A Deeper Look at 3D Shape Classifiers

Jong-Chyi Su, Matheus Gadelha, Rui Wang, Subhransu Maji

University of Massachusetts, Amherst
{jcsu,mgadelha,ruiwang,smaji}@cs.umass.edu

Abstract. We investigate the role of representations and architectures for classifying 3D shapes in terms of their computational efficiency, generalization, and robustness to adversarial transformations. By varying the number of training examples and employing cross-modal transfer learning we study the role of initialization of existing deep architectures for 3D shape classification. Our analysis shows that multiview methods continue to offer the best generalization even without pretraining on large labeled image datasets, and even when trained on simplified inputs such as binary silhouettes. Furthermore, the performance of voxel-based 3D convolutional networks and point-based architectures can be improved via cross-modal transfer from image representations. Finally, we analyze the robustness of 3D shape classifiers to adversarial transformations and present a novel approach for generating adversarial perturbations of a 3D shape for multiview classifiers using a differentiable renderer. We find that point-based networks are more robust to point position perturbations while voxel-based and multiview networks are easily fooled with the addition of imperceptible noise to the input.

1 Introduction

Techniques for analyzing 3D shapes are becoming increasingly important due to the vast number of sensors that are capturing 3D data, as well as numerous computer graphics applications. In recent years a variety of deep architectures have been approached for classifying 3D shapes. These range from multiview approaches that render a shape from a set of views and deploy image-based classifiers, to voxel-based approaches that analyze shapes represented as a 3D occupancy grid, to point-based approaches that classify shapes represented as collection of points. However, there is relatively little work that studies the tradeoffs offered by these modalities and their associated techniques.

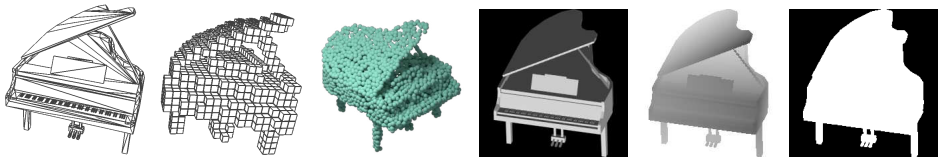
This paper aims to study three of these tradeoffs, namely the ability to generalize from a few examples, computational efficiency, and robustness to adversarial transformations. We pick a representative technique for each modality. For multiview representation we choose the Multiview CNN (MVCNN) architecture [31]; For voxel-based representation we choose the VoxNet [17,37] constructed using convolutions and pooling operations on a 3D grid; For point-based representation we choose the PointNet architecture [32]. The analysis is done on the widely-used ModelNet40 shape classification benchmark [37].

Some of our analysis leads to surprising results. For example, with deeper architectures and a modification in the rendering technique that renders with black background and better centers the object in the image the performance of a vanilla MVCNN can be improved to **95.0%** per-instance accuracy on the benchmark, outperforming several recent approaches. Another example is that while it is widely believed that the strong performance of MVCNN is due to the use of networks pretrained on large image datasets (e.g., ImageNet [26]), we find that even without such pretraining the MVCNN obtains **91.3%** accuracy, outperforming several voxel-based and point-based counterparts that also do not rely on such pretraining. Furthermore, the performance of MVCNN remains at **93.6%** even when trained with binary silhouettes (instead of shaded images) of shapes, suggesting that shading offer relatively little extra information on this benchmark for MVCNN.

We then systematically analyze the generalization ability of the models. First we analyze the accuracy of various models by varying the number of training examples per category. We find that the multiview approaches generalize faster obtaining near optimal performance with far fewer examples compared to the other approaches. We then analyze the role of initialization of these networks. As 3D shape datasets are currently lacking in comparison to large image datasets, we employ cross-modal distillation techniques [10,7] to guide learning. In particular we use representations extracted from pretrained MVCNNs to guide learning of voxel-based and point-based networks. Cross-modal distillation improves the performance of VoxNet and PointNet, especially when training data is limited.

Finally we analyze the robustness of these classifiers to adversarial perturbations. While generating adversarial inputs to VoxNet and PointNet is straightforward, it is not the case for multiview methods due to the rendering step. To this end we design an end-to-end differentiable MVCNN that takes an input a voxel representation and generates a set of views using a differentiable renderer. We analyze the robustness of these networks by estimating the amount of perturbation needed to obtain a misclassification. We find that PointNet is more robust, while MVCNN and VoxNet are both easily fooled by adding a small amount of noise. This is similar to the observations in prior work of adversarial inputs for image-based networks [33,6,16]. Somewhat surprisingly ImageNet pretraining reduces the robustness of the MVCNNs to adversarial perturbations.

In summary, we performed a detailed analysis of several recently proposed approaches for 3D shape classification. This resulted in a new state-of-the-art of **95.0%** on the ModelNet40 benchmark. The technical contributions include the use of cross-modal distillation for improving networks that operate on voxel-based and point-based representations and a novel approach for generating adversarial inputs for 3D shape classification for multiview approaches using a differentiable renderer. This allows us to directly compare and generate adversarial inputs for voxel-based and view-based methods using gradient-based techniques. The conclusion is that while PointNet architecture is less accurate, the use of orderless aggregation mechanism likely makes it more robust to adversarial perturbations compared to VoxNet and MVCNN, both of which are easily fooled.



