Neural Processing Letters (2020) 52:581–602
https://doi.org/10.1007/s11063-020-10268-x

Latent-MVCNN: 3D Shape Recognition Using Multiple Views from Pre-defined or Random Viewpoints

Qian Yu[1] · Chengzhuan Yang[2] · Honghui Fan[1] · Hui Wei[3]

Published online: 29 May 2020

Abstract
The Multi-view Convolution Neural Network (MVCNN) has achieved considerable success in 3D shape recognition. However, 3D shape recognition using view-images from random viewpoints has not been yet exploited in depth. In addition, 3D shape recognition using a small number of view-images remains difficult. To tackle these challenges, we developed a novel Multi-view Convolution Neural Network, "Latent-MVCNN" (LMVCNN), that recognizes 3D shapes using multiple view-images from pre-defined or random viewpoints. The LMVCNN consists of three types of sub Convolution Neural Networks. For each view-image, the first type of CNN outputs multiple category probability distributions and the second type of CNN outputs a latent vector to help the first type of CNN choose the decent distribution. The third type of CNN outputs the transition probabilities from the category probability distributions of one view to the category probability distributions of another view, which further helps the LMVCNN to find the decent category probability distributions for each pair of view-images. The three CNNs cooperate with each other to the obtain satisfactory classification scores. Our experimental results show that the LMVCNN achieves competitive performance in 3D shape recognition on ModelNet10 and ModelNet40 for both the pre-defined and the random viewpoints and exhibits promising performance when the number of view-images is quite small.

Keywords 3D shape recognition · 3D shape classification · 3D shape retrieval

✉ Qian Yu
yuqian@jsut.edu.cn

Chengzhuan Yang
chengzhuanyang@zufe.edu.cn

Honghui Fan
fanhonghui@jsut.edu.cn

Hui Wei
weihui@fudan.edu.cn

[1] School of Computer Engineering, Jiangsu University of Technology, No. 1801 Zhongwu Road, Changzhou 213001, China

[2] School of Information, Zhejiang University of Finance and Economics, No. 18 Xueyuan Road, Hangzhou 310018, China

[3] School of Computer Science, Fudan University, No. 825 Zhangheng Road, Shanghai 201203, China

1 Introduction

In the last few years, people have created large amounts of 3D shape data by using advanced sensing devices and CAD software. Due to the increasing demands from autonomous driving and other related applications, 3D shape analysis has emerged as a critical topic. Three-dimensional shape recognition is a fundamental task with broad application prospects. Because of the outstanding performance of deep learning in many computer vision tasks, deep neural networks have been applied to 3D shape analysis. Various deep neural networks have been proposed for 3D shape recognition that is based on different formats of 3D shape, such as views, voxel, mesh and point cloud. However, among these methods, only the view-based methods have achieved outstanding performance. Su et al. [27] first proposed the Multi-view Convolution Neural Network (MVCNN) to recognize 3D shapes, and developed the view-pooling layer to synthesize the information from all views into a single and compact 3D shape descriptor. The view-pooling layer applies the max operation across the convolutional features of all views discarding the smaller feature values. Although this view-pooling layer provides a convenient way to aggregate views, it obviously limits the performance improvement of MVCNN because it does not make full use of all of the view information.

Group-View Convolutional Neural Networks (GVCNN) [9] was proposed to better incorporate the intrinsic hierarchical correlation and discriminability among views, which uses a complex view-pooling module instead of the max view-pooling layer (with view grouping and fusion) to more effectively aggregate views. Compared with MVCNN, GVCNN achieved a better accuracy for 3D shape recognition by using the improved view-pooling module. Note that both MVCNN and GVCNN make use of the view information by fusing of the convolutional features of all views. However, another effective way to aggregate views is to combine the classification scores from the view-images. A deep neural network processes every view-image independently to output a category probability distribution, then average the category probability distributions of all views as the final output. This idea is extended by RotationNet, which was proposed by Kanezaki et al. [15]. RotationNet achieves the state-of-the-art performance on 3D shape recognition. The final layer of RotationNet outputs many category probability distributions for each view-image, and each is implemented by a softmax layer and corresponds to an object pose. RotationNet uses a latent variable as the 3D object pose estimation. However, the latent variable has only one dimension, and its range is very small. In RotationNet, the latent variable can be assigned to one of the integers from one to the number of views. Moreover, RotationNet is very sensitive to the pre-defined view assumption. In general, these methods have not been applied to 3D shape recognition for random viewpoints. In addition, when the number of the view-images is quite small, 3D shape recognition will face great challenge.

In order to facilitate the CNN's ability with 3D shape recognition for pre-defined and random viewpoints, and deal with the case of a small number of available view-images, we developed a novel Multi-view Convolution Neural Network "Latent-MVCNN" (LMVCNN), that recognizes 3D shapes and consists of three types of CNNS. The first type of CNN outputs multiple category probability distributions for each view-image as RotationNet, and the second type of CNN outputs a latent vector to help the first type of CNN choose the decent distribution. The role of the latent vector is similar to the view Grouping Module in GVCNN, but here it is used to choose the decent distribution. The third type of CNN outputs the transition probabilities from the category probability distributions of one view to the category probability distributions of another view, which further helps LMVCNN to find the decent category probability distributions for each pair of view-images.

There are much more object poses in the random viewpoint setup than in the pre-defined viewpoint setup. Therefore, it is difficult for RotationNet to output many category probability distributions to correspond to large number of object poses for each input view-image. If we give up the assumption that each category probability distribution must correspond to an object pose, then the model for 3D shape recognition may work better in the random viewpoint setup. Accordingly, LMVCNN is built based on this idea. Note that RotationNet selects the category probability distribution independently for each input view-image by using the view variable, while the three CNNs in LMVCNN cooperate with each other to attempt to obtain a feasible combination of the category probability distributions from the different views as the classification result. In this way, LMVCNN would have a larger representational capacity and can get better classification scores under random viewpoint setup than other methods.

