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

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

Abstract

Multi-view projection methods have demonstrated their ability to reach state-of-the-art performance on 3D shape recognition. Those methods learn different ways to aggregate information from multiple views. However, the camera view-points for those views tend to be heuristically set and fixed for all shapes. To circumvent the lack of dynamism of current multi-view methods, we propose to learn those view-points. In particular, we introduce the Multi-View Transformation Network (MVTN) that regresses optimal view-points for 3D shape recognition, building upon advances in differentiable rendering. As a result, MVTN can be trained end-to-end along with any multi-view network for 3D shape classification. We integrate MVTN in a novel adaptive multi-view pipeline that can render either 3D meshes or point clouds. MVTN exhibits clear performance gains in the tasks of 3D shape classification and 3D shape retrieval without the need for extra training supervision. In these tasks, MVTN achieves state-of-the-art performance on ModelNet40, ShapeNet Core55, and the most recent and realistic ScanObjectNN dataset (up to 6% improvement). Interestingly, we also show that MVTN can provide network robustness against rotation and occlusion in the 3D domain. The code is available at `https://github.com/ajhamdi/MVTN`.

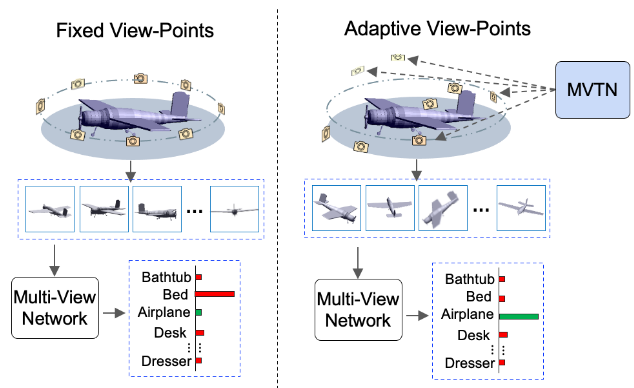

Figure 1. **Multi-View Transformation Network (MVTN).** We propose a differentiable module that predicts the best view-points for a task-specific multi-view network. MVTN is trained jointly with this network without any extra training supervision, while improving the performance on 3D classification and shape retrieval.

1. Introduction

Given its success in the 2D realm, deep learning naturally expanded to the 3D vision domain. In 3D, deep networks achieve impressive results in classification, segmentation, and detection. 3D deep learning pipelines operate directly on 3D data, commonly represented as point clouds [55, 57, 66], meshes [18, 29], or voxels [52, 13, 24]. However, other methods choose to represent 3D information by rendering multiple 2D views of objects or scenes [61]. Such multi-view methods are more similar to a human approach, where the human visual system is fed with streams of rendered images instead of more elaborate 3D representations.

Recent developments in multi-view methods show impressive performance, and in many instances, achieve state-of-the-art results in 3D shape classification and segmentation [38, 67, 41, 37, 15]. Multi-view approaches bridge the gap between 2D and 3D learning by solving a 3D task using 2D convolutional architectures. These methods render several views for a given 3D shape and leverage the rendered images to solve the end task. As a result, they build upon the recent advances in 2D grid-based deep learning and leverage larger image datasets for pre-training (*e.g.* ImageNet [59]) to compensate for the general scarcity of labeled 3D datasets. However, the manner of choosing the rendering view-points for such methods remains mostly unexplored. Current methods rely on heuristics like random sampling in scenes [41] or predefined canonical view-points in oriented datasets [67]. There is no evidence suggesting that such heuristics are empirically the best choice. To address this shortcoming, we propose to learn better view-points by introducing a Multi-View Transformation Network (MVTN). As shown in Fig. 1, MVTN learns to regress view-points, renders those views with a differentiable renderer, and trains the downstream task-specific network in an end-to-end fashion, thus leading to the most suitable views for the task. MVTN is inspired by the Spatial Transformer Network (STN) [32], which was developed for the 2D image domain. Both MVTN and STN learn spatial transformations for the input without leveraging any extra supervision nor adjusting the learning process.

The paradigm of perception by predicting the best environment parameters that generated the image is called Vision as Inverse Graphics (VIG) [25, 40, 70, 36, 77]. One approach to VIG is to make the rendering process invertible or differentiable [51, 39, 47, 12, 45]. In this paper, MVTN takes advantage of differentiable rendering [39, 47, 58]. With such a renderer, models can be trained end-to-end for a specific target 3D vision task, with the view-points (*i.e*. camera poses) being inferred by MVTN in the same forward pass. To the best of our knowledge, we are the first to integrate a learnable approach to view-point prediction in multi-view methods by using a differentiable renderer and establishing an end-to-end pipeline that works for *both* mesh and 3D point cloud classification and retrieval.

Contributions: (i) We propose a Multi-View Transformation Network (MVTN) that regresses better view-points for multi-view methods. Our MVTN leverages a differentiable renderer that enables end-to-end training for 3D shape recognition tasks. **(ii)** Combining MVTN with multi-view approaches leads to state-of-the-art results in 3D classification and shape retrieval on standard benchmarks ModelNet40 [71], ShapeNet Core55 [7, 60], and ScanObjectNN [64]. **(iii)** Additional analysis shows that MVTN improves the robustness of multi-view approaches to rotation and occlusion, making MVTN more practical for realistic scenarios, where 3D models are not perfectly aligned or partially cropped.

2. Related Work

Deep Learning on 3D Data. PointNet [55] paved the way as the first deep learning algorithm to operate directly on 3D point clouds. PointNet computes point features independently and aggregates them using an order invariant function like max-pooling. Subsequent works focused on finding neighborhoods of points to define point convolutional operations [57, 66, 46, 43, 42, 65]. Voxel-based deep networks allow for 3D CNNs yet suffer from cubic memory complexity [52, 13, 24]. Several recent works combine point cloud representations with other 3D modalities like voxels [50] or multi-view images [75, 33]. In this paper, we leverage a point encoder to predict the optimal view-points, from which images are rendered and fed to a multi-view network.

