MVTN: Multi-View Transformation Network for 3D Shape Recognition Supplementary Material

Abdullah Hamdi Silvio Giancola Bernard Ghanem
King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
{abdullah.hamdi, silvio.giancola, bernard.ghanem}@kaust.edu.sa

1. Detailed Experimental Setup

1.1. Datasets

ModelNet40. We show in Fig. 1 examples of the mesh renderings of ModelNet40 used in training our MVTN. Note that the color of the object and the light direction are randomized in training for augmentation but are fixed in testing for stable performance.

ShapeNet Core55. In Fig. 2, we show examples of the point cloud renderings of ShapeNet Core55 [4, 27] used in training MVTN. Note how point cloud renderings offer more information about content hidden from the camera view-point, which can be useful for recognition. White color is used in training and testing for all point cloud renderings. For visualization purposes, colors are inverted in the main paper examples (Fig. 4 in the main paper).

ScanObjectNN. ScanObjectNN [30] has three main variants: object only, object with background, and the PB_T50_RS variant (hardest perturbed variant). Fig. 3 show examples of multi-view renderings of different samples of the dataset from its three variants. Note that adding the background points to the rendering gives some clues to our MVTN about the object, which explains why adding background improves the performance of MVTN in Table 2.

1.2. MVTN Details

MVTN Rendering. Point cloud rendering offers a light alternative to mesh rendering in ShapeNet because its meshes contain large numbers of faces that hinders training the MVTN pipeline. Simplifying theses "high-poly" meshes (similar to ModelNet40) results in corrupted shapes that lose their main visual clues. Therefore, we use point cloud rendering for ShapeNet, allowing to process all shapes with equal memory requirements. Another benefit of point cloud rendering is making it possible to train MVTN with a large batch size on the same GPU (bath size of 30 on V100 GPU).

MVTN Architecture. We incorporate our MVTN into MVCNN [28] and ViewGCN [32]. In our experiments, we select PointNet [23] as the default point encoder of MVTN. All MVTNs and their baseline multi-view networks use ResNet18 [10] as backbone in our main experiments with output feature size $d = 1024$. The azimuth angle maximum range ($\mathbf{u}_{\text{bound}}$) is $\frac{180^\circ}{M}$ for MVTN-circular and MVTN-spherical, while it is 180° for MVTN-direct. On the other hand, the elevation angle maximum range ($\mathbf{u}_{\text{bound}}$) is 90°. We use a 4-layer MLP for MVTN's regression network $\mathbf{G}$. For MVTN-spherical/MVTN-spherical, the regression network takes as input M azimuth angles, M elevation angles, and the point features of shape $\mathbf{S}$ of size $b = 40$. The widths of the MVTN networks are illustrated in Fig. 4.MVTN concatenates all of its inputs, and the MLP outputs the offsets to the initial $2 \times M$ azimuth and elevation angles. The size of the MVTN network (with $b = 40$) is $14M^2 + 211M + 3320$ parameters, where M is the number of views. It is a shallow network of only around 9K parameters when $M = 12$.

View-Points. In Fig. 5, we show the basic views configurations for M views previously used in the literature: circular, spherical, and random. MVTN's learned views are shown later in 3.1 Since ViewGCN uses view sampling as a core operation, it requires the number of views to be at least 12, and hence, our MVTN with ViewGCN follows accordingly.

Training MVTN. We use AdamW [21] for our MVTN networks with a learning rate of 0.001. For other training details (*e.g.* training epochs and optimization), we follow the previous works [32, 28] for a fair comparison. The training of MVTN with MVCNN is done in 100 epochs and a batch size of 20, while the MVTN with ViewGCN is performed in two stages as proposed in the official code of the paper [32]. The first stage is 50 epochs of training the backbone CNN on the single view images, while the second stage is 35 epochs on the multi-view network on the M views of the 3D object. We use learning rates of 0.0003 for MVCNN and 0.001 for ViewGCN, and a ResNet-18 [10] as the backbone CNN for both baselines and our MVTN-based networks. A weight decay of 0.01 is applied for both the multi-view network and in the MVTN networks. Due to gradient instability from the renderer, we introduce gradient clipping in the MVTN to limit the ℓ_2 norm of gradient updates to 30 for$\mathbf{G}$. The code is available at https://github.com/ajhamdi/MVTN.

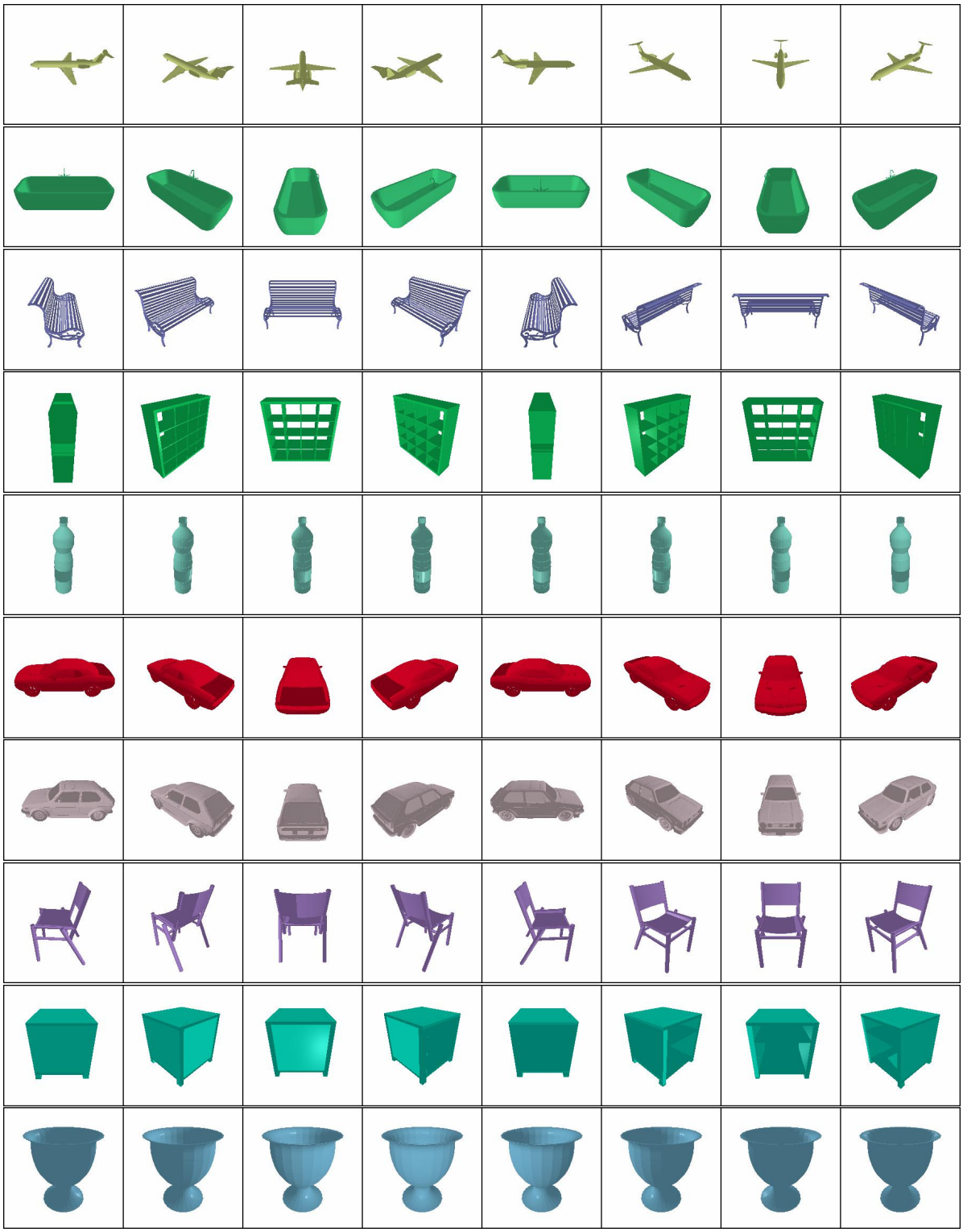

Figure 1. **Training Data with Randomized Color and Lighting.** We show examples of mesh renderings of ModelNet40 used in training our MVTN. The color of the object and the light's direction are randomized during training for augmentation purposes and fixed in testing for stable performance. For this figure, eight circular views are shown for each 3D shape.