In the case of limited number of available images, LMVCNN still attempts to choose a feasible combination of the category probability distributions to recognize 3D shape, rather than selects the category probability distributions independently for each input view-image. Therefore, LMVCNN also shows promising performance of recognizing 3D shapes when the number of view-images is quite small.

The main contributions of this paper are as follows. First, we proposed a novel problem to recognize 3D object using multiple view-images from random viewpoints, which is much more challenging than using pre-defined viewpoints. Second, we show that LMVCNN performs well and remains competitive with other related methods for both the pre-defined and the random viewpoints. Third, LMVCNN works well even when the number of view-images is quite small. The rest of this paper is organized as follows. We first introduce the related work in Sect. 2. We then present our proposed LMVCNN architecture in Sect. 3. Experiments and discussions are provided in Sect. 4. Finally, we conclude this paper in Sect. 5.

2 Related Work

Typically, 3D shapes come in different raw formats, such as views, voxel, mesh and point cloud. Here, we review some typical methods for 3D shape recognition based on different formats.

Voxel-based methods Voxel-based methods usually transform the 3D shape into a volume consisting of many voxels. Therefore, a CNN can be extended into a 3D architecture to easily process the voxelized shape. Wu et al. [31] represented a 3D shape as a probability distribution of binary variables on a 3D voxel grid by using the Convolutional DeepBelief Network. This model can be used for joint object recognition and shape completion from a 2.5D depth map. Qi et al. [21] aimed to improve both volumetric CNNs and multi-view CNNs by performing an extensive analysis of the existing approaches. There are many other generative models for describing and generating 3D objects based on the volume format. Wu et al. [30] first proposed a 3D Generative Adversarial Network, which generates 3D objects from a probabilistic space by using volumetric convolutional networks and generative adversarial nets. Xie et al. [32] developed a deep convolutional energy-based model for modeling volumetric shapes. However, when the resolution of 3D volumetric shape increases, the total number of the voxels grows immensely so that the volumetric shape cannot be processed. Due to this limitation of voxel-based methods, their performance is usually not as good as that of view-based methods.

Mesh-based methods An alternative approach is to represent the 3D object as a polygon mesh or a graph. Hamza [11] developed a graph-theoretical approach for 3D shape classification

using graph regularized sparse coding with a biharmonic distance map. There are some methods that convert the graph into its spectral representation, and then execute convolution in the spectral domain [3,34]. There is another class of methods that transform the input shapes into 2D parametric domains and then perform convolutions in these domains. Ayan et al. [25] first converted the mesh of a 3D object into the geometry images and then performed convolution on them. Ghodratiet al. [10] proposed a framework that first extracts the local feature of vertices of 3D shape, then represents each 3D shape as a 2D image, and then feeds the 2D image into a convolutional neural network to learn the representation of 3D shape. The method proposed by [18] parameterizes the surface into local patches and carry out surface-based convolution within these patches. Bronstein et al. [2] provided a detailed review of the principle and applications of the graph-based deep learning.

Point-cloud-based methods Qi et al. [5] designed a novel type of neural network, PointNet, that directly processes point clouds, and respects the permutation invariance of the input points. This neural network provides an efficient and effective approach for 3D object recognition. Next, Qi et al. [22] developed another hierarchical neural network, PointNet++, that uses PointNet recursively on a nested subset of the input points. Recently, Su et al. [26] proposed a network for processing point clouds that directly operates on the points, and it is represented as a sparse set of samples in a high-dimensional lattice. Point cloud processing has always been an important module in autonomous driving. However, the point cloud format is susceptible to noise interference and lacks the ability to describe local details.

View-based methods Compared with other methods mentioned above, the view-based 3D shape representation methods have a better adaptability and can more easily to acquire 2D view-images. Su et al. [27] first describes the general framework of Multi-view Convolution Neural Network (MVCNN). MVCNN is an effective method for obtaining a representation of a 3D object, and it usually includes three steps: first, take 2D images of 3D objects from different viewpoints with cameras; second, extract convolution features of these images with the CNN; finally, fuse these convolution features with the View Pooling Layer entering the full connection layer. Qi et al. [21] improved the performance of MVCNN by combining the convolution network features of a multi-resolution 3D shape. There are many other methods to further improve MVCNN. The Group-View Convolutional Neural Networks (GVCNN) [9] contains a complex view-pooling module instead of a max view-pooling layer, including view grouping and fusion, to more effectively aggregate views. RotationNet was proposed by Kanezaki et al. [15], which outputs multiple classification results for each input view-image and achieves the state-of-the-art performance on 3D shape recognition. Bu et al. [4] proposed a novel 3D feature learning framework to promote the discriminability for 3D shape recognition. Yu et al. [35] obtained an effective 3D object representation by aggregating the local convolutional features through a novel bilinear pooling. Cohen et al. [8] built a spherical convolution neural network, that projects the object into sphere surface and then takes into account all of the spherical information, but the recognition accuracy still has much room for growth. In 3D object retrieval, there are also many view-based methods. Nie et al. [19] developed a composition-based multi-graph matching method for 3D model retrieval. The classic method proposed by He et al. [13] achieves good retrieval performance using Triplet-Center Loss.