Multi-View 3D Shape Classification. The first work on using 2D images to recognize 3D objects was proposed by Bradski *et al*. [5]. Twenty years later and after the success of deep learning in 2D vision tasks, MVCNN [61] emerged as the first use of deep 2D CNNs for 3D object recognition. The original MVCNN uses max pooling to aggregate features from different views. Several follow-up works propose different strategies to assign weights to views to perform weighted average pooling of view-specific features [76, 74, 19, 11]. RotationNet [38] classifies the views and the object jointly. Equivariant MV-Network [17] uses a rotation equivariant convolution operation on multi-views by utilizing rotation group convolutions [14]. The more recent work of ViewGCN [67] utilizes dynamic graph convolution operations to adaptively pool features from different fixed views for the task of 3D shape classification. All these previous methods rely on fixed rendered datasets of 3D objects. The work of [11] attempts to select views adaptively through reinforcement learning and RNNs, but this comes with limited success and an elaborate training process. In this paper, we propose a novel MVTN framework for predicting optimal view-points in a multi-view setup. This is done by jointly training MVTN with a multi-view task-specific network, without the need for any extra supervision nor adjustment to the learning process.

3D Shape Retrieval. Early methods in the literature compare the distribution of hand-crafted descriptors to retrieve similar 3D shapes. Those shape signatures could either represent geometric [53] or visual [9] cues. Traditional geometric methods would estimate distributions of certain characteristics (*e.g*. distances, angles, areas, or volumes) to measure the similarity between shapes [1, 8, 6]. Gao *et al*. [21] use multiple camera projections, and Wu *et al*. [72] use a voxel grid to extract analogous model-based signatures. Su *et al*. [61] introduce a deep learning pipeline for multi-view classification, with aggregated features achieving high retrieval performance. They use a low-rank Mahalanobis metric atop extracted multi-view features to improve retrieval performance. This seminal work on multi-view learning is extended for retrieval with volumetric-based descriptors [56], hierarchical view-group architectures [19], and triplet-center loss [31]. Jiang *et al*. [35] investigate better views for retrieval using many loops of circular cameras around the three principal axes. However, these approaches consider fixed camera view-points compared to MVTN's learnable ones.

Vision as Inverse Graphics (VIG). A key issue in VIG is the non-differentiability of the classical graphics pipeline. Recent VIG approaches focus on making the graphics operations differentiable, allowing gradients to flow from the image to the rendering parameters directly [51, 39, 47, 45, 26]. NMR [39] approximates non-differentiable rasterization by smoothing the edge rendering, where SoftRas [47] assigns a probability for all mesh triangles to every pixel in the image. Synsin [68] proposes an alpha-blending mechanism for differentiable point cloud rendering. The Pytorch3D [58] renderer improves the speed and modularity of SoftRas and Synsin and allows for customized shaders and point cloud rendering. MVTN harnesses advances in differentiable rendering to train jointly with the multi-view network in an end-to-end fashion. Using both mesh and point cloud differentiable rendering enables MVTN to work on 3D CAD models and the more accessible 3D point cloud data.

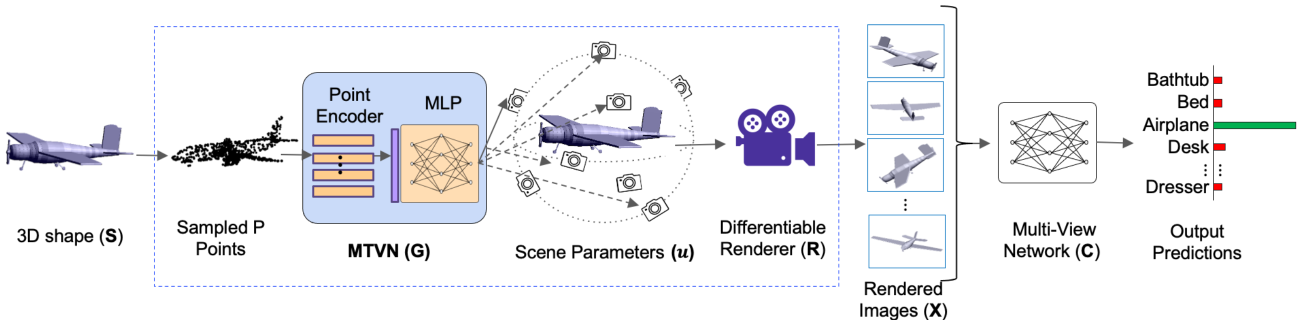

Figure 2. **End-to-End Learning Pipeline for Multi-View Recognition.** To learn adaptive scene parameters $\mathbf{u}$ that maximize the performance of a multi-view network $\mathbf{C}$ for every 3D object shape $\mathbf{S}$, we use a differentiable renderer $\mathbf{R}$. MVTN extracts coarse features from $\mathbf{S}$ by a point encoder and regresses the adaptive scene parameters for that object. In this example, the parameters $\mathbf{u}$ are the azimuth and elevation angles of cameras pointing towards the center of the object. The MVTN pipeline is optimized end-to-end for the task loss.

3. Methodology

We illustrate our proposed multi-view pipeline using MVTN in Fig. 2. MVTN is a generic module that learns camera view-point transformations for specific 3D multi-view tasks, *e.g*. 3D shape classification. In this section, we review a generic framework for common multi-view pipelines, introduce MVTN details, and present an integration of MVTN for 3D shape classification and retrieval.

3.1. Overview of Multi-View 3D Recognition

3D multi-view recognition defines M different images $\{\mathbf{x}_i\}_{i=1}^M$ rendered from multiple view-points of the same shape $\mathbf{S}$. The views are fed into the same backbone network $\mathbf{f}$ that extracts discriminative features per view. These features are then aggregated among views to describe the entire shape and used for downstream tasks such as classification or retrieval. Specifically, a multi-view network $\mathbf{C}$ with parameters $\boldsymbol{\theta}_\mathbf{C}$ operates on an input set of images $\mathbf{X} \in \mathbb{R}^{M\times h\times w\times c}$ to obtain a softmax probability vector for the shape $\mathbf{S}$.

