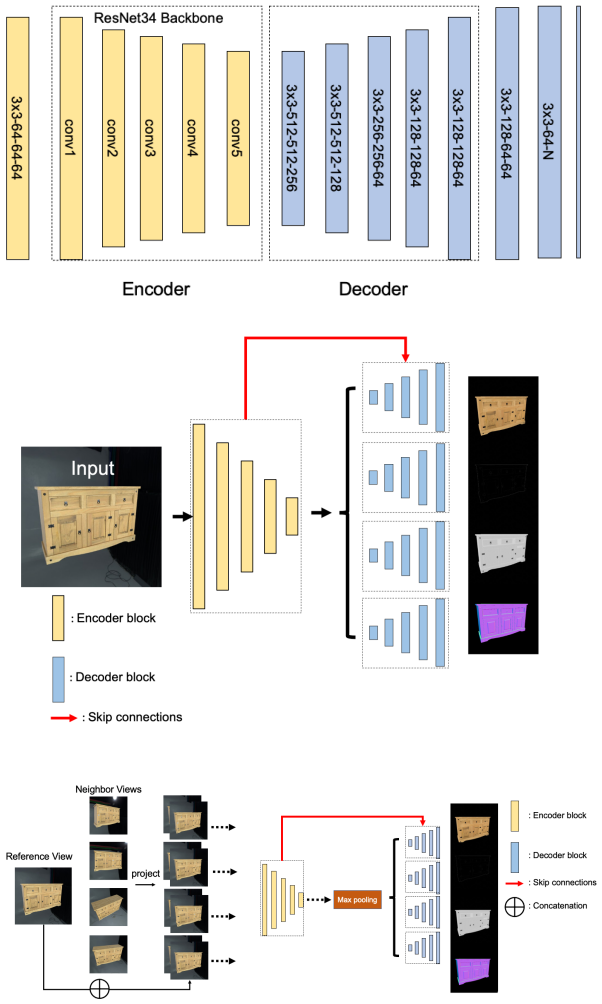

Figure 3. **Top:** Encoder-decoder architecture. Encoder uses a ResNet-34 (conv1-conv5) backbone. *KxK-N-M-X* denotes a double convolution block of *KxK* filter, *N* input channels, *M* intermediate channels, and *X* output channels. We use BatchNorm and leaky ReLU. **Middle:** Single-view baseline. **Bottom:** Multi-view baseline. Given a reference view, neighboring views are selected and projected to the reference view and passed as input to the network.

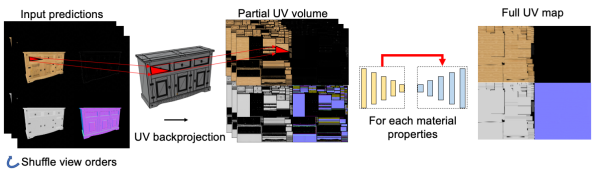

Figure 4. **Pipeline for full UV map prediction.** Per-view predictions are back projected onto the UV and fed to an encoder-decoder network for smoothing and aggregation.

6. Multi-View Cross-Domain Object Retrieval

Dataset Curation For this task, we used product type annotations to focus only on rigid objects, removing items such as garments, home linens, and some accessories (cellphone accessories, animal harnesses, cables, etc.). As the set of products beyond just those with 3D models are likely to contain near-duplicates (i.e. different sizes of the same shoe), we then applied a hierarchical Union-Find algorithm for near-duplicate detection and product grouping, based on shared imagery as a heuristic. We considered near-duplicates as correct matches of a single instance and thus assigned a unique *instance id* to all near-duplicate listings. Product groups are sets of such instances that are from product lines that may share design details, materials, patterns and thus may have common images (close-up detail of fabric, fact sheet image, ...). Consequently, we ensured that all instances in a group are assigned to the same data split (train, val or test). The val and test sets contain only instances with 3D models and, while their catalog images compose their respective target set (val-target and test-target), we use rendered images as queries (val-query and test-query).

Dataset Curation Statistics The hierarchical Union-Find algorithm yielded $29,988$ groups of $50,756$ instances, of which $1,334$ have 3D models ($5,683$ instances). We then sampled groups accounting for 836 instances with 3D models for the *test* set, using their combined $4,313$ catalog images as *test-target* images and sampled 8 rendered views for each of the environment maps as *test-query* images. Again, we sampled groups with 854 of the remaining instances with 3D models for the *validation* set, using catalog images ($4,707$) as *val-target* and 8 rendered images per envmap as *val-query*. We used the rest of the instances as the *train* set: $49,066$ instances ($3,993$ with 3D model), $187,912$ catalog images and $110,928$ rendered images ($298,840$ total).

Implementation Details Train images are pre-processed by the following: square padding, resize to 256x256, random resized crop of 227x227 (scale between 0.16 and 1 and ratio between 0.75 and 1.33) and random horizontal flip ($p = 0.5$). Test images are padding to square, resized to 256x256 and center cropped to 227x227.

The trunk used is ResNet-50 pre-trained on ImageNet, leading to a 2048D vector. We did not freeze these weights, including the BatchNorm parameters. The embedding module is a LayerNorm normalization followed by linear projection to 128D embedding. The batch sampler is class-balanced and domain-balanced. Domain-balancing is implemented as a hierarchical sampling: we first sample N classes with rendered images and N classes without, then sample K images for each class among all available images for the class, leading to batches of $2NK$ samples. For NormSoftmax and ProxyNCA we used batches of 32 samples, 1 sample per class, 16 classes with rendered images and 16 without. For all other methods we used batches of 256 samples, 4 samples per class, 64 classes with rendered images and 64 classes without.