The proposed LMVCNN is one of view-based methods, which is motivated by MVCNN [27] and RotationNet [15]. MVCNN outputs one classification result independently for each input view-image. In order to improve the performance of MVCNN, RotationNet outputs many category probability distributions for each view-image, and each is a classification

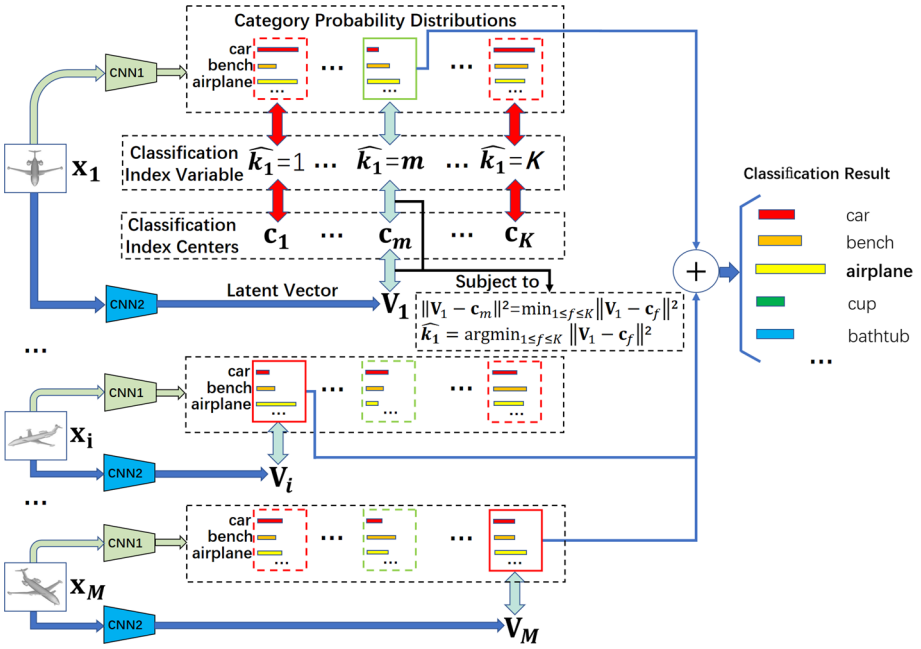


Fig. 1 Overview of the first and second types of CNNs (CNN1 and CNN2) in LMVCNN. The first type of CNN outputs K category probability distributions, with each corresponding to one classification index variable and one classification index center. Each category probability distribution is represented as a histogram with N bins, whose norm is 1. Each bin of one histogram represents the probability that the input object belongs to the corresponding category. The second type of CNN outputs a latent vector to find a decent category probability distribution with a classification index center close to the output latent vector. The operation "+" means that the sum of the log likelihoods is used to get the final classification score. The input object is assigned to the category with the highest classification score

result for the input view-image and corresponds to an object pose. RotationNet owns the assumption that each category probability distribution must correspond to an object pose. LMVCNN was developed without this assumption to better deal with the case of 3D shape recognition in the random viewpoint setup.

3 Latent-MVCNN

In this section, we discuss the proposed LMVCNN in detail. LMVCNN consists of three types of CNNS, as illustrated in Figs. 1 and 2 . The bone network of these CNNS is made up of one type of image classification network, such as AlexNet [17], VggNet [24], and ResNet [12]. Let M be the number of pre-defined or random viewpoints and N be the number of target object categories. We can obtain M view-images $\{\mathbf{x}_i | 1 \leq i \leq M\}$, denoted by $\{\mathbf{x}_i\}_1^M$, by rendering a 3D object under those M viewpoints. As shown in Fig. 1, the last layer of the first type of the CNN outputs K category probability distributions $\{P(\hat{y}|\mathbf{x}_i, \hat{k}_i)\}_1^K$, each of which is a classification result for the input view-image and represented by a softmax layer with N dimension, where $\hat{y}$ is the category label variable and $\hat{k}_i$ is the index variable of the category probability distributions. If $\{\hat{k}_i\}_1^M$ has been exactly determined, we can compute the whole probability of category y_{label} for M view-images:

$$\prod_{i=1}^{M} P(\hat{y} = y_{label}|\mathbf{x}_i, \hat{k}_i). \tag{1}$$

The probability is usually less than 1, so the product of multiple probabilities will result in a very small value. To address this, Eq. (1) is rewrote as the sum of the log likelihoods:

$$\sum_{i=1}^{M} log\, P(\hat{y} = y_{label}|\mathbf{x}_i, \hat{k}_i). \tag{2}$$

Here, the key step is to determine $\{\hat{k}_i\}_1^M$. Therefore, we developed the second type of CNN, which selects a suitable category probability distribution. As illustrated in Fig. 1, this CNN outputs a **latent vector** $\mathbf{V}_i$ and its activation function of the last layer is set as sigmoid. Because no supervised signal of $\mathbf{V}_i$ is provided, $\mathbf{V}_i$ is a latent vector. The benefit of using a vector rather than a variable is that the neural network outputting a vector will have greater representational capacity. We introduced K classification index centers $\{\mathbf{c}_f\}_1^K$ with the same dimension to the output latent vector, and these correspond to the category probability distributions respectively.

If the output latent vector $\mathbf{V}_i$ is closest to the classification index centers $\mathbf{c}_m$, then it means that $\hat{k}_i$ is more likely assigned to the index of the m-th category probability distribution. As shown in Fig. 1, $\mathbf{V}_i$, $\mathbf{c}_m$ and $\hat{k}_i$ should be subject to the following equations:

$$\begin{aligned} \|\mathbf{V}_i - \mathbf{c}_m\|^2 &= \min_{1 \le f \le K} \|\mathbf{V}_i - \mathbf{c}_f\|^2, \\ \hat{k}_i &= \arg \min_{1 \le f \le K} \|\mathbf{V}_i - \mathbf{c}_f\|^2. \end{aligned} \tag{3}$$

In other words, the smaller the distance from $\mathbf{V}_i$ to $\mathbf{c}_m$, the greater the probability the index m is assigned to $\hat{k}_i$. By using the softmax function, we defined the probability as follows:

$$P(\mathbf{V}_i, \mathbf{c}_f) = \frac{\exp -\|\mathbf{V}_i - \mathbf{c}_f\|^2}{\sum_{l=1}^{K} \exp -\|\mathbf{V}_i - \mathbf{c}_l\|^2}. \tag{4}$$

In the training phase, we defined the following optimization problem using the Negative Log Likelihoods:

$$\begin{aligned} \max \sum_{i=1}^{M} [-log\, P(\hat{y} = y_{label}|\mathbf{x}_i, \hat{k}_i) - log\, P(\mathbf{V}_i, \mathbf{c}_{\hat{k}_i}) \\ - \ log(1 - \sum_{f \neq \hat{k}_i} P(\mathbf{V}_i, \mathbf{c}_f))] \end{aligned} \tag{5}$$

The first item in Eq. (5) ensures that the input view-images $\{\mathbf{x}_i\}_1^M$ should be classified as correctly as possible. The second and the third items show that if the latent vector $\mathbf{V}_i$ is closest to $\mathbf{c}_{\hat{k}_i}$, then $\mathbf{V}_i$ must be far from the other classification index centers. These centers are not identified at the initialization, and they have to be trained simultaneously with other model parameters.