(a) Input (b) Voxel (c) Point cloud (d) Phong (e) Depth (f) Silhouette

Fig. 1. Different representations of the input shape. From the left (a) the shapes in the database are represented as triangle meshes, (b) the shape converted to a 30^3 voxel grid, (c) point cloud representation with 2048 points, and (d-f) the model rendered using Phong shading, and as depth and binary silhouette images respectively.

2 Method

This Section describes the protocol for evaluating the performance of various classifiers. We describe the dataset, performance metrics, and training setup in Section 2.1, followed by the details of the deep classifiers we consider in Section 2.2, and the approach for generating adversarial examples in Section 2.3.

2.1 3D Shape Classification

Classification benchmark. All our evaluation is done on the ModelNet40 shape classification benchmark [37] following the standard training and test splits provided in the dataset. There are 40 categories with 9483 training models and 2468 test models. The numbers of models are not equal across classes hence we report both the per-instance and per-class accuracy on the test set. While most of the literature report results by training on the entire training set, some earlier work, notably [31] reports results on training and evaluation on a subset consisting of 80 training and 20 test examples per category.

Input representations. The dataset presents each shape as a collection of triangles, hence it is important to describe the exact way in which these are converted to point clouds, voxels, and images for input to different network architectures. These inputs are visualized in Figure 1 and described below:

- **Voxel representation.** To get voxel representations we follow the dataset from [23] where models are discretized to a $30 \times 30 \times 30$ occupancy grid. The data is available from the author's page.
- **Point cloud.** For point cloud representation we use the data from the PointNet approach [32] where 2048 points are uniformly sampled for each model.
- **Image representation.** To generate multiple views of the model we use a setup similar to [31]. Since the models are assumed to be upright oriented a set of virtual cameras are placed at 12 radially symmetric locations, i.e. every 30 degrees, facing the object center and at an elevation of 30 degrees. Comparing to [31], we render the images with black background and set the field-of-view of the camera such that the object is bounded by image

canvas and rendered as an image of size 224×224. A similar scheme was used to generate views for semantic segmentation of shapes in the Shape PFCN approach [12]. This had a non-negligible impact on the performance of the downstream models as discussed in Section 3.1. Given the setup we considered three different ways to render the models described below:

1. Phong shading, where images are rendered with the Phong reflection model [21] using Blender software [2]. The light and material setup is similar to the approach in [31].
2. Depth rendering, where only the depth value is recorded.
3. Silhouette rendering, where images are rendered as binary images for pixels corresponding to foreground.

Data augmentation. Models in the dataset are upright oriented, but not consistently oriented along the axis, i.e., models could be rotated arbitrarily along the upright direction. Models that rely on voxel or point cloud input often benefit from rotation augmentation along the upright axis during training and testing. Similar to the multiview setting we consider models rotated by 30 degree increments as additional data during training, and optionally aggregating votes across these instances at test time.

2.2 Classification Architectures

We consider the following deep architectures for shape classification.

Multiview CNN (MVCNN) The MVCNN architecture [31] uses rendered images of the model from different views as input. Each image is fed into a CNN with shared weights. A max-pooling layer across different views is used to perform an orderless aggregation of the individual representations followed by several non-linear layers for classification. While the original paper [31] used the VGG-M network [30] we also report results using:

- The VGG-11 network, which is the model with configuration A from [30]. The view-pooling layer is added before the first `fc` layer.
- Variants of residual networks proposed in [8] such as ResNet18, ResNet34, and ResNet50. The view-pooling layer is added before the final `fc` layer.

Voxel network (VoxNet) The VoxNet was first proposed in several early works [17,37] that uses convolution and pooling layers defined on 3D voxel grids. The early VoxNet models [17] used two 3D convolutional layers and 2 fully-connected layers. In our initial experiments we found the capacity of this network is limited. We also experimented with the deeper VoxNet architecture proposed in [32] which has five blocks of (`conv3d-batchnorm-LeakyReLU`) and includes batch normalization [11]. All `conv3d` layers have kernel size 5, stride 1 and channel size 32. The `LeakyReLU` has slope 0.1. Two fully-connected layers (`fc-batchnorm-ReLU-fc`) are added on top to obtain class predictions.

VoxMVCNN We also consider a hybrid model that takes voxels as input and uses a MVCNN approach for classification using a differentiable renderer. To achieve this we make two simplifications. First, only six renderings are considered corresponding to viewing directions along six dimensions $(\pm x, \pm y, \pm z)$. Second, the rendering is approximated using the approach suggested in PrGAN [5] where line integrals are used to compute pixel shading color. For example the line integral of a volume occupancy grid V along the axis k is given by $P((i,j),V) = 1-\exp(-\sum_k V(i,j,k))$. The idea is that the higher the sum of occupancy values along the axis, the closer the integral is to 1. The generated views for two models are shown in Figure 2. The renderings generated this way approximate silhouette renderings as described earlier. The primary advantage of this rendering method is that it's differentiable, and hence we use this model to analyze the robustness of the MVCNN architecture to adversarial inputs (described in Section 2.3).

Point network (PointNet) We follow the same architecture as PointNet [32] that operates on point cloud representations of a model. The architecture applies a series of non-linear mappings individually to each input point and performs orderless aggregations using max-pooling operations. Thus the model is invariant to the order in which the points are presented and can directly operate on point clouds without additional preprocessing such as spatial partitioning, or graph construction. Additionally, some initial layers are used to perform spatial transformations (rotation, scaling, etc.) Despite its simplicity the model and its variants have been shown to be effective at shape classification and segmentation tasks [32,22].