One epoch consists of 200 batches sampled from the above procedure. We trained the models for 1000 epochs, using early stopping when no improvements on the validation accuracy are found for over 250 consecutive epochs. The epoch for testing is selected based on the maximum validation Recall@1 for validation. We used 8 workers for data loading on a AWS p3.8xlarge instance (32 cores, 4 Nvidia V100 GPUs). As [12], we did not use tuple mining within a batch. The embeddings are normalized before indexing and querying. We used Recall@1 using *val-query* rendered images as queries against *val-target* catalog images as the validation metric, computing it at every even epoch. RMSProp with a learning rate $1e{-}6$, weight decay $1e{-}4$, and momentum 0.9 is used to optimize both the trunk and mebedding layers. For metric losses, where applicable, the learning rate is optimized via hyperparameter optimization.

Hyperparameters are estimated by 30 runs of Bayesian hyperparameter optimization for 100 epochs, using the implementation of Powerful-Benchmarker, the epoch is chosen on the full run after optimization:

- Contrastive. Pos margin: 0.5287; Neg margin: 1.0523; Epoch: 614
- Multi-similarity. Alpha: 0.0240; Beta: 49.1918; Base: 0.5567; Epoch: 76
- NormSoftmax. Temperature: 0.0776; Metric loss learning rate: 0.0013; Epoch: 448
- NTXent. Temperature: 0.0665; Epoch: 114
- ProxyNCA. Softmax scale: 5.5915; Metric loss learning rate: 0.0010; Epoch: 566
- Triplet. Margin: 0.0743; Epoch: 706

Additional Metrics Tables 3 and 4 provide a more complete version of Table 5 from the main manuscript. In addition to Recall@k, we also report mean average precision (MAP), mean average precision at R (MAP@R) and R-Precision as desribed and implemented by [12]. These metrics provide additional insights compared to recall, as they give higher value to retrieval results with correct rankings and having as many correct results as possible in the first ranks.

In Table 3, we compare the results when using rendered images of test classes as queries, against the union of catalog images of test classes and catalog images of train classes. In Table 4, we instead compare the results when using catalog images of test classes as queries, against the same union of catalog images of test classes and catalog

Loss	Recall@1 (%)	Recall@2 (%)	Recall@4 (%)	Recall@8 (%)	MAP (%)	MAP@R (%)	R-Precision (%)
Pre-trained	4.97	8.10	11.41	15.30	7.69	2.27	3.44
Constrastive	28.56	38.34	48.85	59.10	31.19	**14.16**	**19.19**
Multi-similarity	23.12	32.24	41.86	52.13	26.77	11.72	16.29
NormSoftmax	**30.02**	**40.32**	**50.19**	59.96	**32.61**	14.03	18.76
NTXent	23.86	33.04	42.59	51.98	27.00	12.05	16.51
ProxyNCA	29.36	39.47	50.05	**60.11**	32.38	14.05	19.00
TripletMargin	22.15	31.10	41.32	51.90	25.80	10.87	15.41

Table 3. **Test metrics for the ABO-MVR benchmark, using rendered images as queries**. The gallery images are derived from catalog images and contain classes from the train and test classes.

Loss	Recall@1 (%)	Recall@2 (%)	Recall@4 (%)	Recall@8 (%)	MAP (%)	MAP@R (%)	R-Precision (%)
Pre-trained	17.99	23.93	31.72	38.65	22.57	6.99	9.55
Constrastive	**39.67**	**52.21**	**64.41**	**71.64**	**42.96**	**22.52**	**28.07**
Multi-similarity	38.05	50.06	61.79	68.17	40.87	21.06	26.32
NormSoftmax	35.50	46.70	57.38	64.78	38.07	18.63	23.42
NTXent	37.51	49.34	61.37	69.23	40.12	20.03	25.32
ProxyNCA	35.64	46.53	57.36	65.06	38.50	18.81	23.65
TripletMargin	36.87	48.34	60.98	69.44	40.03	19.94	25.46

Table 4. **Test metrics for the ABO-MVR benchmark, using catalog images as queries**. The gallery images are derived from catalog images and contain classes from the train and test classes.

images of train classes. Naturally, the image used as query is discarded from its own search results before computing the metrics. Still, as we can see, these is a significant gap between retrieving from rendered queries compared to catalog images. This highlights the difficulty of this new benchmark.

Qualitative Results In Figure 5 we show qualitative results for a few queries (low elevation, mid elevation, high elevation), showing some success and failure cases of NormSoftmax, ProxyNCA and Contrastive.

Figure 5. **Qualitative retrieval results for low, medium, and high elevation products.** The leftmost column shows the query image, the other 8 columns show the top-8 results, highlighted in green if correct and red if incorrect. The top 3 rows are results of Contrastive, the middle 3 are of NormSoftmax and the bottom 3 are ProxyNCA. Each group of 3 rows have the same queries: one of low elevation (side table), one of mid-elevation (air conditioner), one of high elevation (cart).

References

[1] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016.

[2] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017.

[3] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9785–9795, 2019.

[4] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *ICCV*, 2019.

[5] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5356–5364, 2019.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[7] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9799–9808, 2020.

[8] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *arXiv preprint arXiv:2006.14616*, 2020.

[9] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[10] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[11] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[12] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020.

[13] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Josh Tenenbaum, Bill Freeman, and Jiajun Wu. Learning to reconstruct shapes from unseen classes. In *Advances in Neural Information Processing Systems*, pages 2257–2268, 2018.