As shown in Fig. 1, the second type of CNN uses only one view-image to obtain its index variable. In order to calculate better values for $\{\hat{k}_i\}_1^M$, the third type of CNN is introduced as shown in Fig. 2. The third type of CNN shares the weights and the pair of the third type of CNN forms a Siamese network. This network makes use of a pair of view-images to output the transition probabilities from the category probability distributions of one view

to the category probability distributions of another view, in other words, to develop the transition probabilities from the candidate values of $\hat{k}_i$ to the candidate values of $\hat{k}_j$, denoted as $P(\mathbf{x}_i, \mathbf{x}_j, \hat{k}_i, \hat{k}_j)$. The input of of the Siamese network is the pair of view-images and the next to the output is the subtraction of the features of the final convolution layers. The softmax layer with dimension K^2 is the output of the Siamese network. We added the third type of CNN into our method, and we rewrote Eq. (5) as the following:

$$\begin{aligned}\max \sum_{i=1}^{M} [&-log\, P(\hat{y} = y_{label}|\mathbf{x}_i, \hat{k}_i) - log\, P(\mathbf{V}_i, \mathbf{c}_{\hat{k}_i}) \\ &-log(1 - \sum_{f \neq \hat{k}_i} P(\mathbf{V}_i, \mathbf{c}_f))] - \sum_{j \neq i} log\, P(\mathbf{x}_i, \mathbf{x}_j, \hat{k}_i, \hat{k}_j),\end{aligned} \tag{6}$$

Because the parameters $\{\hat{k}_i\}_1^M$ is discrete, it cannot be optimized by the back-propagation algorithm. Therefore, **Eq. (6) is iteratively updated in two training steps**: our method first determines $\hat{k}_i$ according to the following Algorithm (1), and then updates the three types of CNNS and the classification index centers by fixing $\{\hat{k}_i\}_1^M$.

When the number of category probability distributions K and the number of input view-images M become large, it is much more difficult to predict $\{\hat{k}_i\}_1^M$. We developed a greedy algorithm, described in Algorithm (1), to quickly calculate the values of $\{\hat{k}_i\}_1^M$ individually. For simplicity, we defined a matrix $eng(m, \hat{k}_m = n)$ to denote the probability of the best assignment for $\{\hat{k}_i\}_1^m$ when $\hat{k_m} = n$, and determine $eng(m, \hat{k}_m = n)$ by maximizing the following equation:

$$\begin{aligned}eng(m, \hat{k}_m = n) = \max_{\hat{k}_{m-1}} \quad & eng(m-1, \hat{k}_{m-1}) - log\, P(\hat{y} = y_{label}|\mathbf{x}_m, n) \\ & - log\, P(\mathbf{V}_m, \mathbf{c}_n) - log(1 - \sum_{f \neq n} P(\mathbf{V}_m, \mathbf{c}_f)) \\ & - log\, P(\mathbf{x}_{m-1}, \mathbf{x}_m, \hat{k}_{m-1}, \hat{k}_m = n),\end{aligned} \tag{7}$$

Here, we used another matrix $pos(m, \hat{k}_m = n)$ to remember the best assignment H to $\hat{k}_{m-1}$ when $\hat{k_m} = n$ in Eq. (7). Note that H should be unique, in other words, H has not been assigned to any variable in $\{\hat{k}_i\}_1^m$ except $\hat{k}_{m-1}$. Our experiment results show that if H is not unique, LMVCNN will degenerate so that all index variables $\{\hat{k}_i\}_1^M$ are assigned the same value. Once the two matrices eng and pos have been determined, the decent values of $\{\hat{k}_i\}_1^M$ can be obtain according to Algorithm (1).

In the testing phase, for the input view-images, we first employed the three types of CNNs to obtain the category probability distributions, the latent vectors and the transition probabilities respectively, then obtained the decent values of $\{\hat{k}_i\}_1^M$ according to Algorithm (1), and finally computed the classification scores using Eq. (2).

Generally speaking, the proposed LMVCNN outputs many category probability distributions, and thus it can choose a decent one to the best of its ability for any view-image from both cases of pre-defined and random viewpoints. In addition, LMVCNN is able to process few view-images. Therefore, if the number of the test view-images is quite small, our method still works well.

View-image Generation To capture the 2D view-images of the 3D shape as completely as possible, we designed three types of viewpoint setups, such as Upright, Dodecahedron and Random setups, and then generated the rendering views from the 3D shape under these