Training details. All the MVCNN models are trained in two stages as suggested in [31]. The model is first trained as a single-image classification task where the view-pooling layer is removed, then trained to jointly classify all the views with view-pooling layer in the second stage. We use the Adam optimizer [14] with learning rate 5×10^{-5} and 1×10^{-5} for first and second stage respectively and each stage is trained with 30 epochs. The batch size is set to 64 and 96 (eight models with twelve views) for each stage and the weight decay parameter is set to 0.001. The VoxNet is trained with Adam optimizer with learning rate 1×10^{-3} for 150 epochs. The batch size is set to 64 and weight decay parameter is 0.001. The VoxMVCNN is trained using the same procedure as MVCNN. For PointNet [32] we use the publicly available implementation released by the authors.

2.3 Generating Adversarial Inputs

Adversarial examples to image-based deep neural networks have been thoroughly explored in the literature [33,6,16,19]. However, there is no prior work that addresses adversarial examples to deep neural networks based on 3D representations. Here we want to investigate if adversarial shapes can be obtained from different 3D shape recognition models, and perhaps more importantly which 3D representation is more robust to adversarial examples. We define an adversarial example as follows. Consider $\phi(\mathbf{s}, y)$ as the score that a classifier ϕ gives to an

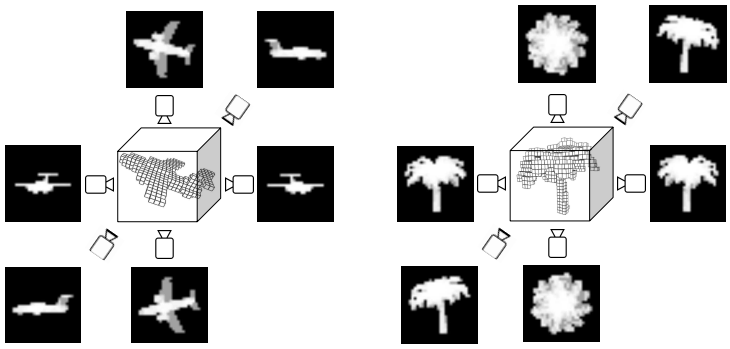


Fig. 2. Examples of rendered images for the VoxMVCNN architecture. The voxel input is rendered using the simplified technique described in Section 2.2 to generate 6 images which are processed using the MVCNN architecture [31] for classification.

input $\mathbf{s}$ belonging to a class y. An adversarial example $\mathbf{s}'$ is a sample that is perceptually similar to $\mathbf{s}$, but $\arg\max_y \phi(\mathbf{s}, y) \neq \arg\max_y \phi(\mathbf{s}', y)$. It is known from [33] that an effective way to compute adversarial examples to image-based models is the following. Given a sample from the dataset $\mathbf{s}$ and a class y' that one wishes to maximize the score, an adversarial example $\mathbf{s}'$ can be computed as follows

$$\mathbf{s}' = \mathbf{s} + \alpha \nabla_{\mathbf{s}} \phi(\mathbf{s}, y')$$

where $\nabla_{\mathbf{s}}$ is the gradient of the classifier with respect to the input $\mathbf{s}$, and α is the learning rate. For many image models, this single step procedure is able to generate perceptually indistinguishable adversarial examples. However, for some of the examples we experimented, a single step is not enough to generate a misclassification. Thus, we employ the following iterative procedure based on [16]:

$$\mathbf{s}'_{\mathbf{0}} = \mathbf{s}, \quad \mathbf{s}'_{\mathbf{t+1}} = clip_{\mathbf{s},\epsilon}\{\mathbf{s}_t + \alpha \nabla_{\mathbf{s}} \phi(\mathbf{s_t}, y')\} \tag{1}$$

where $clip_{\mathbf{s},\epsilon}\{x\}$ is an operator that clips the values of x to make sure the result will be in the L_∞ ϵ-neighborhood of $\mathbf{s}$. Notice that this procedure is agnostic to the representation used by the input data $\mathbf{s}$. Thus, we use the same method to generate adversarial examples for multiple modalities of 3D representation: voxels, point clouds, multi-view images. For voxel grids, we also clip the values of $\mathbf{s}$ to make sure their values are in $[0, 1]$. For multi-view representations, we need to make sure that all views are consistent with each other. We address this issue by using the VoxMVCNN architecture that generates multiple views from the same object through a differentiable renderer, i.e. line integral, as described in Section 2.2.

3 Experiments

We begin by investigating the model generalization in Section 3.1. Section 3.2 analyzes the effect of different architectures and renderings for the MVCNN.

Section 3.3 uses cross-modal distillation to improve the performance of VoxNet and PointNet. Section 3.4 compares the tradeoffs between different representations. Section 3.5 compares the robustness of different classifiers to adversarial perturbations. Finally, Section 3.6 puts the results presented in this paper in the context of prior work.

3.1 Learning From a Few Examples

One of the most desirable properties of a classifier is its ability to generalize from a few examples. We test this ability by evaluating the accuracy of different models as a function of training set size. We select the first M_k models in the training set for each class, where

$$M_k = \min(N_k, \{10, 20, 40, 80, 160, 320, 889\}),$$

and N_k is the number of models in class k. The maximum number of models per-class in the training set of ModelNet40 is 889. Figure 3 shows the per-class and per-instance accuracy for three different models as a function of the training set size. The MVCNN with the VGG-11 architecture has better generalization than VoxNet and PointNet across all training set sizes. MVCNN obtains 77.8% accuracy using only 10 training models per class, while PointNet and VoxNet obtain 62.5% and 57.9% respectively. The performance of MVCNN is near optimal with 160 models per class, far fewer than PointNet and VoxNet. When using the whole dataset for training, MVCNN (95.0%) outperforms PointNet (89.1%) and VoxNet (85.6%) by a large margin.