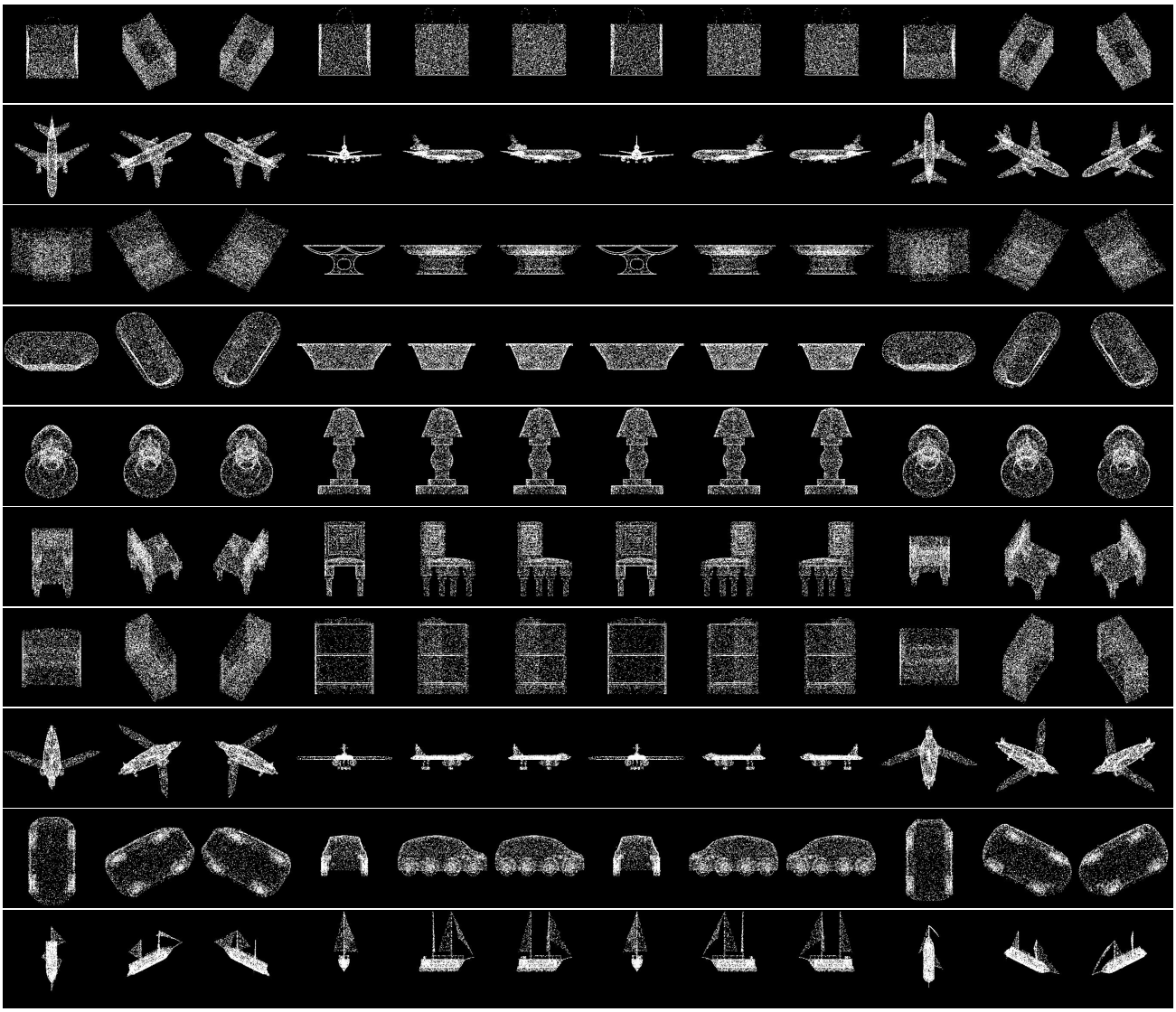

Figure 2. **ShapeNet Core55.** We show some examples of point cloud renderings of ShapeNet Core55 [4] used in training MVTN. Note how point cloud renderings offer more information about content hidden from the camera view-point (*e.g.* car wheels from the occluded side), which can be useful for recognition. For this figure, 12 spherical views are shown for each 3D shape.

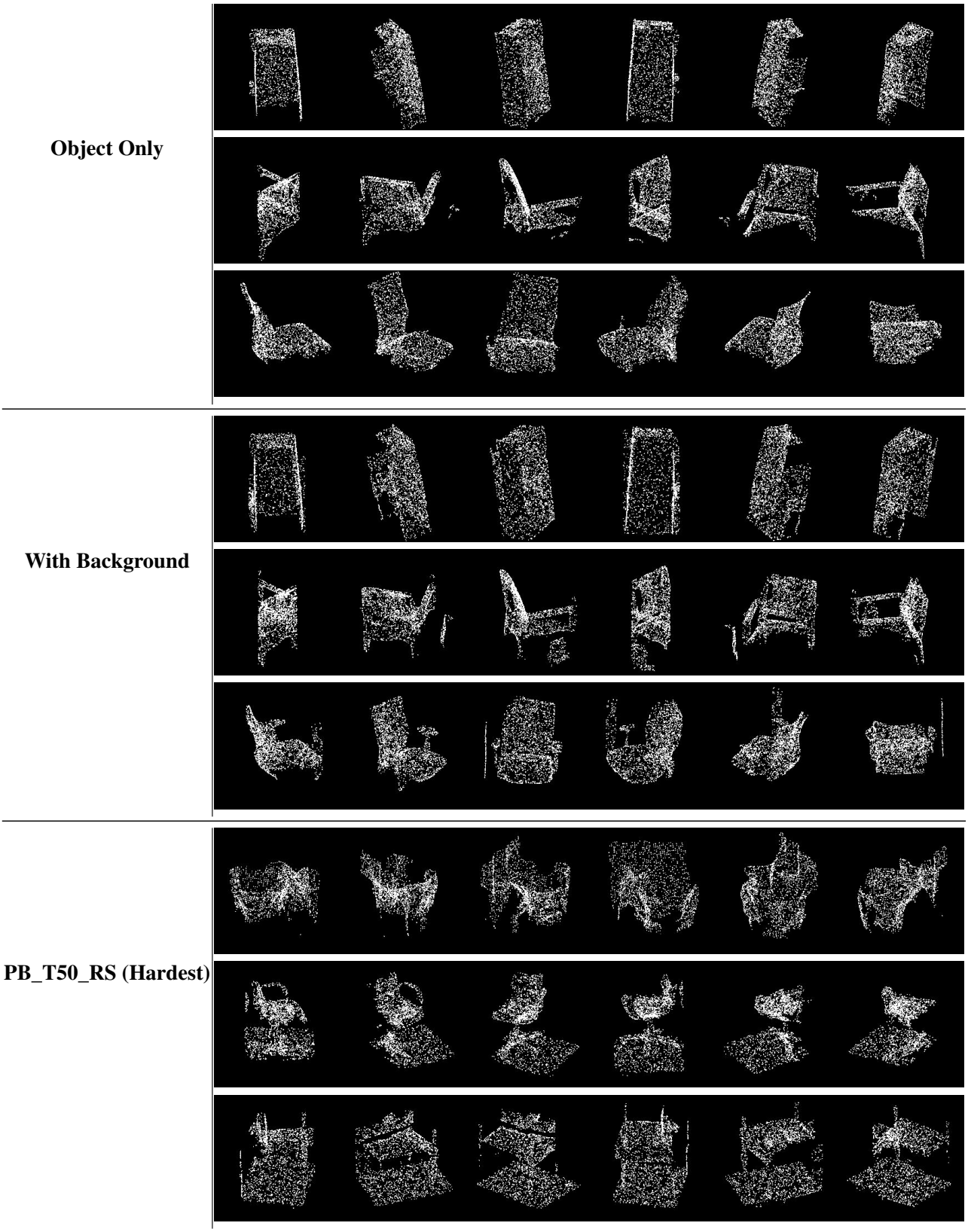

Figure 3. **ScanObjectNN Variants.** We show examples of point cloud renderings of different variants of the ScanObjectNN [30] point cloud dataset used to train MVTN. The variants are: object only, object with background, and the hardest perturbed variant (with rotation and translation). For this figure, six circular views are shown for each 3D shape.

```
MVTN_regressor = Sequential(
    MLP([b+2*M, b, b, 5 *M, 2*M],activation="relu", dropout=0.5, batch_norm=True),
    MLP([2*M, 2*M], activation=None, dropout=0, batch_norm=False),
    nn.Tanh()
    )
```