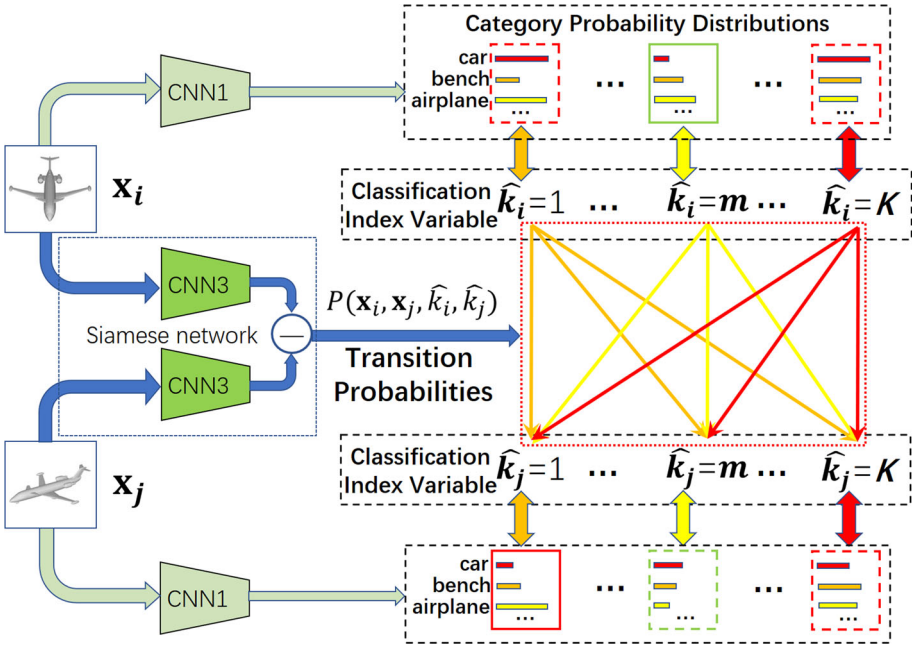


Fig. 2 Overview of the third type of CNN (CNN3) in LMVCNN. The first type of CNN (CNN1) is the same to that in Fig. 1. For convenience, the second type of CNN (CNN2) in Fig. 1 is not shown here. The output of the Siamese network is the transition probabilities from the category probability distributions of one view to the category probability distributions of another view. The operation "-" means the subtraction of the features of the final convolution layers

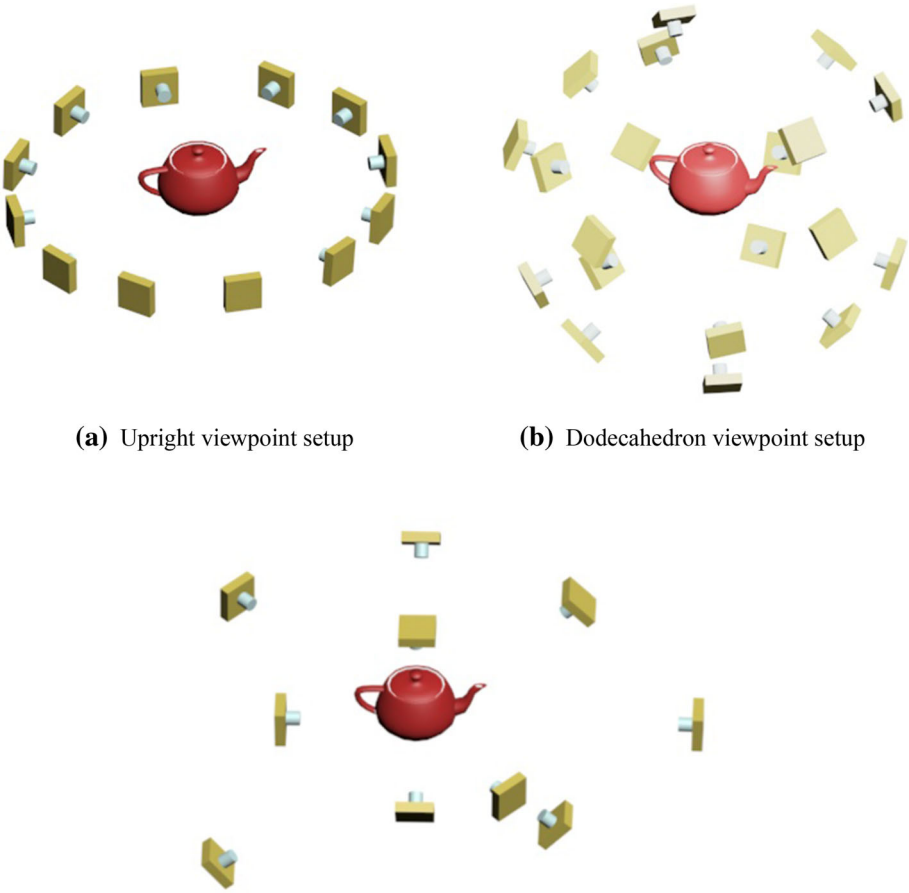
Algorithm 1 Greedy Calculation of $\{\hat{k}_i\}_1^M, \hat{k}_i \in \{1, 2..., K\}$

Input: Input parameters $\{\mathbf{x}_i\}_1^M$, $\{\mathbf{V}_i\}_1^M$, $\{\mathbf{c}_j\}_1^K$, the category label y_{label}, the number of input view-images M, and the number of category probability distributions K.

Output: Assignment to $\{\hat{k}_i\}_1^M$

$j \in \{1, 2..., K\}$
$eng(1, j) = -logP(\hat{y} = y_{label}|\mathbf{x}_1, j) - logP(\mathbf{V}_1, \mathbf{c}_j) - log(1 - \sum_{f \neq j} P(\mathbf{V}_1, \mathbf{c}_f))$;
$pos(1, j) = j$;
for m = 2:M **do**
  for n = 1:K **do**
    calculate $eng(m, n)$ according to Eq. (7),
    $pos(m, n)$ is equal to the best assignment H to $\hat{k}_{m-1}$ in Eq. (7),
    where H has not been assigned to any variable in $\{\hat{k}_i\}_1^m$ except $\hat{k}_{m-1}$;
$\hat{k_M} = arg\max_h eng(i, h)$;
for i = M-1:1 **do**
  $\hat{k_i} = pos(i + 1, \hat{k_{i+1}})$;
return $\{\hat{k_i}\}_1^M$;

viewpoints as shown in Fig. 3. **(Upright)** In the first case, we fix the z-axis as the rotation axis and then place viewpoints at intervals of the angle θ around the axis, elevated by 30° from the ground plane. We set $\theta = 30°$ to yield 12 views for an object. **(Dodecahedron)** We set up viewpoints on the M = 20 vertices of a dodecahedron encompassing the object. This is because a dodecahedron with 20 vertices is a regular polyhedron with the largest number