Several improvements have been proposed for both point-based and voxel-based architectures. The best performing point-based models to the best of our knowledge is the Kd-Networks [15] which achieves 91.8% per-instance accuracy. For voxel-based models, O-CNN [35] uses sparse convolutions to handle higher resolution with Octave trees [18] and achieves 90.6% per-instance accuracy. However, all of them are far below the MVCNN approach. More details and comparison to the state-of-the-art are in Section 3.6.

3.2 Dissecting the MVCNN Architecture

Given the high performance of MVCNN we investigate what factors contribute to its performance as described next.

Effect of model architecture. The MVCNN model in [31] used VGG-M architecture. However a number of different image networks have since been proposed. We used different CNN architectures for MVCNN and report the accuracies in Table 1. All models have similar performance suggesting that MVCNN is robust across different CNN architectures. In Table 3 we also compare with the results using VGG-M and AlexNet. With the same shaded images and training subset, VGG-11 achieves 89.1% and VGG-M has 89.9% accuracy.

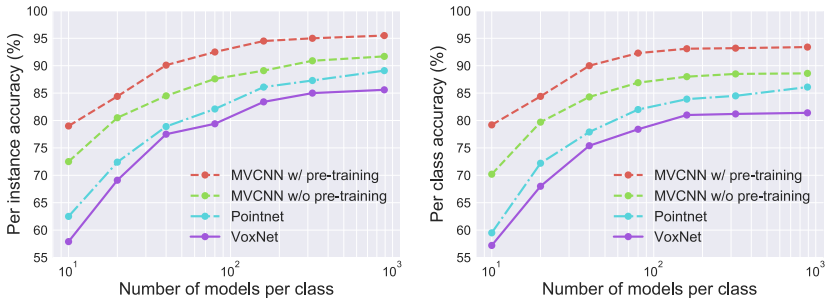


Fig. 3. Classification accuracy as a function of training set size. MVCNN generalizes better than the other approaches. The two MVCNN curves correspond to variants with and without ImageNet pretraining.

Model	Per class	Per instance
VGG-11	92.4	95.0
ResNet 18	92.8	95.6
ResNet 34	93.4	95.9
ResNet 50	94.0	95.5

Table 1. Accuracy (%) of MVCNN with different CNN architectures. The VGG-11 architectures are on par with the residual network variants.

Effect of ImageNet pretraining. MVCNN benefits from transfer learning from ImageNet classification task. However, even without ImageNet pretraining, the MVCNN achieves 91.3% per-instance accuracy (Table 2). This is higher than several point-based and voxel-based approaches. Figure 3 plots the performance of the MVCNN with VGG-11 network without ImageNet pretraining across training set sizes showing this trend is true throughout the training regime. In Section 3.3 we study if ImageNet pretraining can benefit such approaches using cross-modal transfer learning.

Model	Per class	Per instance
VGG-11 w/ ImageNet pretraining	92.4	95.0
VGG-11 w/o ImageNet pretraining	88.7	91.3

Table 2. Effect of ImageNet pretraining on the accuracy (%) of MVCNN. The VGG-11 architecture is used and the full training/test split of the ModelNet40 dataset is used.

Effect of shape rendering. We analyze the effect of different rendering approaches for input to a MVCNN model in Table 3. Sphere rendering proposed in [24] refers to rendering each point as a sphere and was shown to improve performance with AlexNet MVCNN architectures. We first compared the tight field-of-view rendering with black background in this work to the rendering in [31]. Since [31] only reported results on the 80/20 training/test split, we first compared the performance of the VGG-11 networks using images from [31]. The performance difference was negligible. However with our shaded images the performance of the VGG-11 network improves by more than 2%.

Using depth images, the per instance accuracy is 3.4% lower than using shaded images, but concatenating shaded images with depth images gives 1.2% improvement. Furthermore, we found the shading information only provides 1.4% improvements over the binary silhouette images. This suggests that most of the discriminative shape information used by the MVCNN approaches lie in the boundary of the object.

Model	Rendering	Full training/test		80/20 training/test	
		Per class	Per instance	Per class	Per instance
VGG-M	Shaded from [31]	-	-	89.9	89.9
VGG-M	Shaded from [31] (80×)	-	-	90.1	90.1
VGG-11	Shaded from [31]	-	-	89.1	89.1
VGG-11	Shaded	**92.4**	**95.0**	**92.4**	**92.4**
VGG-11	Depth	89.8	91.6		
VGG-11	Shaded + Depth	94.7	96.2		
VGG-11	Silhouettes	90.7	93.6		
AlexNet	Sphere rendering (20×)	89.7	92.0		
AlexNet-MR	Sphere rendering (20×)	91.4	93.8		

Table 3. Accuracy (%) of MVCNN with different rendering methods. The number in the brackets are the number of views used. 12 views are used if not specified.