Training Multi-View Networks. The simplest deep multi-view classifier is MVCNN, where $\mathbf{C} = \text{MLP}\left(\max_i \mathbf{f}(\mathbf{x}_i)\right)$ with $\mathbf{f} : \mathbb{R}^{h\times w\times c} \rightarrow \mathbb{R}^d$ being a 2D CNN backbone (*e.g*. ResNet [30]) applied individually on each rendered image. A more recent method like ViewGCN would be described as $\mathbf{C} = \text{MLP}\left(\text{cat}_{\text{GCN}}\left(\mathbf{f}(\mathbf{x}_i)\right)\right)$, where cat_{GCN} is an aggregation of views' features learned from a graph convolutional network. In general, learning a task-specific multi-view network on a labeled 3D dataset is formulated as:

$$\begin{aligned} &\arg\min_{\boldsymbol{\theta}_\mathbf{C}} \ \sum_n^N L\left(\mathbf{C}(\mathbf{X}_n)\ ,\ y_n\right) \\ =\ &\arg\min_{\boldsymbol{\theta}_\mathbf{C}} \sum_n^N L\left(\mathbf{C}\big(\mathbf{R}(\mathbf{S}_n, \mathbf{u}_0)\big)\ ,\ y_n\right), \end{aligned} \tag{1}$$

where L is a task-specific loss defined over N 3D shapes in the dataset, y_n is the label for the n^{th} 3D shape $\mathbf{S}_n$, and $\mathbf{u}_0 \in \mathbb{R}^\tau$ is a set of τ fixed scene parameters for the entire dataset. These parameters represent properties that affect the rendered image, including camera view-point, light, object color, and background. $\mathbf{R}$ is the renderer that takes as input a shape $\mathbf{S}_n$ and the parameters $\mathbf{u}_0$ to produce M multi-view images $\mathbf{X}_n$ per shape. In our experiments, we choose the scene parameters $\mathbf{u}$ to be the azimuth and elevation angles of the camera view-points pointing towards the object center, thus setting $\tau = 2M$.

Canonical Views. Previous multi-view methods rely on scene parameters $\mathbf{u}_0$ that are pre-defined for the entire 3D dataset. In particular, the fixed camera view-points are usually selected based on the alignment of the 3D models in the dataset. The most common view configurations are *circular* that aligns view-points on a circle around the object [61, 76] and *spherical* that aligns equally spaced view-points on a sphere surrounding the object [67, 38]. Fixing those canonical views for all 3D objects can be misleading for some classes. For example, looking at a bed from the bottom could confuse a 3D classifier. In contrast, MVTN learns to regress per-shape view-points, as illustrated in Fig. 3.

3.2. Multi-View Transformation Network (MVTN)

Previous multi-view methods take the multi-view image $\mathbf{X}$ as the only representation for the 3D shape, where $\mathbf{X}$ is rendered using fixed scene parameters $\mathbf{u}_0$. In contrast, we consider a more general case, where $\mathbf{u}$ is *variable* yet within bounds $\pm\mathbf{u}_{\text{bound}}$. Here, $\mathbf{u}_{\text{bound}}$ is positive and it defines the permissible range for the scene parameters. We set $\mathbf{u}_{\text{bound}}$ to $180°$ and $90°$ for each azimuth and elevation angle.

Differentiable Renderer. A renderer $\mathbf{R}$ takes a 3D shape $\mathbf{S}$ (mesh or point cloud) and scene parameters $\mathbf{u}$ as inputs, and outputs the corresponding M rendered images $\{\mathbf{x}_i\}_{i=1}^M$. Since $\mathbf{R}$ is differentiable, gradients $\frac{\partial \mathbf{x}_i}{\partial \mathbf{u}}$ can propagate backward from each rendered image to the scene parameters, thus establishing a framework that suits end-to-end deep learning pipelines. When $\mathbf{S}$ is represented as a 3D mesh, $\mathbf{R}$ has two components: a *rasterizer* and a *shader*. First, the rasterizer transforms meshes from the world to view coordinates given

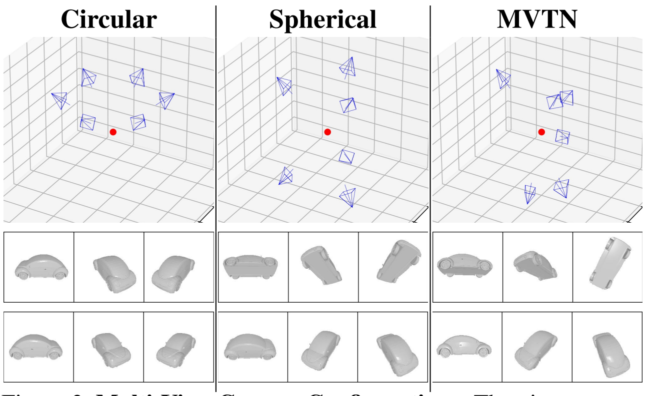

Figure 3. **Multi-View Camera Configurations**: The view setups commonly used in the multi-view literature are circular [61] or spherical [67, 38]. Our MVTN learns to predict specific view-points for each object shape at inference time. The shape's center is shown as a red dot, and the view-points as blue cameras with their mesh renderings shown at the bottom.

the camera view-point and assigns faces to pixels. Using these face assignments, the shader creates multiple values for each pixel then blends them. On the other hand, if $\mathbf{S}$ is represented by a 3D point cloud, $\mathbf{R}$ would use an alpha-blending mechanism instead [68]. Fig. 3 and Fig. 4 illustrate examples of mesh and point cloud renderings used in MVTN.