(a) Upright viewpoint setup

(b) Dodecahedron viewpoint setup

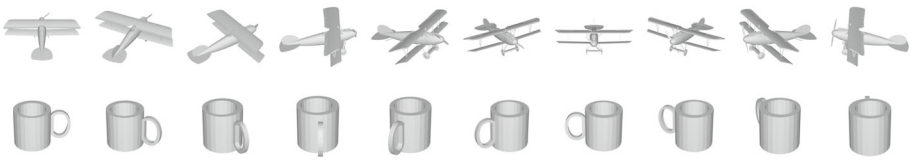
(c) Random viewpoint setup

Fig. 3 Illustration to three types of viewpoint setups. Each viewpoint is set at the center of one camera

of vertices. **(Random)** We randomly set the viewpoints of the camera. We uniformly select the azimuths and the elevations between 0° and 360°. In addition, the radial distance of the camera to the original coordinate is also set randomly. When people recognize a 3D object, their viewpoints cannot be defined at first and are usually random. Therefore, the view-images randomly generated are similar to those that people captured from the real world. Figure 4 shows some examples under different viewpoint setups. Note that 2D view-images generated from random viewpoints are variable due to the variation of the observed viewpoints.

4 Experiments

We ran some experiments on ModelNet10 and ModelNet40. ModelNet10 consists of 4,899 object instances in 10 categories, whereas ModelNet40 consists of 12,311 object instances in 40 categories. We rendered every object instance to obtain the view-images to generate the



(a) View-image examples generated under **upright viewpoint setup**.

(b) View-image examples generated under **dodecahedron viewpoint setup**.

(c) View-image examples generated under **random viewpoint setup**.

Fig. 4 Some view-image examples are generated under different viewpoint setups. Note that 2D view-images generated from random viewpoints can be large or small and vary due to differences in the observed viewpoints

training and test datasets. Our method uses two kinds of CNNs with different architectures, AlexNet [17] and VggNet [24]. The AlexNet architecture is smaller than the VGG-M network architecture used in MVCNN [27]. To train LMVCNN, we utilized the pre-trained model trained with the ILSVRC 2012 dataset and fine-tuned the weights of the pre-trained model on ModelNet10 and ModelNet40. The classical momentum SGD was used to train LMVCNN with a learning rate of 0.00001 and a momentum of 0.9. After the classification experiments on ModelNet10 and ModelNet40, we further used our model for 3D shape retrieval. All of the experiments were performed in the computer with four Titan Xp GPUs. Finally, we discussed the illustration to the Latent Vector.

4.1 3D Shape Classification

We conducted some experiments for the pre-defined viewpoints. A total of 12 and 20 views were captured in the upright and the dodecahedron cases, respectively. AlexNet and VggNet-11 pre-trained in the ILSVRC 2012 dataset were adopted and then fine-tuned on ModelNet10 and ModelNet40. We compared our method with some canonical methods, such as 3D ShapeNet [31] and PointNet [5], and some state-of-the-art view-based methods, such MVCNN [27], MVCNN-MultiRes [21], GVCNN [9], RotationNet [15], SliceNet [7], NormalNet [29], and PVR [36]. In order to make fair comparisons, if a method did not provide the classification results, we executed the code in the view-image set generated in our experiments to obtain the classification results, which are marked with an asterisk. As shown in Table 1, our proposed method outperforms the non-view-based methods. LMVNN_AlexNet with 12 pre-defined views achieves an accuracy of 91.8%, which is better than MVCNN-MultiRes

and MVCNN. Note that MVCNN-MultiRes uses Resnet [12], and MVCNN utilizes VggNet-M [6] as the pre-trained CNNs, which are much larger than AlexNet [17]. If the number of view-images increases from 12 to 20, LMVNN_AlexNet can achieve the same accuracy as GVCNN, but GVCNN uses GoogleNet [28] as its CNN architecture. LMVNN_VggNet-11 (VggNet with 11 weighted layers) with 12 pre-defined views obtains 92.8% and 96.6% accuracies on ModelNet40 and ModelNet10 respectively. When the number of view-images increases from 12 to 20, the accuracy of LMVNN_VggNet-11 has a slight decline on ModelNet10, but is better than that of other methods, including RotationNet [15]. As reported in [15], RotationNet has achieved 97.37% and 98.46% classification accuracies on ModelNet10 and ModelNet40 respectively. However, RotationNet must adjust the camera system to render many view-image sets. RotationNet was executed once for each view-image set, and the best accuracy of the 3D shape classification was chosen as the final output. In order to make fair comparisons, we performed RotationNet and LMVCNN in the same view-image sets generated in our experiments. In addition, we found it remarkable that RotationNet uses brute force to enumerate the values of the latent variable, while our methods utilize greedy programming to quickly calculate the assignments of the index variables as shown in Algorithm (1).

Next, some experiments were conducted for the random viewpoint setting. We generated 80 view-images from the random viewpoints, and then selected 10 groups of 12 (and 20) view-images. We executed our methods and other methods in 10 trials, and took the average accuracy as the final classification result. As shown in Table 2, both LMVCNN_AlexNet and LMVCNN_VggNet-11 performed better than MVCNN. Regardless of whether we used the random or pre-defined viewpoints, MVCNN has almost the same classification accuracy. This is because MVCNN uses the view-pooling layer with the max operation to aggregate views, so it does not make full use of all view information. Our methods were compared with RotationNet and achieves better performance for the random viewpoint setting on ModelNet10 and ModelNet40. In RotationNet, the latent variable was illustrated as the pose estimation, and can only be assigned to one of the integers from one to the number of views. However, the number of the pose is far greater than the number of views. In addition, RotationNet uses only one CNN to simultaneously output the classification results and the estimated pose, which may interfere with each other. Therefore, RotationNet is very sensitive to the pre-defined view assumption and does not work better than LMVCNN. As shown in Table 2, the classification accuracies of our proposed method almost exceed 90%, while the classification accuracies of other methods do not.