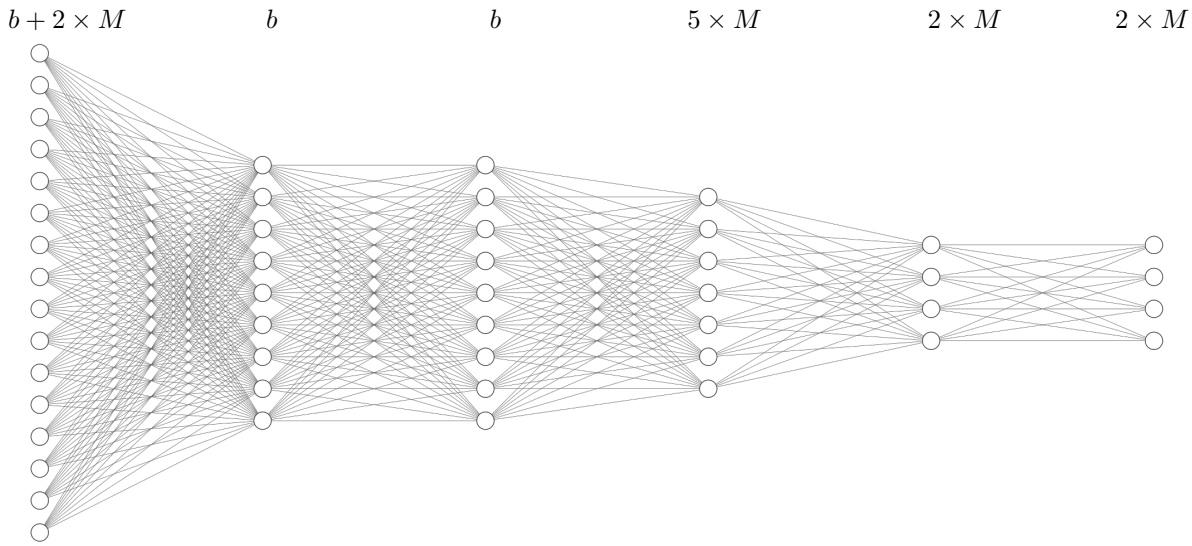

Figure 4. **MVTN Network Architecture.** We show a schematic and a code snippet for MVTN-spherical/MVTN-circular regression architectures used, where b is the size of the point features extracted by the point encoder of MVTN and M is the number of views learned. In most of our experiments, $b = 40$, while the output is the azimuth and elevation angles for all the M views used. The network is drawn using [17]

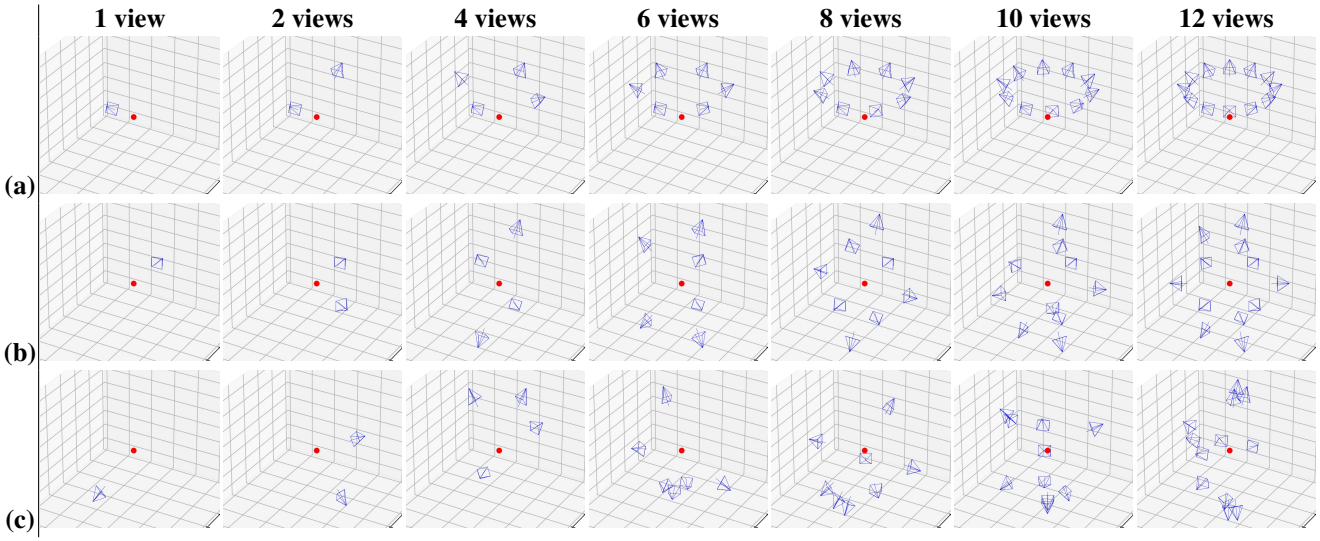

Figure 5. **Views Configurations.** We show some possible views configurations that can be used with a varying number of views. **(a):** circular, **(b):** spherical, **(c):** random

2. Additional Results

2.1. Classification and Retrieval Benchmarks

We provide in Tables 1,2, and 3 comprehensive benchmarks of 3D classifications and 3D shape retrieval methods on ModelNet40 [34], ScanObjectNN [30], and ShapeNet Core55 [4, 27]. These tables include methods that use points as representations as well as other modalities like multi-view and volumetric representations. Our reported results of four runs are presented in each table as "max (avg $\pm$ std)". Note in Table 1 how our MVTN improves the previous state-of-the-art in classification (ViewGCN [32]) when tested on the same setup. Our implementations (highlighted using $*$) slightly differ from the reported results in their original paper. This can be attributed to the specific differentiable renderer of Pytorch3D [26] that we are using, which might not have the same quality of the non-differentiable OpenGL renderings [33] used in their setups.

2.2. Rotation Robustness

A common practice in the literature in 3D shape classification is to test the robustness of models trained on the aligned dataset by injecting perturbations during test time [20]. We follow the same setup as [20] by introducing random rotations during test time around the Y-axis (gravity-axis). We also investigate the effect of varying rotation perturbations on the accuracy of circular MVCNN when $M = 6$ and $M = 12$. We note from Fig. 6 that using less views leads to higher sensitivity to rotations in general. Furthermore, we note that our MVTN helps in stabilizing the performance on increasing thresholds of rotation perturbations.

2.3. Occlusion Robustness

To quantify the occlusion effect due to the viewing angle of the 3D sensor in our setup of 3D classification, we simulate realistic occlusion by cropping the object from canonical directions. We train PointNet [23], DGCNN [31], and MVTN on the ModelNet40 point cloud dataset. Then, at test time, we crop a portion of the object (from 0% occlusion ratio to 75%) along the $\pm$X, $\pm$Y, and $\pm$Z directions independently. Fig. 8 shows examples of this occlusion effect with different occlusion ratios. We report the average test accuracy (on all the test set) of the six cropping directions for the baselines and MVTN in Fig. 7. Note how MVTN achieves high test accuracy even when large portions of the object are cropped. Interestingly, MVTN outperforms PointNet [23] by 13% in test accuracy when half of the object is occluded. This result is significant, given that PointNet is well-known for its robustness [23, 9].