View-Points Conditioned on 3D Shape. We design $\mathbf{u}$ to be a function of the 3D shape by learning a Multi-View Transformation Network (MVTN), denoted as $\mathbf{G} \in \mathbb{R}^{P \times 3} \rightarrow \mathbb{R}^{\tau}$ and parameterized by $\boldsymbol{\theta}_{\mathbf{G}}$, where P is the number of points sampled from shape $\mathbf{S}$. Unlike Eq (1) that relies on constant rendering parameters, MVTN predicts $\mathbf{u}$ adaptively for each object shape $\mathbf{S}$ and is optimized along with the classifier $\mathbf{C}$. The pipeline is trained end-to-end to minimize the following loss on a dataset of N objects:

$$\begin{aligned} \underset{\boldsymbol{\theta}_{\mathbf{C}}, \boldsymbol{\theta}_{\mathbf{G}}}{\arg\min} \sum_{n}^{N} L\left(\mathbf{C}\big(\mathbf{R}(\mathbf{S}_n, \mathbf{u}_n)\big)\ ,\ y_n\right), \\ \text{s. t.} \quad \mathbf{u}_n = \mathbf{u}_{\text{bound}}.\tanh\big(\mathbf{G}(\mathbf{S}_n)\big) \end{aligned} \tag{2}$$

Here, $\mathbf{G}$ encodes a 3D shape to predict its optimal view-points for the task-specific multi-view network $\mathbf{C}$. Since the goal of $\mathbf{G}$ is only to predict view-points and not classify objects (as opposed to $\mathbf{C}$), its architecture is designed to be simple and light-weight. As such, we use a simple point encoder (*e.g.* shared MLP as in PointNet [55]) that processes P points from $\mathbf{S}$ and produces coarse shape features of dimension b. Then, a shallow MLP regresses the scene parameters $\mathbf{u}_n$ from the global shape features. To force the predicted parameters $\mathbf{u}$ to be within a permissible range $\pm\mathbf{u}_{\text{bound}}$, we use a hyperbolic tangent function scaled by $\mathbf{u}_{\text{bound}}$.

MVTN for 3D Shape Classification. To train MVTN for 3D shape classification, we define a cross-entropy loss in Eq (2), yet other losses and regularizers can be used here as well. The multi-view network ($\mathbf{C}$) and the MVTN ($\mathbf{G}$) are trained jointly on the same loss. One merit of our multi-view pipeline is its ability to seamlessly handle 3D point clouds, which is absent in previous multi-view methods. When $\mathbf{S}$ is a 3D point cloud, we simply define $\mathbf{R}$ as a differentiable point cloud renderer.

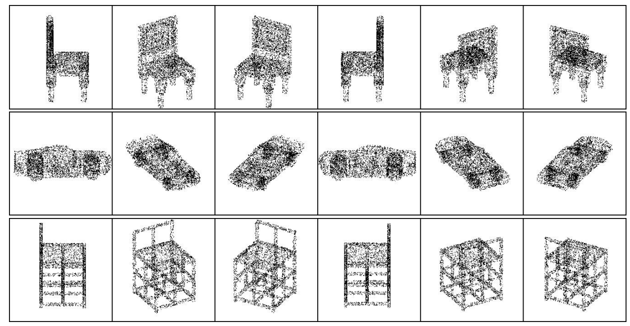

Figure 4. **Multi-View Point Cloud Renderings.** We show some examples of point cloud renderings used in our pipeline. Note how point cloud renderings offer more information about content hidden from the camera view-point (*e.g.* car wheels from the occluded side), which can be useful for recognition.

MVTN for 3D Shape Retrieval. The shape retrieval task is defined as follows: given a query shape $\mathbf{S}_q$, find the most similar shapes in a broader set of size N. For this task, we follow the retrieval setup of MVCNN [61]. In particular, we consider the deep feature representation of the last layer before the classifier in $\mathbf{C}$. We project those features into a more expressive space using LFDA reduction [62] and consider the reduced feature as the signature to describe a shape. At test time, shape signatures are used to retrieve (in order) the most similar shapes in the training set.

4. Experiments

We evaluate MVTN for the tasks of 3D shape classification and retrieval on ModelNet40 [71], ShapeNet Core55 [7], and the more realistic ScanObjectNN [64].

4.1. Datasets

ModelNet40. ModelNet40 [71] is composed of 12,311 3D objects (9,843/2,468 in training/testing) labelled with 40 object classes. Since we render 3D models in the forward pass, we limit the number of triangles in the meshes due to hardware constraints. In particular, we simplify the meshes to 20k vertices using the official Blender API [4, 22].

ShapeNet Core55. ShapeNet Core55 is a subset of ShapeNet [7] comprising 51,162 3D mesh objects labelled with 55 object classes. The training, validation, and test sets consist of 35764, 5133, and 10265 shapes, respectively. It is designed for the shape retrieval challenge SHREK [60].

ScanObjectNN. ScanObjectNN [64] is a recently released point cloud dataset for 3D classification that is more realistic and challenging than ModelNet40, since it includes background and considers occlusions. The dataset is composed of 2902 point clouds divided into 15 object categories. We consider its three main variants: object only, object with background, and the hardest perturbed variant (PB_T50_RS variant). These variants are used in the 3D Scene Understanding Benchmark associated with the ScanObjectNN dataset. This dataset offers a more challenging setup than ModelNet40 and tests the generalization capability of 3D deep learning model in more realistic scenarios.

4.2. Metrics

Classification Accuracy. The standard evaluation metric in 3D classification is accuracy. We report overall accuracy (percentage of correctly classified test samples) and average per-class accuracy (mean of all true class accuracies).

Retrieval mAP. Shape retrieval is evaluated by mean Average Precision (mAP) over test queries. For every query shape $\mathbf{S}_q$ from the test set, AP is defined as $AP = \frac{1}{\text{GTP}} \sum_{n}^{N} \frac{\mathbb{1}(\mathbf{S}_n)}{n}$, where GTP is the number of ground truth positives, N is the size of the ordered training set, and $\mathbb{1}(\mathbf{S}_n) = 1$ if the shape $\mathbf{S}_n$ is from the same class label of query $\mathbf{S}_q$. We average the retrieval AP over the test set to measure retrieval mAP.

4.3. Baselines

Voxel Networks. We choose VoxNet [52], DLAN [20], and 3DShapeNets [71] as baselines that use voxels.

Point Cloud Networks. We select PointNet [55], PointNet++ [57], DGCNN [66], PVNet [75], and KPConv [63] as baselines that use point clouds. These methods leverage different convolution operators on point clouds by aggregating local and global point information.