3.3 Cross Modal Distillation

Knowledge distillation [4,10] was proposed for model compression tasks. They showed the performance of the model can be improved by training to imitate the output of a more accurate model. This technique has also been applied on transferring rich representations across modalities. For example, a model trained with images can be used to guide learning of a model for depth images [7], or to a model for sound waves [1]. We investigate such techniques for learning across different 3D representations; Specifically from MVCNN model to PointNet and VoxNet models.

To do this we first train the ImageNet initialized VGG-11 network on the full training set. The logits (the last layer before the softmax) are extracted on the training set. A PointNet (or VoxNet) model is then trained to minimize

$$\sum_{i=1}^{n} \mathcal{L}\left(\sigma(\mathbf{z}_i), y_i\right) + \lambda \sum_{i=1}^{n} \mathcal{L}\left(\sigma\left(\frac{\mathbf{z}_i}{T}\right), \sigma\left(\frac{\mathbf{x}_i}{T}\right)\right) \tag{2}$$

where $\mathbf{x}_i$ and $\mathbf{z}_i$ are the logits from the MVCNN model and from the model being trained respectively, y_i is the class label of the input $\mathbf{s}_i$, $\sigma(x)$ is the softmax function, and $\mathcal{L}$ is the cross-entropy loss $\mathcal{L}(p, q) = -\sum p_i \log q_i$. T is the temperature for smoothing the targets. λ, T are set by grid search for $T \in [1, 20], \lambda \in [1, 100]$. For example, in PointNet the best hyper-parameters are $T = 20, \lambda = 50$ when training set is small, and $T = 15, \lambda = 10$ when the training set is larger. In VoxNet we set $T = 10, \lambda = 100$ in all cases. Figure 4 shows the result of training

VoxNet and PointNet with distillation. For VoxNet the per instance accuracy is improved from 85.6% to 87.4% with whole training set; For PointNet the accuracy is improved from 89.1% to 89.4%. The improvement is slightly bigger when there is less training data.

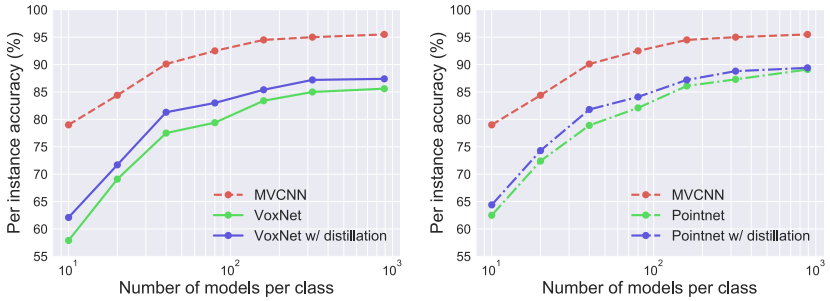


Fig. 4. Model distillation from MVCNN to VoxNet and PointNet. The accuracy is improved by 1.8% for VoxNet and 0.3% for PointNet with whole training set.

3.4 Tradeoffs between Learned Representations

In this Section we analyze the tradeoffs between the different shape classifiers. Table 4 compares their speed, memory, and accuracy. The MVCNN model has more parameters and is slower, but the accuracy is 5.9% better than PointNet and 9.4% better than VoxNet. Even though the number of FLOPS are far higher for MVCNN the relative efficiency of 2D convolutions results in slightly longer evaluation time compared to VoxNet and PointNet.

We further use an ensemble model combining images, voxels, and point cloud representations. A simple way is to average the predictions from different models. As shown in Figure 5, the ensemble of VoxNet and PointNet has better performance than using single model. However, the predictions from MVCNN dominate VoxNet and PointNet and gives no benefit for combining the predictions from other models with MVCNN. A more complex scheme where we trained a linear model on top of features extracted from penultimate layers of these networks did not provide any improvements either.

3.5 Robustness to Adversarial Examples

In this section we analyze and compare the robustness of three shape classification models to adversarial examples. Adversarial examples are generated using the stochastic gradient ascent procedure described in Section 2.3. We search the threshold ϵ from 0.001 to 0.9, and find the minimum value of ϵ where we can generate an adversarial example in 1000 iterations with learning rate $\alpha = 1 \times 10^{-6}$.

To make a quantitative analysis between VoxMVCNN and VoxNet, we use the following procedure. Given an input $\mathbf{s}$ and a classifier $\phi(\mathbf{s}, y)$, the "hardest"

Model	Forward-pass time	#params	Memory (GB)	Per class acc. (%)	Per ins. acc. (%)
MVCNN	25.8 ms	128.9M	10.0	92.4	95.0
VoxNet	1.3 ms	1.4M	2.0	81.4 (82.5)	85.6 (87.4)
PointNet	3.1 ms	3.5M	4.4	86.1 (86.7)	89.1 (89.4)

Table 4. Accuracy, speed and memory comparison of different models. Memory usage during training which includes parameters, gradients, and layer activations, for a batch size 64 is shown. Forward-pass time is also calculated with batch size 64 using PyTorch with a single GTX Titan X for all the models. The input resolutions are 224×224 for MVCNN, 32^3 for VoxNet, and 1024 points for PointNet. The accuracy numbers in brackets are for models trained with distillation as described in Section 3.3.