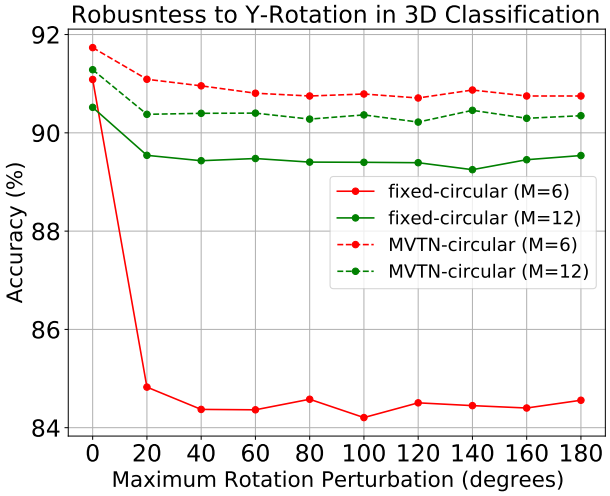

Figure 6. **Robustness on a Varying Y-Rotation**. We study the effect of varying the maximum rotation perturbation on the classification accuracies on ModelNet40. We compare the performance of circular MVCNN [28] to our circular-MVTN when it equips MVCNN when the number of views is 6 and 12. Note how MVTN stabilizes the drop in performance for larger Y-rotation perturbations, and the improvement is more significant for the smaller number of views M.

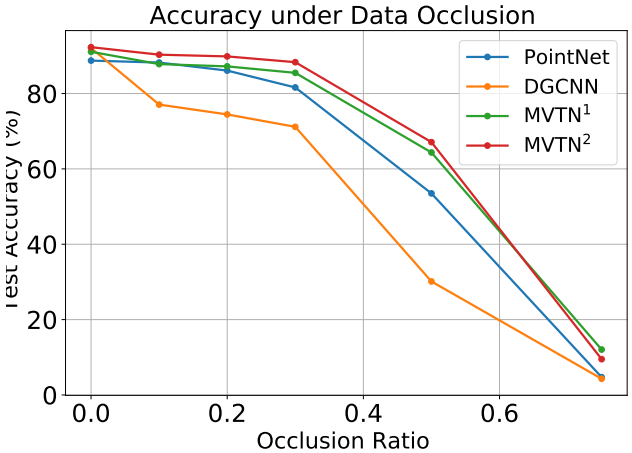

Figure 7. **Occlusion Robustness of 3D Methods.** We plot test accuracy *vs*. the Occlusion Ratio of the data to simulate the occlusion robustness of different 3D methods: PointNet [23], DGCNN [31], and MVTN. Our MVTN achieves close to 13% better than PointNet when half of the object is occluded. MVTN^1 refers to MVTN with MVCNN as the multi-view network while MVTN^2 refers to MVTN with View-GCN as the multi-view network.

Method	Data Type	Classification Accuracy (**Per-Class**)	(**Overall**)
SPH [15]	Voxels	68.2	-
LFD [5]	Voxels	75.5	-
3D ShapeNets [34]	Voxels	77.3	-
VoxNet [22]	Voxels	83.0	85.9
VRN [3]	Voxels	-	91.3
MVCNN-MS [24]	Voxels	-	91.4
FusionNet [11]	Voxels+MV	-	90.8
PointNet [23]	Points	86.2	89.2
PointNet++ [25]	Points	-	91.9
KD-Network [16]	Points	88.5	91.8
PointCNN [19]	Points	88.1	91.8
DGCNN [31]	Points	90.2	92.2
KPConv[29]	Points	-	92.9
PVNet[36]	Points	-	93.2
PTransformer[37]	Points	**90.6**	**93.7**
MVCNN [28]	12 Views	90.1	90.1
GVCNN [6]	12 Views	90.7	93.1
ViewGCN [32]	20 Views	**96.5**	**97.6**
ViewGCN [32]*	12 views	90.7 (90.5 $\pm$ 0.2)	93.0 (92.8 $\pm$ 0.1)
ViewGCN [32]*	20 views	91.3 (91.0 $\pm$ 0.2)	93.3 (93.1 $\pm$ 0.2)
MVTN (ours)*	12 Views	92.0 (91.2 $\pm$ 0.6)	**93.8** (93.4 $\pm$ 0.3)
MVTN (ours)*	20 Views	**92.2** (91.8 $\pm$ 0.3)	93.5 (93.1 $\pm$ 0.5)

Table 1. **3D Shape Classification on ModelNet40**. We compare MVTN against other methods in 3D classification on ModelNet40 [34]. * indicates results from our rendering setup (differentiable pipeline), while other multi-view results are reported from pre-rendered views. **Bold** denotes the best result in its setup. In brackets, we report the average and standard deviation of four runs

Method	Classification Overall Accuracy **Object with Background**	**Object Only**	**PB_T50_RS (Hardest)**
3DMFV [2]	68.2	73.8	63.0
PointNet [23]	73.3	79.2	68.0
SpiderCNN [35]	77.1	79.5	73.7
PointNet ++ [25]	82.3	84.3	77.9
PointCNN [19]	86.1	85.5	78.5
DGCNN [31]	82.8	86.2	78.1
SimpleView [8]	-	-	79.5
BGA-DGCNN [30]	-	-	79.7
BGA-PN++ [30]	-	-	80.2
ViewGCN *	91.9 (91.12 $\pm$ 0.5)	90.4 (89.7 $\pm$ 0.5)	80.5 (80.2 $\pm$ 0.4)
MVTN (ours)	**92.6** (92.5 $\pm$ 0.2)	**92.3** (91.7 $\pm$ 0.7)	**82.8** (81.8 $\pm$ 0.7)

Table 2. **3D Point Cloud Classification on ScanObjectNN**. We compare the performance of MVTN in 3D point cloud classification on three different variants of ScanObjectNN [30]. The variants include object with background, object only, and the hardest variant. * indicates results from our rendering setup (differentiable pipeline), and we report the average and standard deviation of four runs in brackets.

Method	Data Type	Shape Retrieval (mAP) **ModelNet40**	**ShapeNet Core**
ZDFR [18]	Voxels	-	19.9
DLAN [7]	Voxels	-	66.3
SPH [15]	Voxels	33.3	-
LFD [5]	Voxels	40.9	-
3D ShapeNets [34]	Voxels	49.2	-
PVNet[36]	Points	89.5	-
MVCNN [28]	12 Views	80.2	73.5
GIFT [1]	20 Views	-	64.0
MVFusionNet [12]	12 Views	-	62.2
ReVGG [27]	20 Views	-	74.9
RotNet [14]	20 Views	-	77.2
ViewGCN [32]	20 Views	-	78.4
MLVCNN [13]	24 Views	92.2	-
MVTN (ours)	12 Views	**92.9** (92.4 ± 0.6)	**82.9** (82.4 ± 0.6)

Table 3. **3D Shape Retrieval**. We benchmark the shape retrieval capability of MVTN on ModelNet40 [34] and ShapeNet Core55 [4, 27]. MVTN achieves the best retrieval performance among recent state-of-the-art methods on both datasets with only 12 views. In brackets, we report the average and standard deviation of four runs.