Multi-view Networks. We compare against MVCNN [61], RotationNet [38], GVCNN [19] and ViewGCN [67] as representative multi-view methods. These methods are limited to meshes, pre-rendered from canonical view-points.

4.4. MVTN Details

Rendering. We choose the differentiable mesh and point cloud renderers $\mathbf{R}$ from Pytorch3D [58] in our pipeline for their speed and compatibility with Pytorch libraries [54]. We show examples of the rendered images for meshes (Fig. 3) and point clouds (Fig. 4). Each rendered image has a size of 224×224. For ModelNet40, we use the differentiable *mesh* renderer. We direct the light randomly and assign a random color for the object for augmentation purposes in training. In testing, we keep a fixed light pointing towards the object center and color the object white for stable performance. For ShapeNet Core55 and ScanObjectNN, we use the differentiable *point cloud* renderer using 2048 and 5000 points, respectively. Point cloud rendering offers a light alternative to mesh rendering when the mesh contains a large number of faces that hinders training the MVTN pipeline.

Method	Data Type	Classification Accuracy (**Per-Class**)	(**Overall**)
VoxNet [52]	Voxels	83.0	85.9
PointNet [55]	Points	86.2	89.2
PointNet++ [57]	Points	-	91.9
PointCNN [46]	Points	88.1	91.8
DGCNN [66]	Points	90.2	92.2
SGAS [44]	Points	-	93.2
KPConv[63]	Points	-	92.9
PTransformer[78]	Points	**90.6**	**93.7**
MVCNN [61]	12 Views	90.1	90.1
GVCNN [19]	12 Views	90.7	93.1
ViewGCN [67]	20 Views	**96.5**	**97.6**
ViewGCN [67]*	12 views	90.7	93.0
ViewGCN [67]*	20 views	91.3	93.3
MVTN (ours)*	12 Views	92.0	**93.8**
MVTN (ours)*	20 Views	**92.2**	93.5

Table 1. **3D Shape Classification on ModelNet40**. We compare MVTN against other methods in 3D classification on ModelNet40 [71]. * 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.

Method	Classification Overall Accuracy **OBJ_BG**	**OBJ_ONLY**	**Hardest**
3DMFV [3]	68.2	73.8	63.0
PointNet [55]	73.3	79.2	68.0
SpiderCNN [73]	77.1	79.5	73.7
PointNet ++ [57]	82.3	84.3	77.9
PointCNN [46]	86.1	85.5	78.5
DGCNN [66]	82.8	86.2	78.1
SimpleView [23]	-	-	79.5
BGA-DGCNN [64]	-	-	79.7
BGA-PN++ [64]	-	-	80.2
MVTN (ours)	**92.6**	**92.3**	**82.8**

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 [64]. The variants include object with background, object only, and the hardest variant.

View-Point Prediction. As shown in Eq (2), the MVTN $\mathbf{G}$ network learns to predict the view-points directly (*MVTN-direct*). Alternatively, MVTN can learn relative offsets w.r.t. initial parameters $\mathbf{u}_0$. In this case, we concatenate the point features extracted in $\mathbf{G}$ with $\mathbf{u}_0$ to predict the offsets to apply on $\mathbf{u}_0$. The learned view-points $\mathbf{u}_n$ in Eq (2) are defined as: $\mathbf{u}_n = \mathbf{u}_0 + \mathbf{u}_{\text{bound}}.\tanh\big(\mathbf{G}(\mathbf{u}_0\ ,\ \mathbf{S}_n)\big)$. We take $\mathbf{u}_0$ to be the circular or spherical configurations commonly used in multi-view classification pipelines [61, 38, 67]. We refer to these learnable variants as *MVTN-circular* and *MVTN-spherical*,

Method	Data Type	Shape Retrieval (mAP) ModelNet40	ShapeNet Core
LFD [10]	Voxels	40.9	-
3D ShapeNets [71]	Voxels	49.2	-
Densepoint[48]	Points	88.5	-
PVNet[75]	Points	89.5	-
MVCNN [61]	12 Views	80.2	73.5
GIFT [2]	20 Views	-	64.0
MVFusionNet [34]	12 Views	-	62.2
ReVGG [60]	20 Views	-	74.9
RotNet [38]	20 Views	-	77.2
ViewGCN [67]	20 Views	-	78.4
MLVCNN [35]	24 Views	92.2	-
MVTN (ours)	12 Views	**92.9**	**82.9**

Table 3. **3D Shape Retrieval**. We benchmark the shape retrieval mAP of MVTN on ModelNet40 [71] and ShapeNet Core55 [7, 60]. MVTN achieves the best retrieval performance among recent state-of-the-art methods on both datasets with only 12 views.

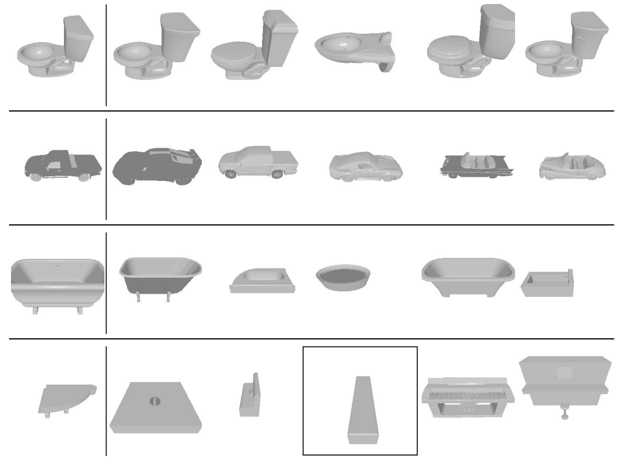

Figure 5. **Qualitative Examples for Object Retrieval**: *(left):* we show some query objects from the test set. *(right)*: we show top five retrieved objects by our MVTN from the training set. Images of negative retrieved objects are framed.