Figure 5 shows some classification results on the ModelNet40 and ModelNet10 datasets. The 3D shapes in the same row were classified into the same category. Note that the wrongly classified 3D shapes are very similar to the 3D shapes in the predicted category. Regardless of whether we investigated the pre-defined or the random viewpoints, the proposed LMVCNN performs well and remains competitive with other related methods. These experimental results clearly demonstrate the effectiveness of the proposed LMVCNN for 3D shape classification.

4.1.1 On the Number of View-Images

It is challenging to recognize a 3D object when the number of testing and training view-images is limited, but it is of great practical importance. This is because that people can perceive world to recognize 3D object with few views. In this section, we discuss two types of 3D shape classification experiments. First, LMVCNN was trained and tested when the train and test views were varied from 1 to 20. Second, LMVCNN was trained with a large number of view-images, and then tested with a small number of view-images. In the first case, we

Table 1 Comparison of the classification accuracies on the ModelNet40 and ModelNet10 datasets for the **pre-defined viewpoints**

Method	Training config.			Test config.	Classification	Classification
	Pre-train	Rendering	#Views	#Views	ModelNet40 Acc. (%)	ModelNet10 Acc. (%)
3D ShapeNet [31]	ModelNet40	–	–	–	77.3	49.2
PointNet [5]	–	–	–	–	89.2	–
MVCNN-MultiRes [21]	–	–	–	20	91.4	–
KD-Network [16]	–	–	–	–	91.8	–
NormalNet [29]	–	–	–	–	88.8	93.1
SliceNet [7]	–	–	–	–	89.9	92.7
PVR [36]	–	–	–	–	92.7	91.7
GVCNN [9]	ImageNet1K	Upright	12	12	92.6	–
MVCNN [27]	ImageNet1K	Upright	12	12	89.9	–
RotationNet* [15]	ImageNet1K	Upright	12	12	90.8	92.7
LMVCNN_AlexNet	ImageNet1K	Upright	12	12	91.8	95.2
LMVCNN_VggNet-11	ImageNet1K	Upright	12	12	92.8	**96.6**
MVCNN* [27]	ImageNet1K	Dodecahedron	20	20	88.2	89.5
RotationNet* [15]	ImageNet1K	Dodecahedron	20	20	93.3	94.6
LMVCNN_AlexNet	ImageNet1K	Dodecahedron	20	20	92.7	93.7
LMVCNN_VggNet-11	ImageNet1K	Dodecahedron	20	20	**93.5**	93.9

In order to make fair comparisons, we executed the code in the view-image set generated in our experiments to obtain the classification results which are marked with an asterisk *

Table 2 Comparison of the classification accuracies on the ModelNet40 and ModelNet10 datasets for the **random viewpoints**

Method	Training config.			Test config.	Classification	Classification
	Pre-train	Rendering	#Views	#Views	ModelNet40 Acc. (%)	ModelNet10 Acc. (%)
MVCNN* [27]	ImageNet1K	Random	12	12	88.1	89.2
RotationNet* [15]	ImageNet1K	Random	12	12	85.6	88.2
LMVCNN_AlexNet	ImageNet1K	Random	12	12	90.2	91.7
LMVCNN_VggNet-11	ImageNet1K	Random	12	12	91.3	91.4
MVCNN* [27]	ImageNet1K	Random	20	20	87.9	88.1
RotationNet* [15]	ImageNet1K	Random	20	20	86.5	89.5
LMVCNN_AlexNet	ImageNet1K	Random	20	20	91.2	91.5
LMVCNN_VggNet-11	ImageNet1K	Random	20	20	**91.8**	**91.6**

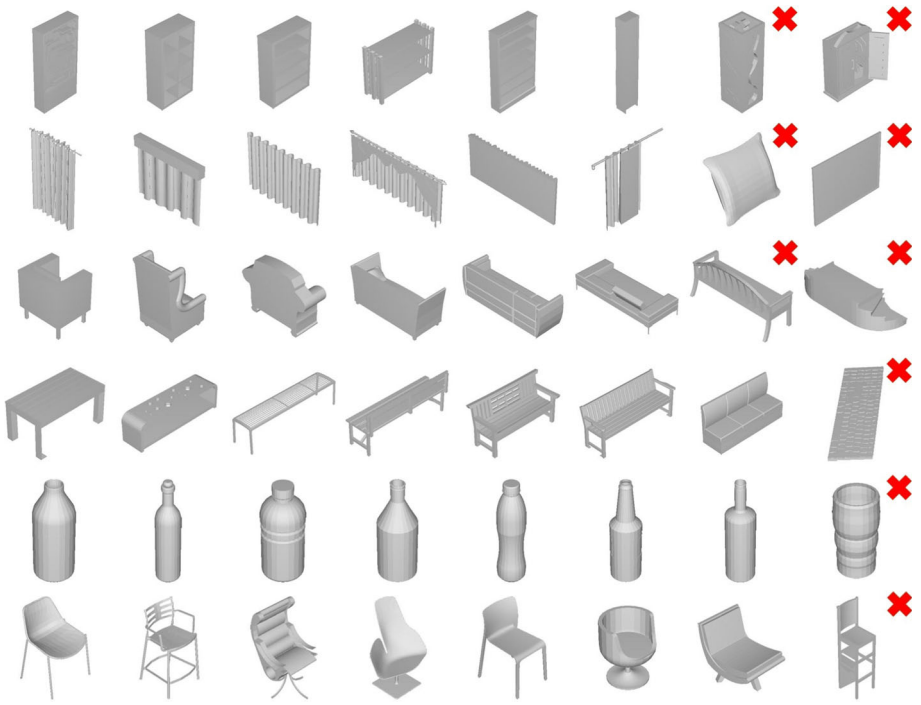
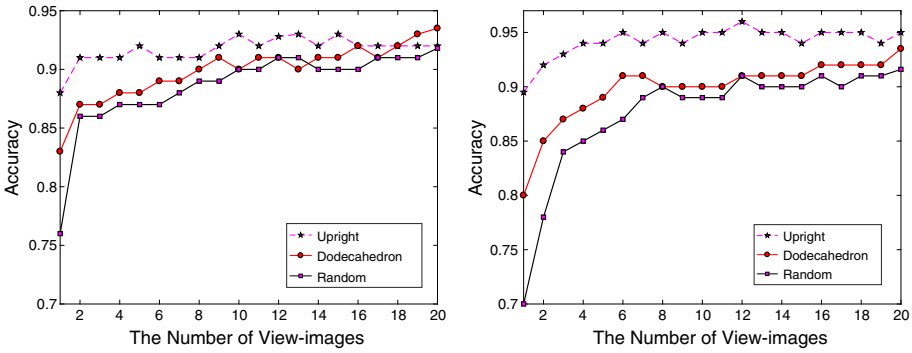