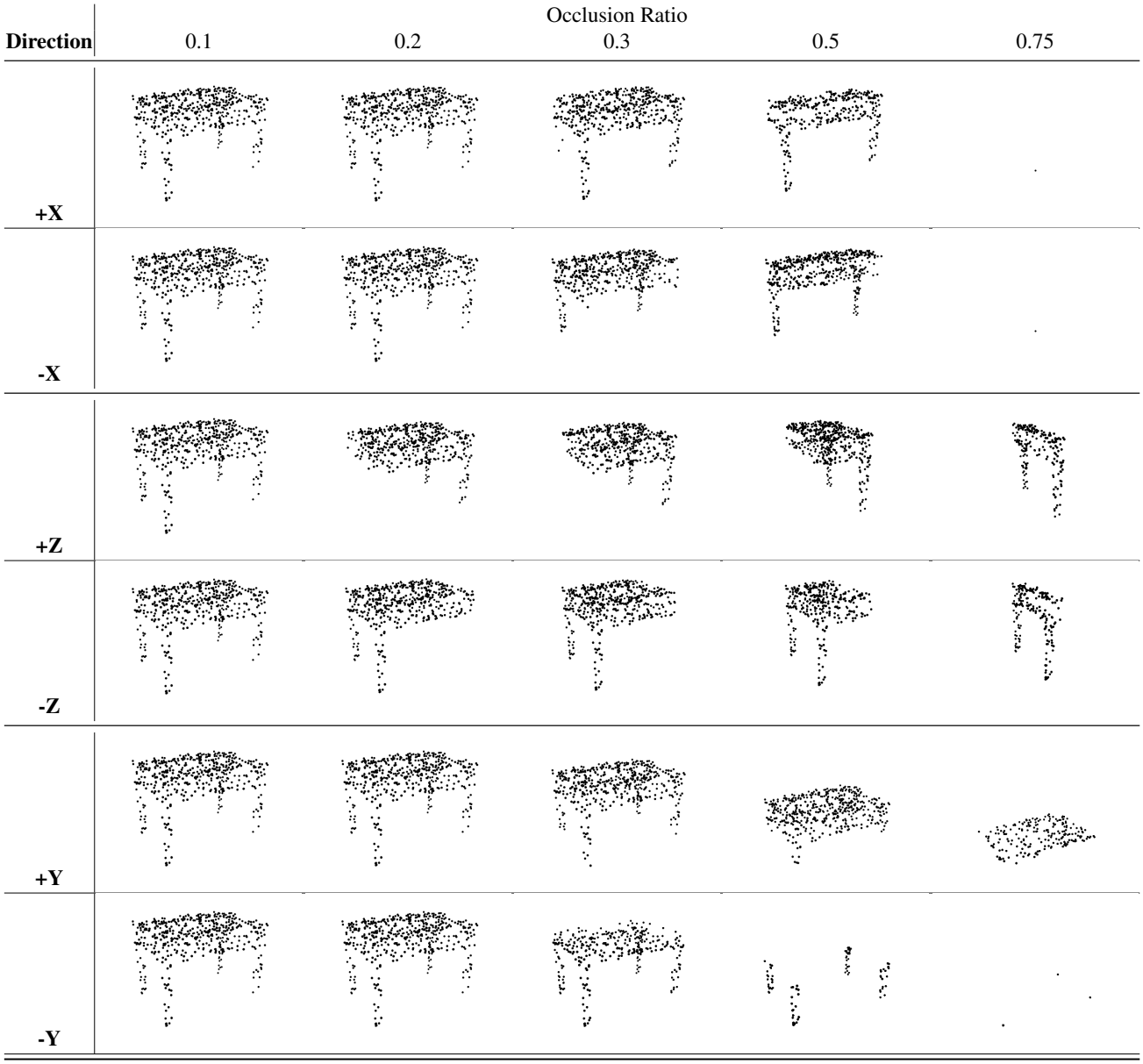

Figure 8. **Occlusion of 3D Objects**: We simulate realistic occlusion scenarios in 3D point clouds by cropping a percentage of the object along canonical directions. Here, we show an object occluded with different ratios and from different directions.

3. Analysis and Insights

3.1. Ablation Study

This section introduces a comprehensive ablation study on the different components of MVTN, and their effect on test accuracy on the standard ModelNet40 [34].

MVTN Variants. We study the effect of the number of views M on the performance of different MVTN variants (direct, circular, spherical). The experiments are repeated four times, and the average test accuracies with confidence intervals are shown in Fig. 9. The plots show how learned MVTN-spherical achieves consistently superior performance across a different number of views. Also, note that MVTN-direct suffers from over-fitting when the number of views is larger than four (*i.e.* it gets perfect training accuracy but deteriorates in test accuracy). This can be explained by observing that the predicted view-points tend to be similar to each other for MVTN-direct when the number of views is large. The similarity in views leads the multi-view network to memorize the training but to suffer in testing.

Backbone. In the main manuscript (Table 6), we study MVTN with ViewGCN as the multi-view network. Here, we study the backbone effect on MVTN with MVCNN as the multi-view network and report all results in Table 6. The study includes the backbone choice, and the point encoder choice. Note that including more sophisticated backbones does not improve the accuracy

Late Fusion. In the MVTN pipeline, we use a point encoder and a multi-view network. One can argue that an easy way to combine them would be to fuse them later in the architecture. For example, PointNet [23] and MVCNN [28] can be max pooled together at the last layers and trained jointly. We train such a setup and compare it to MVTN. We observe that MVTN achieves 91.8% compared to 88.4% by late fusion. More results are reported in Table 6

Light Direction Effect. We study the effect of light's direction on the performance of multi-view networks. We note that picking a random light in training helps the network generalize to the test set. Please see Fig. 10 for the results on circular MVTN with MVCNN when comparing this strategy to fixed light from the top or from camera (*relative*). Note that we use relative light in test time to stabilize the performance.

Effect of Object Color. Our main experiments used random colors for the objects during training and fixed them to white in testing. We tried different coloring approaches, like using a fixed color during training and test. The results are illustrated in Table 4.

Image size and number of points. We study the effect of rendered image size and the number of points sampled in a 4-view MVTN trained on ModelNet40 and report the overall accuracies (averaged over four runs) as follows. For image sizes 160×160, 224×224, and 280×280, the results are 91.0%, 91.6%, and 91.9% respectively. For the number of randomly sampled points $P = 512$, 1024, and 2048, the results are 91.2% 91.6% , and 91.6% respectively.

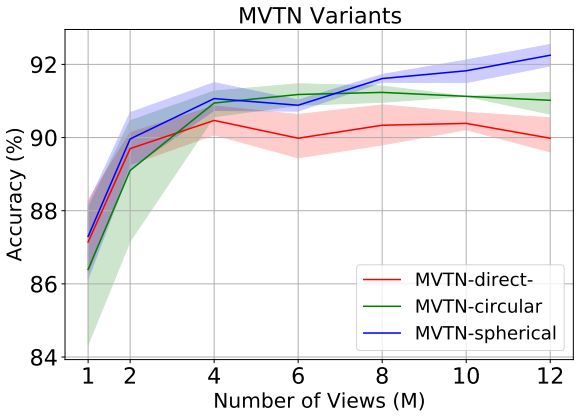

Figure 9. **Variants of MVTN.** We plot test accuracy vs. the number of views used in training different variants of our MVTN. Note how MVTN-spherical is generally more stable in achieving better performance on ModelNet40. 95% confidence interval is also plotted on each setup (repeated four times).

Learning Distance to the Object. One possible ablation to the MVTN is to learn the distance to the object. This feature should allow the cameras to get closer to details that might be important to the classifier to understand the object properly. However, we observe that MVTN generally performs worse or does not improve with this setup, and hence, we refrain from learning it. In all of our main experiments, we fixed the distance to 2.2 units, which is a good middle ground providing best accuracy. Please see Fig. 11 for the effect of picking a fixed distance in training spherical ViewGCN.

3.2. Time and Memory of MVTN

We compare the time and memory requirements of different parts of our pipeline to assess the MVTN module's contribution. We record FLOPs and MACs to count each module's operations and record the time of a forward pass for a single input sample and the number of parameters for each module. We find in Table 5 that MVTN contributes negligibly to the time and memory requirements of the multi-view networks and the 3D point encoders.

3.3. Transferability of MVTN View-Points

We hypothesize that the views learned by MVTN are transferable across multi-view classifiers. Looking at results in Fig. 13, 14, we believe MVTN picks the best views based on the actual shape and is less influenced by the multi-view network. This means that MVTN learns views that are more representative of the object, making it easier for *any* multi-view network to recognize it. As such, we ask the following: *can we transfer the views MVTN learns under one setting to*

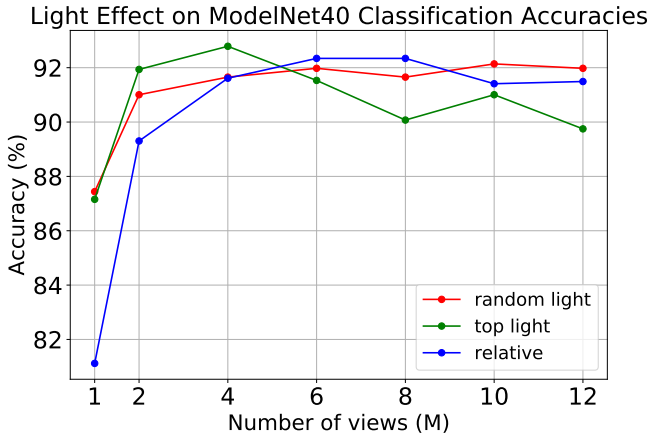

Figure 10. **Light Direction Effect**. We study the effect of light direction in the performance of the MVTN-circular. We note that randomizing the light direction in training reduce overfitting for larger number of views and leads to better generalization.