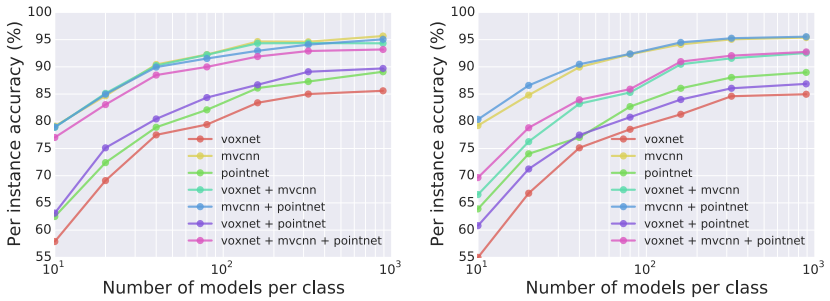


Fig. 5. Accuracy obtained by ensembling models. Left shows results by averaging the predictions while right shows results by training a linear model on the concatenated features extracted from different models.

target class is defined as $y'_{\mathbf{s},\phi(\mathbf{s},y)} = \arg\min_y \phi(\mathbf{s}, y)$. We select the first five models for each class from the test set. For each model, we select two target classes $y'_{\mathbf{s},\phi_1(\mathbf{s},y)}$ and $y'_{\mathbf{s},\phi_2(\mathbf{s},y)}$ where ϕ_1 is VoxMVCNN and ϕ_2 is VoxNet. There are total 400 pairs of test cases $(\mathbf{s}, y')$. We say an adversarial example $\mathbf{s}'$ can be found when $y' = \arg\max_y \phi(\mathbf{s}', y)$ with $\epsilon \leq 0.9$.

As shown in Table 5, we can generate 399 adversarial examples out of 400 test cases for VoxMVCNN, but only 370 adversarial examples for VoxNet. We then report the minimum ϵ where adversarial examples can be found in each test case. The average ϵ of VoxMVCNN is smaller than VoxNet, which suggests that VoxMVCNN is easier to be fooled by adversarial examples than VoxNet. We also use the VoxMVCNN model trained without ImageNet pre-training. Surprisingly, the model without pre-training is more robust to adversarial examples as the mean of ϵ is bigger than the model with pre-training. As for PointNet, we can generate 379 adversarial examples. Note that the ϵ value here is not comparable with other voxel representations, since in this case we are changing point coordinates instead of occupancy levels.

We also show some qualitative adversarial examples in Figure 6 and Figure 7. The voxels are scaled according to their occupancy level. In Figure 6 the input and the target classes are the same for each row, where the target labels are cup, keyboard, bench, and door from top to bottom. In Figure 7 we use the

same input model but set different target class for each column. The differences of adversarial examples for VoxMVCNN are almost imperceptible, as the ϵ is too small and the model is easy to be fooled. The classification accuracy of each model is shown in Table 5 as reference.

	PointNet	VoxNet	VoxMVCNN w/o pre-training	VoxMVCNN w/ pre-training
# Adversarial examples	379	370	290	399
ϵ	0.045 ± 0.054	0.061 ± 0.057	0.041 ± 0.027	0.006 ± 0.006
Per inst. acc. (%)	86.8	85.6	84.4	88.2

Table 5. Robustness of three models to adversarial examples. We generate adversarial examples with 400 test cases $(\mathbf{s}, y')$ for each model. The ϵ defines a L_∞ ϵ-neighborhood of the values that are either point coordinates or voxel occupancy. We report the average of the minimum ϵ where an adversarial example can be found. Bigger ϵ means more robustness to adversarial examples. The classification accuracies are reported in the last row as reference. We use our implementation of PointNet in PyTorch [20] for generating adversarial examples.

3.6 Comparison to Prior Work

We compare our MVCNN result with prior works in Table 6. The results are grouped by the input type. For multi-view image-based models, our MVCNN achieves 95.0% per instance accuracy, which is the best result between all competing approaches. Rotation Net [13], which predicts the object pose and class labels at the same time, is 0.2% worse than our MVCNN. Dominant Set Clustering [34] works by clustering image features across views and pooling within the clusters. Its performance is 1.0% lower than RotationNet. MVCNN-MultiRes [23] is the most related to our work. They showed that MVCNN with sphere rendering can achieve better accuracy than voxel-based network, suggesting that there is room for improvement in VoxNet. Our VoxNet experiment corroborates to this conclusion. Furthermore, MVCNN-MultiRes uses images in multiple resolutions to boost its performance.

For point-based methods, PointNet [32] and DeepSets [38] use symmetric functions, i.e. max/mean pooling layers, to generate permutation invariant point cloud descriptions. DynamicGraph [36] builds upon PointNet by performing symmetric function aggregations on points within a neighborhood, instead of the whole set. Such neighborhood is computed dynamically by building nearest neighbor graph using distances defined in the feature space. Similarly, Kd-Networks [15] work by precomputing a graph induced by a binary spatial partitioning tree and use it to apply local linear operations. The best point-based method is 2.8% less accurate then our MVCNN.

For voxel-based methods, VoxNet [17] and 3DShapeNets [37] work by applying 3D convolutions on voxels. ORION [27] is based on VoxNet but predicts the orientation in addition to class labels. OctNet [25] and O-CNN [35] are able to process higher resolution grids by using an octree representation. FusionNet [9] combines the voxel and image representations to improve the performance to 90.8%. Our experiments in Section 3.4 suggests that since MVCNN already has 95.0% accuracy the benefit of combining different representations is not effective.