accordingly. For MVTN-circular, the initial elevations for the views are 30°, and the azimuth angles are equally distributed over 360° following [61]. For MVTN-spherical, we follow the method from [16] that places equally-spaced view-points on a sphere for an arbitrary number of views, which is similar to the "dodecahedral" configuration in ViewGCN.
Architecture. We select MVCNN [61], RotationNet [38], and the more recent ViewGCN [67] as our multi-view networks of choice in the MVTN pipeline. In our experiments, we select PointNet [55] as the 3D point encoder network **G** and experiment with DGCNN in Section 6.1. We sample $P = 2048$ points from each mesh as input to the point encoder and use a 5-layer MLP for the regression network, which takes as input the point features extracted by the point encoder of size $b = 40$. All MVTN variants and the baseline multi-view networks use ResNet-18 [30] pre-trained on ImageNet [59] for the multi-view backbone in C, with output features of size $d = 1024$. The main classification and retrieval results are based on MVTN-spherical with ViewGCN [67] as the multi-view network **C**, unless otherwise specified as in Section 5.3 and Section 6.1.
Training Setup. To avoid gradient instability introduced by the renderer, we use gradient clipping in the MVTN network **G**. We clip the gradient updates such that the ℓ_2 norm of the gradients does not exceed 30. We use a learning rate of 0.001 but refrain from fine-tuning the hyper-parameters introduced in MVCNN [61] and View-GCN [67]. More details about the training procedure are in the **supplementary material**.

5. Results

The main results of MVTN are summarized in Tables 1, 2, 3 and 4. We achieve state-of-the-art performance in 3D classification on ScanObjectNN by a large margin (up to 6%) and achieve a competitive test accuracy of **93.8**% on ModelNet40. On shape retrieval, we achieve state-of-the-art performance on both ShapeNet Core55 (**82.9** mAP) and ModelNet40 (**92.9** mAP). Following the common practice, we report the best results out of four runs in benchmark tables, but detailed results are in **supplementary material**.

5.1. 3D Shape Classification

Table 1 compares the performance of MVTN against other methods on ModelNet40 [71]. Our MVTN achieves a competitive test accuracy of 93.8% compared to all previous methods. ViewGCN [67] achieves higher classification performance by relying on higher quality images from a more advanced yet non-differentiable OpenGL [69] renderer. For a fair comparison, we report with an * the performance of ViewGCN using images generated by the renderer used in MVTN. Using the same rendering process, regressing views with MVTN improves the classification performance of the baseline ViewGCN at 12 and 20 views. We believe future advances in differentiable rendering would bridge the gap between our rendered images and the original high-quality pre-rendered ones.

Table 2 reports the classification accuracy of a 12 view MVTN on the realistic ScanObjectNN benchmark [64]. MVTN improves performance on different variants of the dataset. The most difficult variant of ScanObjectNN (PB_T50_RS) includes challenging scenarios of objects undergoing translation and rotation. Our MVTN achieves state-of-the-art results (+2.6%) on this variant, highlighting the merits of MVTN for realistic 3D point cloud scans. Also, note how adding background points (in OBJ_BG) does not hurt MVTN, contrary to most other classifiers. .

Method	Rotation Perturbations Range $0°$	$\pm 90°$	$\pm 180°$
PointNet [55]	88.7	42.5	38.6
PointNet ++ [57]	88.2	47.9	39.7
RSCNN [49]	90.3	90.3	90.3
MVTN (ours)	**91.7**	**90.8**	**91.2**

Table 4. **Rotation Robustness on ModelNet40.** At test time, we randomly rotate objects in ModelNet40 around the Y-axis (gravity) with different ranges and report the overall accuracy. MVTN displays strong robustness to such Y-rotations.

5.2. 3D Shape Retrieval

Table 3 reports the retrieval mAP of MVTN compared with recent methods on ModelNet40 [71] and ShapeNet Core55 [7]. The results of the latter methods are taken from [35, 67, 75]. MVTN achieves state-of-the-art retrieval performance (92.9% mAP) on ModelNet40. It also improves the state-of-the-art by a large margin in ShapeNet, while only using 12 views. It is important to note that the baselines in Table 3 include strong and recent methods trained specifically for retrieval, such as MLVCNN [35]. Fig. 5 shows qualitative examples of objects retrieved using MVTN.

5.3. Rotation Robustness

A common practice in 3D shape classification literature is to test the robustness of trained models to perturbations at test time. Following the same setup as [49, 27], we perturb the shapes with random rotations around the Y-axis (gravity-axis) contained within $\pm 90°$ and $\pm 180°$. We repeat the inference ten times for each setup and report the average performance in Table 4. The MVTN-circular variant (with MVCNN) reaches state-of-the-art performance in rotation robustness (91.2% test accuracy) compared to more advanced methods trained in the same setup. The baseline RSCNN [49] is a strong baseline designed to be invariant to translation and rotation. In contrast, MVTN is learned in a simple setup with MVCNN without targeting rotation invariance.

5.4. Occlusion Robustness

To test the usefulness of MVTN in a realistic scenario, we investigate the common problem of occlusion in 3D computer vision, especially in 3D point cloud scans. Various factors lead to occlusion, including the view angle to the object, the sensor's sampling density (*e.g.* LiDAR), or the presence of noise in the sensor. In such realistic scenarios, deep learning models typically fail. To quantify this 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 [55], DGCNN [66], and MVTN on the ModelNet40 point cloud dataset. Then, at test time, we crop a portion of the object (from 0% occlusion ratio to 100%) along the $\pm$X, $\pm$Y, and $\pm$Z directions. Fig. 6 shows examples of this occlusion effect with different occlusion ratios. In all robustness experiments, the studied transformations (rotation or occlusion) happen only in test time. All the methods compared, including MVTN, are trained naturally without any augmentation by those transformations. We report the average test accuracy of the six cropping directions for the baselines and MVTN in Table 5. Note how MVTN achieves high test accuracy even when large portions of the object are cropped. Interestingly, MVTN outperforms PointNet [55] 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 [55, 28].