Method	Object Color	
	White	Random
Fixed views	92.8 ± 0.1	92.8 ± 0.1
MVTN (learned)	93.3 ± 0.1	**93.4** ± 0.1

Table 4. **Effect of Color Selection**. We ablate selecting the color of the object in training our MVTN and when views are fixed in the spherical configuration. Fixed white color is compared to random colors in training. Note how randomizing the color helps in improving the test accuracy on ModelNet40 a little bit.

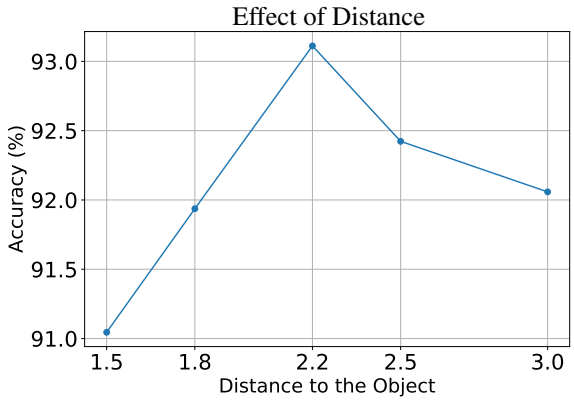

Figure 11. **Effect of Distance to 3D Object**. We study the effect of changing the distance on training a spherical ViewGCN. We show that the distance of 2.2 units to the center is in between far and close it and gives the best accuracy.

a different multi-view network?

To test our hypothesis, we take a 12-view MVTN-spherical module trained with MVCNN as a multi-view network and transfer the predicted views to a ViewGCN multi-view network. In this case, we freeze the MVTN module and only train ViewGCN on these learned but fixed views. ViewGCN with transferred MVTN views reaches 93.1% accuracy in classification. It corresponds to a boost of 0.7% from the 92.4% of the original ViewGCN. Although this result is lower than fully trained MVTN(-0.3%), we observe a decent transferability between both multi-view architectures.

Network	FLOPs	MACs	Params. #	Time
PointNet	1.78 G	0.89 G	3.49 M	3.34 ms
DGCNN	10.42 G	5.21 G	0.95 M	16.35 ms
MVCNN	43.72 G	21.86 G	11.20 M	39.89 ms
ViewGCN	44.19 G	22.09 G	23.56 M	26.06 ms
MVTN*	**18.52 K**	**9.26 K**	**9.09 K**	**0.9 ms**
MVTN°	1.78 G	0.89 G	4.24 M	3.50 ms

Table 5. **Time and Memory Requirements**. We assess the contribution of the MVTN module to time and memory requirements in the multi-view pipeline. MVTN* refers to MVTN's regressor excluding the point encoder, while MVTN° refers to the full MVTN module including PointNet as a point encoder.

3.4. MVTN Predicted Views

We visualize the distribution of predicted views by MVTN for specific classes in Fig. 12. This is done to ensure that MVTN is learning per-instance views and regressing the same views for the entire class (collapse scenario). We can see that the MVTN distribution of the views varies from one class to another, and the views themselves on the same class have some variance from one instance to another. We also show specific examples for predicted views in Fig. 13, 14. Here, we show both the predicted camera view-points and the renderings from these cameras. Note how MVTN shifts every view to better show the discriminative details about the 3D object. To test that these views are per-instance, we average all the views predicted by our 4-view MVTN for every class and test the trained MVCNN on these fixed per-class views. In this setup, MVTN achieves 90.6% on ModelNet40, as compared to 91.0% for the per-instance views and 89% for the fixed views.

3.5. Shape Retrieval Examples

We show qualitative examples of our retrieval results using the MVTN-spherical with ViewGCN in Fig. 15. Note that the top ten retrieved objects for all these queries are positive (from the same classes of the queries).

Views number	Backbone ResNet18	Backbone ResNet50	Point Encoder PointNet[23]	Point Encoder DGCNN[31]	Setup circular	Setup spherical	Fusion late	Fusion MVTN	Results accuracy
6	✓	-	✓	-	✓	-	✓	-	90.48 %
6	✓	-	✓	-	✓	-	-	✓	91.13 %
6	✓	-	✓	-	-	✓	✓	-	89.51 %
6	✓	-	✓	-	-	✓	-	✓	91.94 %
6	✓	-	-	✓	✓	-	✓	-	87.80 %
6	✓	-	-	✓	✓	-	-	✓	91.49 %
6	✓	-	-	✓	-	✓	✓	-	89.82 %
6	✓	-	-	✓	-	✓	-	✓	91.29 %
6	-	✓	✓	-	✓	-	✓	-	89.10 %
6	-	✓	✓	-	✓	-	-	✓	90.40 %
6	-	✓	✓	-	-	✓	✓	-	89.22 %
6	-	✓	✓	-	-	✓	-	✓	90.76 %
6	-	✓	-	✓	✓	-	✓	-	89.99 %
6	-	✓	-	✓	✓	-	-	✓	89.91 %
6	-	✓	-	✓	-	✓	✓	-	89.95 %
6	-	✓	-	✓	-	✓	-	✓	90.43 %
12	✓	-	✓	-	✓	-	✓	-	87.35%
12	✓	-	✓	-	✓	-	-	✓	90.68%
12	✓	-	✓	-	-	✓	✓	-	88.41%
12	✓	-	✓	-	-	✓	-	✓	91.82
12	✓	-	-	✓	✓	-	✓	-	90.24%
12	✓	-	-	✓	✓	-	-	✓	90.28%
12	✓	-	-	✓	-	✓	✓	-	89.83%
12	✓	-	-	✓	-	✓	-	✓	91.98%
12	-	✓	✓	-	✓	-	✓	-	86.87%
12	-	✓	✓	-	✓	-	-	✓	88.86%
12	-	✓	✓	-	-	✓	✓	-	87.16%
12	-	✓	✓	-	-	✓	-	✓	88.41%
12	-	✓	-	✓	✓	-	✓	-	90.15%
12	-	✓	-	✓	✓	-	-	✓	88.37%
12	-	✓	-	✓	-	✓	✓	-	90.48%
12	-	✓	-	✓	-	✓	-	✓	89.63%

Table 6. **Ablation Study**. We study the effect of ablating different components of MVTN on the test accuracy on ModelNet40. Namely, we observe that using more complex backbone CNNs (like ResNet50 [10]) or a more complex features extractor (like DGCNN [31]) does not increase the performance significantly compared to ResNet18 and PointNet [23] respectively. Furthermore, combining the shape features extractor with the MVCNN [28] in *late fusion* does not work as well as MVTN with the same architectures. All the reported results are using MVCNN [28] as multi-view network.

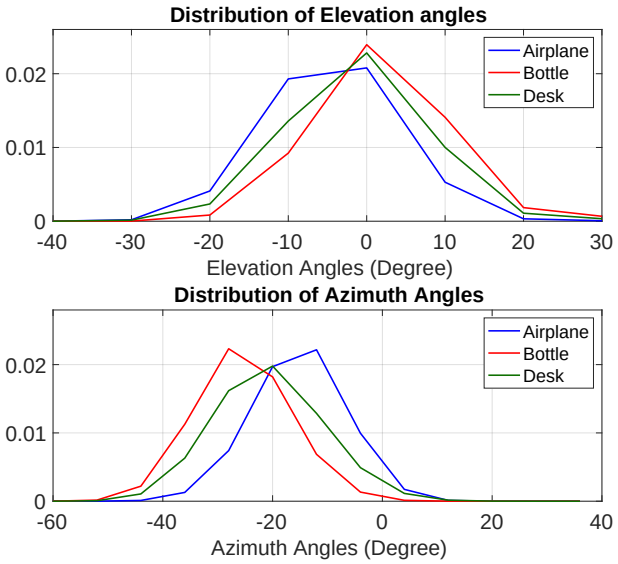

Figure 12. **Visualizing MVTN learned Views.** We visualize the distribution of azimuth and elevation angles predicted by the MVTN for three different classes. Note that MVTN learns inter-class variations (between different classes) and intra-class variations (on the same class).