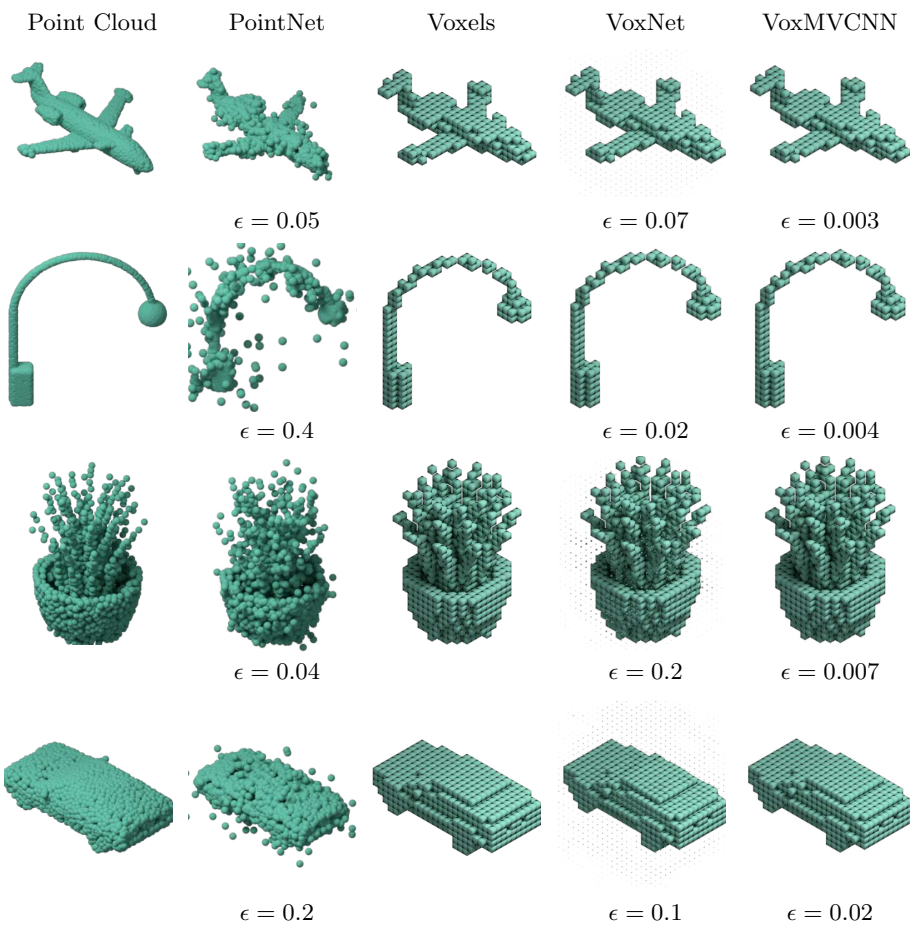


Fig. 6. Adversarial examples of PointNet, VoxNet, and VoxMVCNN. The shapes are misclassified as cup, keyboard, bench, and door for each row from top to bottom. Voxel size represents occupancy level.

4 Conclusion

We investigated on different representations and models for 3D shape classification task, which resulted in a new state-of-the-art on the ModelNet40 benchmark. We analyzed the generalization of MVCNN, PointNet, and VoxNet by varying the number of training examples. Our results indicate that multiview-based methods provide better generalizability and outperform other methods on the full dataset even without ImageNet pretraining or training with binary silhouettes. We also analyzed cross-modal distillation and showed improvements on VoxNet and PointNet by distilling knowledge from MVCNN. Finally, we analyzed the robustness of the models to adversarial perturbations and concluded that point-based networks are more robust to point perturbations, while multiview and voxel-based networks can be fooled by imperceptible perturbations.

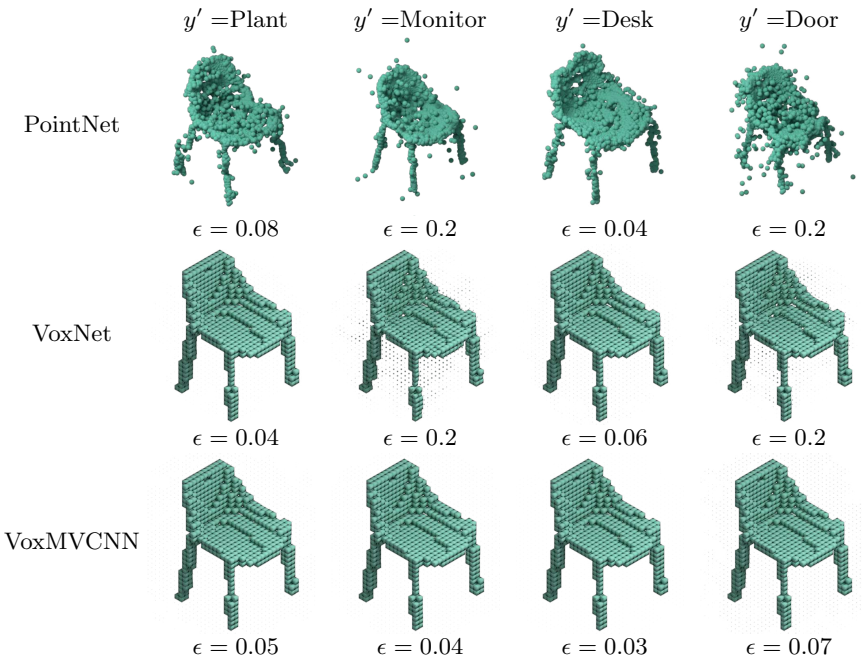


Fig. 7. Adversarial examples of PointNet, VoxNet, and VoxMVCNN. The shapes are misclassified as plant, monitor, desk, and door for each column from left to right.