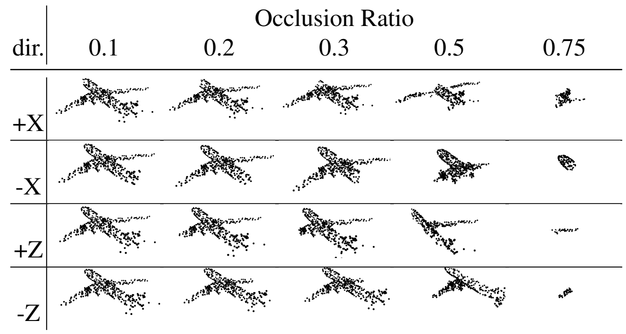

Figure 6. **Occlusion of 3D Objects**: We simulate realistic occlusion scenarios in 3D point clouds by cropping a percentage of the object along canonical directions. Here, we show an object occluded with different ratios and from different directions.

Method	Occlusion Ratio 0	0.1	0.2	0.3	0.5	0.75
PointNet [55]	89.1	88.2	86.1	81.6	53.5	4.7
DGCNN [66]	92.1	77.1	74.5	71.2	30.1	4.3
MVTN (ours)	**92.3**	**90.3**	**89.9**	**88.3**	**67.1**	**9.5**

Table 5. **Occlusion Robustness of 3D Methods.** We report the test accuracy on point cloud ModelNet40 for different occlusion ratios of the data to measure occlusion robustness of different 3D methods. MVTN achieves 13% better accuracy than PointNet (a robust network) when half of the object is occluded.

6. Analysis and Insights

6.1. Ablation Study

This section performs a comprehensive ablation study on the different components of MVTN and their effect on the overall test accuracy on ModelNet40 [71].

Number of Views. We study the effect of the number of views M on the performance of MVCNN when using fixed views (circular/spherical), learned views (MVTN), and random views. The experiments are repeated four times, and the

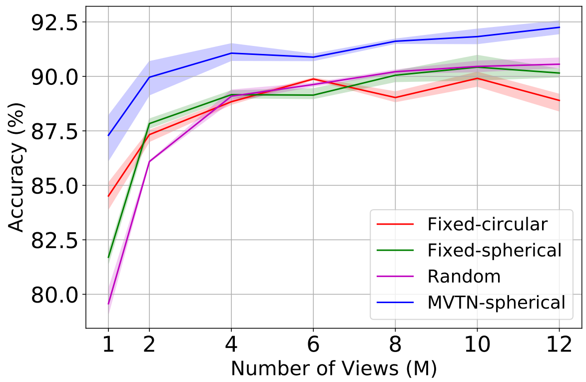

Figure 7. **Effect of the Number of Views.** We plot the test accuracy *vs*. the number of views (M) used to train MVCNN on fixed, random, and learned MVTN view configurations. We observe a consistent 2% improvement with MVTN over a variety of views.

Backbone Network	Point Encoder	MVTN Setup	Results Accuracy
ResNet-18	PointNet	circular	92.83 ± 0.06
		spherical	93.41 ± 0.13
	DGCNN	circular	93.03 ± 0.15
		spherical	93.26 ± 0.04
ResNet-34	PointNet	circular	92.72 ± 0.16
		spherical	92.83 ± 0.12
	DGCNN	circular	92.72 ± 0.03
		spherical	92.63 ± 0.15

Table 6. **Ablation Study**. We analyze the effect of ablating different MVTN components on test accuracy in ModelNet40. Namely, we observe that using deeper backbone CNNs or a more complex point encoder do not increase the test accuracy.

average test accuracies with confidence intervals are shown in Fig. 7. The plots show how learned MVTN-spherical achieves consistently superior performance across a different number of views.

Choice of Backbone and Point Encoders. In all of our main MVTN experiments, we use ResNet-18 as our backbone and PointNet as the point feature extractor. However, different choices could be made for both. We explore using DGCNN [66] as an alternative point encoder and ResNet-34 as an alternative 2D backbone in ViewGCN. We report all MVTN ablation results in Table 6. We observe diminishing returns for making the CNN backbone and the shape feature extractor more complex in the MVTN setup, which justifies using the simpler combination in our main experiments

Choice of Multi-View Network. MVTN integrates smoothly with different multi-view networks and always leads to performance boost. In Table 7. we show the overall accuracies (averaged over four runs) on ModelNet40 of 12 views when fixed views are used versus when MVTN is used with different multi-view networks.

View Selection	Multi-View Networks		
	MVCNN[61]	RotNet[38]	ViewGCN[67]
fixed views	90.4	91.6	93.0
with MVTN	**92.6**	**93.2**	**93.8**

Table 7. **Integrating MVTN with Multi-View Networks**. We show overall classification accuracies on ModelNet40 with 12 views on different multi-view networks when fixed views are used versus when MVTN is used.

Network	GFLOPs	Time (ms)	Parameters # (M)
MVCNN [61]	43.72	39.89	11.20
ViewGCN [67]	44.19	26.06	23.56
MVTN module	**1.78**	**4.24**	**3.5**

Table 8. **Time and Memory Requirements**. We assess the contribution of the MVTN module to the time and memory requirements in the multi-view pipeline. We note that the MVTN's time and memory requirements are negligible.

Other Factors Affecting MVTN. We study the effect of the light direction in the renderer, the camera's distance to the object, and the object's color. We also study the transferability of the learned views from one multi-view network to another, and the performance of MVTN variants. More details are provided in the **supplementary material**.

6.2. Time and Memory Requirements

We compare the time and memory requirements of different parts of our 3D recognition pipeline. We record the number of floating-point operations (GFLOPs) and the time of a forward pass for a single input sample. In Table 8, MVTN contributes negligibly to the time and memory requirements of the multi-view networks.

7. Conclusions and Future Work

Current multi-view methods rely on fixed views aligned with the dataset. We propose MVTN that learns to regress view-points for any multi-view network in a fully differentiable pipeline. MVTN harnesses recent developments in differentiable rendering and does not require any extra training supervision. Empirical results highlight the benefits of MVTN in 3D classification and 3D shape retrieval. Some possible future works for MVTN include extending it to other 3D vision tasks such as shape and scene segmentation. Furthermore, MVTN can include more intricate scene parameters different from the camera view-points, such as light and textures.