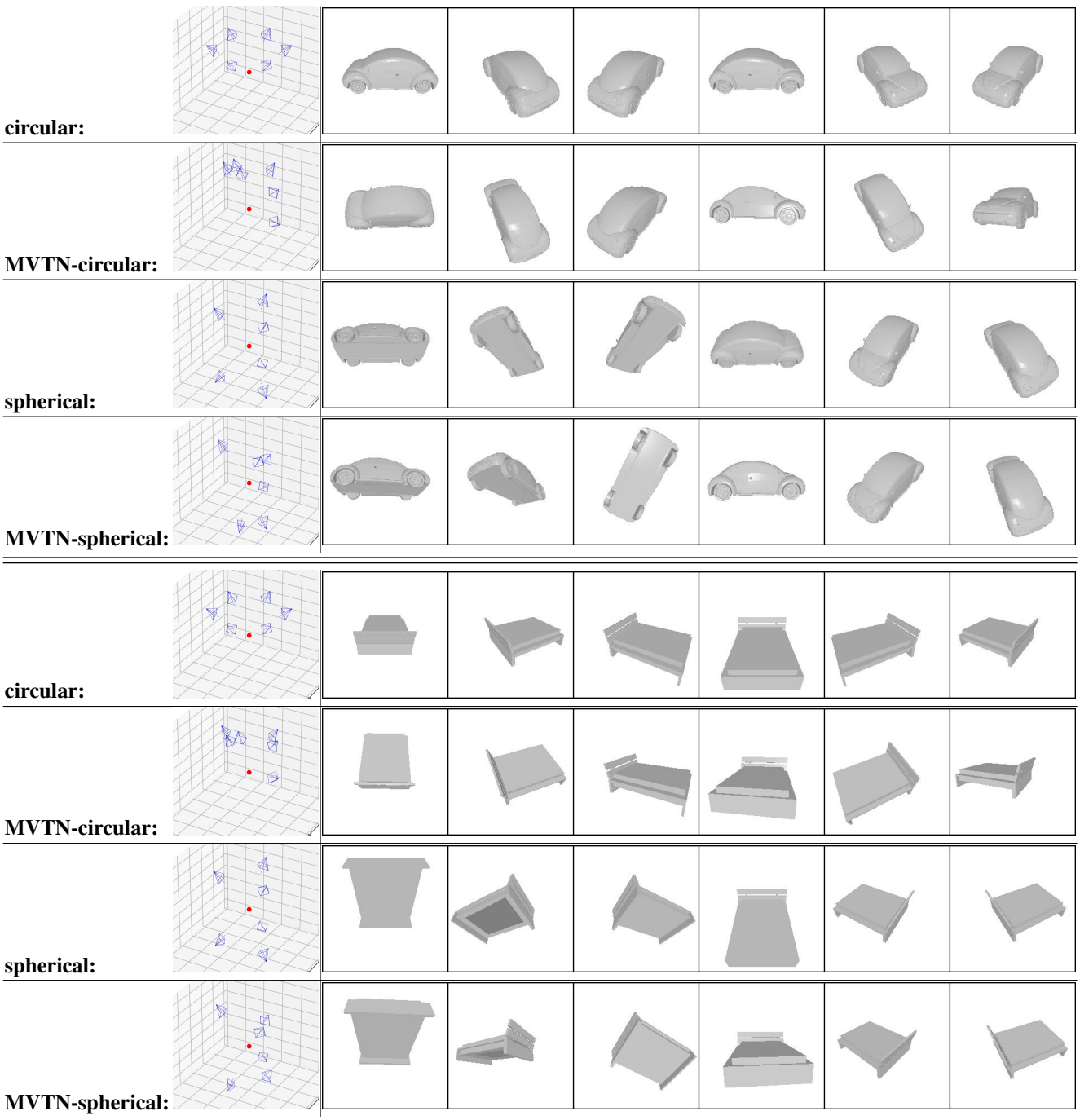

Figure 13. **Qualitative Examples for MVTN predicted views (I)**: The view setups commonly followed in the multi-view literature are circular [28] or spherical [32, 14]. The red dot is the center of the object. MVTN-circular/MVTN-spherical are trained to predict the views as offsets to these common configurations. Note that MVTN adjust the original views to make the 3D object better represented by the multi-view images.

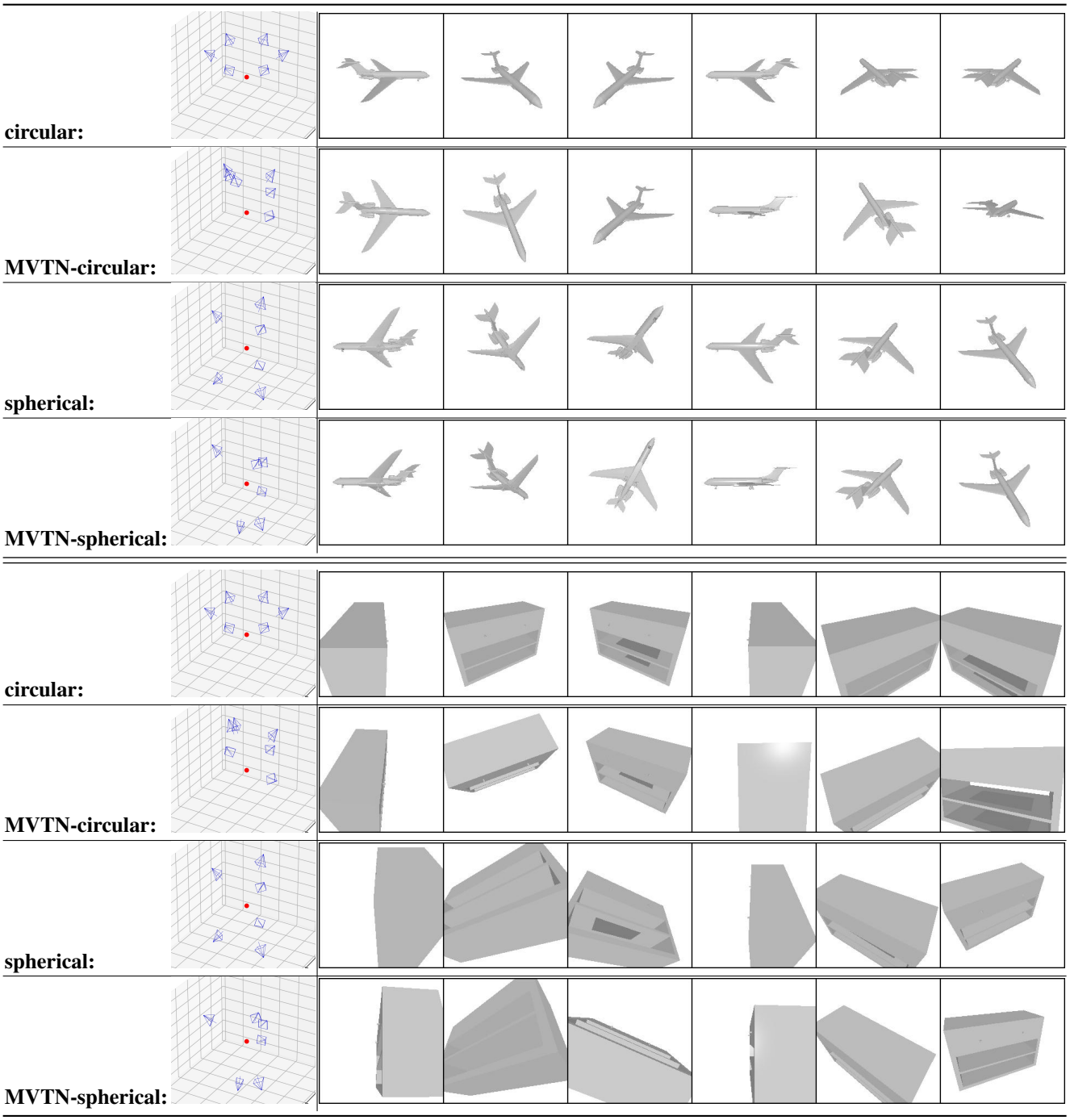

Figure 14. **Qualitative Examples for MVTN predicted views (II)**: The view setups commonly followed in the multi-view literature are circular [28] or spherical [32, 14]. The red dot is the center of the object. MVTN-circular/MVTN-spherical are trained to predict the views as offsets to these common configurations. Note that MVTN adjust the original views to make the 3D object better represented by the multi-view images.