Model	Input	Per class acc.	Per ins. acc.
Our MVCNN	Images	**92.4**	**95.0**
RotationNet [13]		-	94.8
Dominant Set Clustering [34]		-	93.8
MVCNN-MultiRes [23]		91.4	93.8
PANORAMA-NN [28]			90.7
MVCNN [31]		90.1	90.1
DynamicGraph [36]	PC	90.2	92.2
Kd-Networks [15]		-	91.8
LocalFeatureNet [29]		-	90.8
PointNet++ [22]		-	90.7
DeepSets [38]		-	90.0
PointNet [32]		86.2	89.2
VRN Single [3]	Voxels	-	91.3
O-CNN [35]			90.6
ORION [27]		-	89.7
VoxNet [17]		-	83.0
3DShapeNets [37]		77.3	84.7
PointNet++ [22]	PC+Normal	-	91.9
FusionNet [9]	Voxels+Images	-	90.8

Table 6. Accuracy (%) of state-of-the-art methods with different 3D representations. PC refers to point clouds. The order is grouped by input type and sorted by accuracy.

Acknowledgment We acknowledge support from NSF (#1617917, #1749833) and the MassTech Collaborative grant for funding the UMass GPU cluster.

References

1. Aytar, Y., Vondrick, C., Torralba, A.: Soundnet: Learning sound representations from unlabeled video. In: Neural Information Processing Systems (NIPS) (2016)
2. Blender Online Community: Blender - a 3D modelling and rendering package. Blender Foundation, Blender Institute, Amsterdam, http://www.blender.org
3. Brock, A., Lim, T., Ritchie, J.M., Weston, N.: Generative and discriminative voxel modeling with convolutional neural networks. arXiv preprint arXiv:1608.04236 (2016)
4. Buciluǎ, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD) (2006)
5. Gadhela, M., Maji, S., Wang, R.: Unsupervised 3d shape induction from 2d views of multiple objects. In: International Conference on 3D Vision 2018 (2017)
6. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (ICLR) (2015)
7. Gupta, S., Hoffman, J., Malik, J.: Cross modal distillation for supervision transfer. In: Computer Vision and Pattern Recognition (CVPR) (2016)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Computer Vision and Pattern Recognition (CVPR) (2016)
9. Hegde, V., Zadeh, R.: Fusionnet: 3d object classification using multiple data representations. arXiv preprint arXiv:1607.05695 (2016)
10. Hinton, G., Vinyals, O., Dean, J.: Distilling knowledge in a neural network. In: Neural Information Processing Systems (NIPS) (2014)
11. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (ICML) (2015)
12. Kalogerakis, E., Averkiou, M., Maji, S., Chaudhuri, S.: 3d shape segmentation with projective convolutional networks (2017)
13. Kanezaki, A., Matsushita, Y., Nishida, Y.: Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. arXiv preprint arXiv:1603.06208 (2016)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. Klokov, R., Lempitsky, V.: Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In: International Conference on Computer Vision (ICCV) (2017)
16. Kurakin, A., Goodfellow, I.J., Bengio, S.: Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533 (2016)
17. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: IEEE International Conference on Intelligent Robots and Systems (IROS) (2015)
18. Meagher, D.J.: Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer. Electrical and Systems Engineering Department Rensseiaer Polytechnic Institute Image Processing Laboratory (1980)
19. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Computer Vision and Pattern Recognition (CVPR) (2015)

20. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
21. Phong, B.T.: Illumination for computer generated pictures. Communications of the ACM **18**(6), 311–317 (1975)
22. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Neural Information Processing Systems (NIPS) (2017)
23. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.: Volumetric and multi-view cnns for object classification on 3d data. In: Computer Vision and Pattern Recognition (CVPR) (2016)
24. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.: Volumetric and multi-view cnns for object classification on 3d data. In: Computer Vision and Pattern Recognition (CVPR) (2016)
25. Riegler, G., Ulusoy, A.O., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: Computer Vision and Pattern Recognition (CVPR) (2017)
26. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015)
27. Sedaghat, N., Zolfaghari, M., Amiri, E., Brox, T.: Orientation-boosted voxel nets for 3d object recognition. British Machine Vision Conference (BMVC) (2017)
28. Sfikas, K., Theoharis, T., Pratikakis, I.: Exploiting the panorama representation for convolutional neural network classification and retrieval. In: Eurographics Workshop on 3D Object Retrieval (2017)
29. Shen, Y., Feng, C., Yang, Y., Tian, D.: Neighbors do help: Deeply exploiting local structures of point clouds. arXiv preprint arXiv:1712.06760 (2017)
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
31. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.G.: Multi-view convolutional neural networks for 3d shape recognition. In: International Conference on Computer Vision (ICCV) (2015)
32. Su, H., Qi, C., Mo, K., Guibas, L.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Computer Vision and Pattern Recognition (CVPR) (2017)
33. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
34. Wang, C., Pelillo, M., Siddiqi, K.: Dominant set clustering and pooling for multi-view 3d object recognition. In: British Machine Vision Conference (BMVC) (2017)
35. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions on Graphics (SIGGRAPH) **36**(4) (2017)
36. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. arXiv preprint arXiv:1801.07829 (2018)
37. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Computer Vision and Pattern Recognition (CVPR) (2015)
38. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R.R., Smola, A.J.: Deep sets. In: Neural Information Processing Systems (NIPS) (2017)