Acknowledgments. This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding.

References

[1] Ceyhun Burak Akgül, Bülent Sankur, Yücel Yemez, and Francis Schmitt. 3d model retrieval using probability density-based shape descriptors. *IEEE transactions on pattern analysis and machine intelligence*, 31(6):1117–1133, 2009. 2

[2] 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. 6

[3] 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. 5

[4] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2018. 4

[5] Gary Bradski and Stephen Grossberg. Recognition of 3-d objects from multiple 2-d views by a self-organizing neural architecture. In *From Statistics to Neural Networks*, pages 349–375. Springer, 1994. 2

[6] Alexander M Bronstein, Michael M Bronstein, Leonidas J Guibas, and Maks Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG)*, 30(1):1–20, 2011. 2

[7] 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. 2, 4, 6, 7

[8] Siddhartha Chaudhuri and Vladlen Koltun. Data-driven suggestions for creativity support in 3d modeling. In *ACM SIGGRAPH Asia 2010 papers*, pages 1–10. 2010. 2

[9] 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. 2

[10] 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. 6

[11] Songle Chen, Lintao Zheng, Yan Zhang, Zhixin Sun, and Kai Xu. Veram: View-enhanced recurrent attention model for 3d shape classification. *IEEE transactions on visualization and computer graphics*, 25(12):3244–3257, 2018. 2

[12] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems*, pages 9609–9619, 2019. 2

[13] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 1, 2

[14] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999, 2016. 2

[15] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 452–468, 2018. 1

[16] Markus Deserno. How to generate equidistributed points on the surface of a sphere. *If Polymerforshung (Ed.)*, page 99, 2004. 6

[17] Carlos Esteves, Yinshuang Xu, Christine Allen-Blanchette, and Kostas Daniilidis. Equivariant multi-view networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1568–1577, 2019. 2

[18] Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8279–8286, 2019. 1

[19] 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. 2, 5

[20] Takahiko Furuya and Ryutarou Ohbuchi. Deep aggregation of local 3d geometric features for 3d model retrieval. In *BMVC*, volume 7, page 8, 2016. 5

[21] Yue Gao, Jinhui Tang, Richang Hong, Shuicheng Yan, Qionghai Dai, Naiyao Zhang, and Tat-Seng Chua. Camera constraint-free view-based 3-d object retrieval. *IEEE Transactions on Image Processing*, 21(4):2269–2281, 2011. 2

[22] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997. 4

[23] 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. 5

[24] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018. 1, 2

[25] Ulf Grenander. *Pattern analysis: lectures in pattern theory, volume I*. Springer, 1978. 2

[26] Abdullah Hamdi and Bernard Ghanem. Towards analyzing semantic robustness of deep neural networks. In *European Conference on Computer Vision*, pages 22–38. Springer, 2020. 2

[27] Abdullah Hamdi, Matthias Muller, and Bernard Ghanem. SADA: semantic adversarial diagnostic attacks for autonomous applications. In *AAAI Conference on Artificial Intelligence*, 2020. 7

[28] 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. 7

[29] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 1

[30] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3, 6

[31] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. Triplet-center loss for multi-view 3d object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1945–1954, 2018. 2

[32] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 2

[33] Maximilian Jaritz, Jiayuan Gu, and Hao Su. Multi-view pointnet for 3d scene understanding. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[34] 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. 6

[35] 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. 2, 6, 7

[36] Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4996–5004. Curran Associates, Inc., 2016. 2

[37] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3779–3788, 2017. 1

[38] 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. 1, 2, 3, 4, 5, 6, 8

[39] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3907–3916, 2018. 2

[40] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in neural information processing systems (NIPS)*, pages 2539–2547, 2015. 2

[41] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual multi-view fusion for 3d semantic segmentation. *arXiv preprint arXiv:2007.13138*, 2020. 1

[42] Loic Landrieu and Mohamed Boussaha. Point cloud oversegmentation with graph-structured deep metric learning. pages 7440–7449, 2019. 2

[43] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4558–4567, 2018. 2

[44] Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1620–1630, 2020. 5

[45] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. In *SIGGRAPH Asia 2018 Technical Papers*, page 222. ACM, 2018. 2

[46] 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. 2, 5

[47] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7708–7717, 2019. 2

[48] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5239–5248, 2019. 6

[49] 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. 7

[50] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, pages 965–975, 2019. 2

[51] Matthew M Loper and Michael J Black. Opendr: An approximate differentiable renderer. In *European Conference on Computer Vision (ECCV)*, pages 154–169. Springer, 2014. 2

[52] 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. 1, 2, 5

[53] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002. 2

[54] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 5

[55] 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, 2, 4, 5, 6, 7

[56] 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. 2

[57] 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. 1, 2, 5, 7

[58] 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. 2, 5

[59] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. 1, 6

[60] 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. 2, 4, 6

[61] 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, 2, 3, 4, 5, 6, 8

[62] Masashi Sugiyama. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *Journal of machine learning research*, 8(May):1027–1061, 2007. 4

[63] 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. 5

[64] 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. 2, 4, 5, 6

[65] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2569–2578, 2018. 2

[66] 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. 1, 2, 5, 7, 8

[67] 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, 2, 3, 4, 5, 6, 7, 8

[68] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020. 2, 4

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

[70] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural Scene De-rendering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[71] 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. 2, 4, 5, 6, 7

[72] 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*, pages 1912–1920, 2015. 2

[73] 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. 5

[74] Ze Yang and Liwei Wang. Learning relationships for multi-view 3d object recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7505–7514, 2019. 2

[75] 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. 2, 5, 6, 7

[76] Tan Yu, Jingjing Meng, and Junsong Yuan. Multi-view harmonized bilinear network for 3d object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 186–194, 2018. 2, 3

[77] Ye Yu and William AP Smith. Inverserendernet: Learning single image inverse rendering. *arXiv preprint arXiv:1811.12328*, 2018. 2

[78] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020. 5