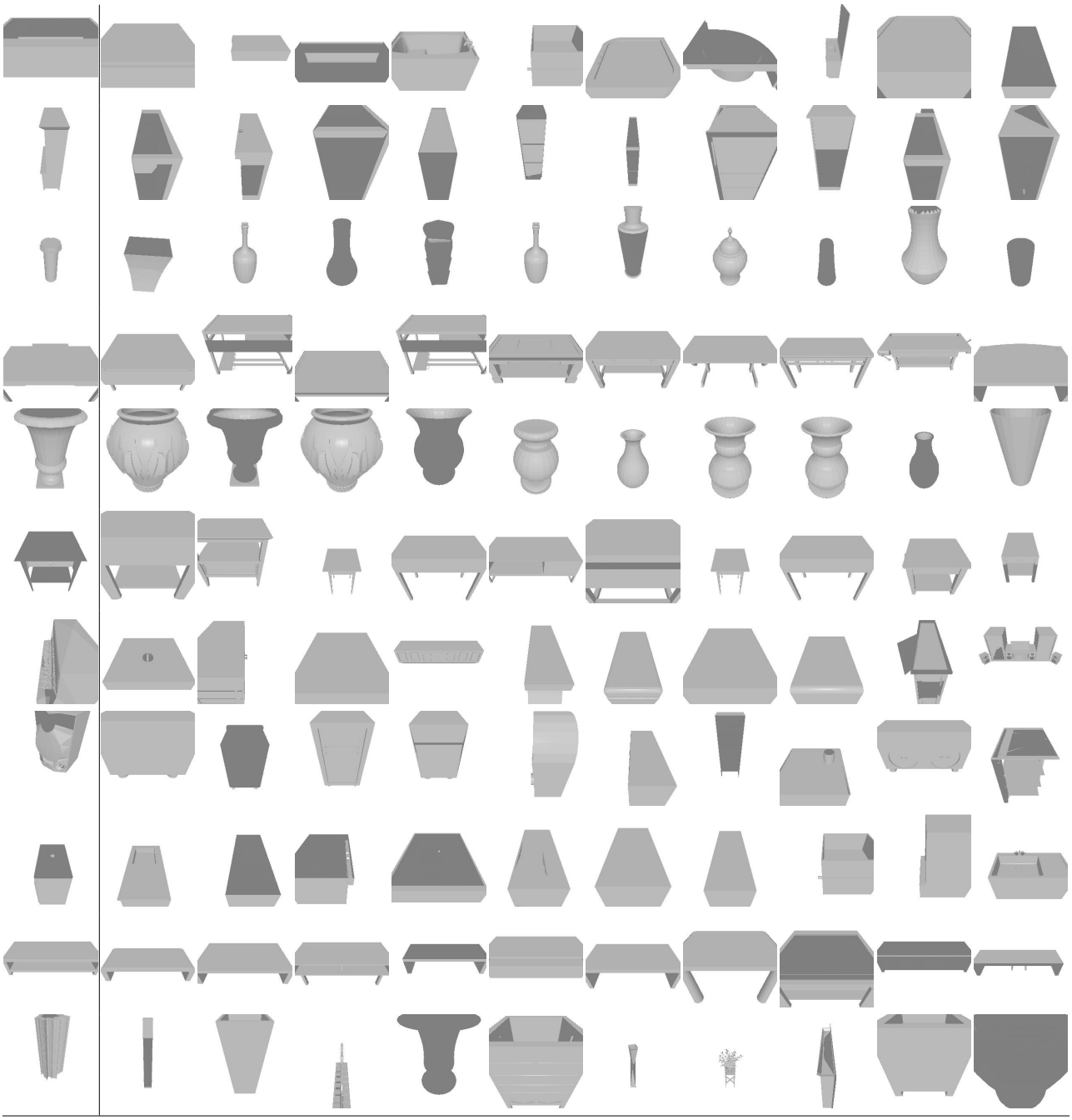

Figure 15. **Qualitative Examples for Object Retrieval**: *(left):* we show some query objects from the test set. *(right)*: we show top ten retrieved objects by our MVTN from the training set.

References

[1] Song Bai, Xiang Bai, Zhichao Zhou, Zhaoxiang Zhang, and Longin Jan Latecki. Gift: A real-time and scalable 3d shape search engine. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5023–5032, 2016. 8

[2] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152, 2018. 7

[3] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016. 7

[4] 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. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1, 3, 6, 8

[5] Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On visual similarity based 3d model retrieval. In *Computer graphics forum*, volume 22, pages 223–232. Wiley Online Library, 2003. 7, 8

[6] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 264–272, 2018. 7

[7] Takahiko Furuya and Ryutarou Ohbuchi. Deep aggregation of local 3d geometric features for 3d model retrieval. In *BMVC*, volume 7, page 8, 2016. 8

[8] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *ICML*, 2021. 7

[9] Abdullah Hamdi, Sara Rojas, Ali Thabet, and Bernard Ghanem. Advpc: Transferable adversarial perturbations on 3d point clouds. In *Computer Vision – ECCV 2020*, pages 241–257, Cham, 2020. Springer International Publishing. 6

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1, 12

[11] Vishakh Hegde and Reza Zadeh. Fusionnet: 3d object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016. 7

[12] Kui Jia, Jiehong Lin, Mingkui Tan, and Dacheng Tao. Deep multi-view learning using neuron-wise correlation-maximizing regularizers. *IEEE Transactions on Image Processing*, 28(10):5121–5134, 2019. 8

[13] Jianwen Jiang, Di Bao, Ziqiang Chen, Xibin Zhao, and Yue Gao. Mlvcnn: Multi-loop-view convolutional neural network for 3d shape retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8513–8520, 2019. 8

[14] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018. 8, 14, 15

[15] Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3 d shape descriptors. In *Symposium on geometry processing*, volume 6, pages 156–164, 2003. 7, 8

[16] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 7

[17] Alex Lenail. Nn-svg, 2020. 5

[18] Bo Li and Henry Johan. 3d model retrieval using hybrid features and class information. *Multimedia tools and applications*, 62(3):821–846, 2013. 8

[19] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems (NIPS)*, pages 820–830, 2018. 7

[20] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019. 6

[21] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[22] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 7

[23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 1, 6, 7, 10, 12

[24] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5648–5656, 2016. 7

[25] 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 (NIPS)*, pages 5099–5108, 2017. 7

[26] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 6

[27] Konstantinos Sfikas, Theoharis Theoharis, and Ioannis Pratikakis. Exploiting the PANORAMA Representation for Convolutional Neural Network Classification and Retrieval. In Ioannis Pratikakis, Florent Dupont, and Maks Ovsjanikov, editors, *Eurographics Workshop on 3D Object Retrieval*, pages 1–7. The Eurographics Association, 2017. 1, 6, 8

[28] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks

for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 1, 6, 7, 8, 10, 12, 14, 15

[29] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6411–6420, 2019. 7

[30] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 4, 6, 7

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 6, 7, 12

[32] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1850–1859, 2020. 1, 6, 7, 8, 14, 15

[33] Mason Woo, Jackie Neider, Tom Davis, and Dave Shreiner. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1*. Addison-wesley, 1998. 6

[34] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 6, 7, 8, 10

[35] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 7

[36] Haoxuan You, Yifan Feng, Rongrong Ji, and Yue Gao. Pvnet: A joint convolutional network of point cloud and multi-view for 3d shape recognition. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1310–1318, 2018. 7, 8

[37] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020. 7