Fig. 5 Some classification results on the ModelNet40 and ModelNet10 datasets. The 3D shapes in the same row are classified into the same category. Note that the 3D shapes classified are marked with the red crosses in the upper-right corner of objects and still very similar to the 3D shapes in the predicted category. (Color figure online)

generated the training and testing datasets with the numbers of view-images from 1 to 20 on ModelNet40 and ModelNet10 respectively. Then the proposed LMVCNN was executed with these datasets in 10 trials for each number of view-images. As the final average classification accuracies shown in Fig. 6, LMVCNN even achieved 88.8% and 89.5% accuracies with only one view under the upright viewpoint setting on ModelNet40 and ModelNet10, respectively, while 3D ShapeNet [31] achieved a 77.3% accuracy by fully using the voxel representation of the 3D shapes. Moreover, all classification accuracies of LMVCNN with only one view under the dodecahedron and random viewpoints exceeded 70%. Note that all classification accuracies almost remained stable under the upright viewpoint setting, while other accuracies under other viewpoint settings increased with the number of views.

In the second case, we trained LMVCNN with the large number of the view-images from the pre-defined or the random viewpoints, and then tested our model with a small number of views. For each number of views, LMVCNN was performed in 10 times to obtain the average classification accuracy. As shown in Table 3, LMVCNN exhibits a promising performance when the number of the test view-images is quite small. Note that when there is only one test view-image, LMVCNN achieves a 93.3% accuracy on ModelNet10 by using 12 view-images from the upright viewpoint setting. Moreover, all of the classification accuracies of LMVCNN exceed 70% with the test view numbers 1-5.

Generally speaking, as shown in Fig. 6 and Table 3, LMVCNN shows promising classification performance when the number of view-images is small. The reason for this performance



(a) ModelNet40 **(b)** ModelNet10

Fig. 6 Classification accuracy vs. number of views used in training and testing on ModelNet40 and ModelNet10

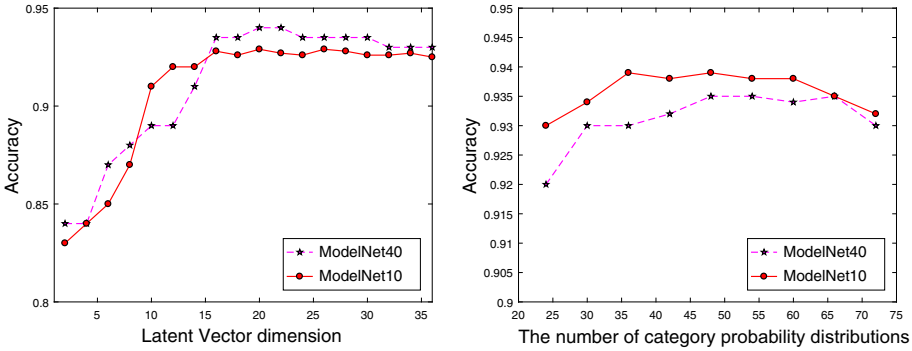
Table 3 Classification accuracies on ModelNet40 and ModelNet10 with **a small number of input test views**

Dataset	Training config.		Test #Views (Classification)				
	Rendering	#Views	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)
Modelnet40	Upright	12	88.0	90.0	91.1	90.5	90.6
	Dodecahedron	20	84.9	88.9	90.5	90.9	91.6
	Random	12	76.4	83.1	86.4	87.3	87.6
	Random	20	70.2	85.3	87.3	87.7	88.3
Modelnet10	Upright	12	93.3	93.5	94.5	94.9	94.1
	Dodecahedron	20	88.7	90.0	91.3	91.2	91.6
	Random	12	77.9	83.1	85.7	87.0	88.1
	Random	20	78.6	84.8	87.6	91.2	91.2

is that LMVCNN still attempts to choose a feasible combination of the category probability distributions to recognize 3D shape, rather than selects the category probability distributions independently for each input view-image.

4.1.2 Parameter Study

In this section, we investigated the influences of the Latent Vector dimension and the number of category probability distributions K to our experiment results. As shown in Fig. 7a, the classification accuracies on ModelNet40 increase as the Latent Vector dimension varies from 1 to 20, while the classification accuracies are almost stable when the Latent Vector dimension exceeds 20. As shown in Fig. 7b, the classification accuracies vary largely when the number of category probability distributions K is less than 36 and greater than 60, in addition, the classification accuracies can reach to the maximum and are stable when K is between 36 and 60. Therefore, it is feasible that the Latent Vector dimension is set as 16 and the number of category probability distributions K is set as 36.



(a) Classification accuracy vs. Latent Vector Dimension

(b) Classification accuracy vs. K

Fig. 7 Influences of the Latent Vector dimension and the number of category probability distributions K to the classification accuracies on ModelNet40 and ModelNet10

Table 4 Comparison of the classification accuracies from the **ablation study** on the ModelNet40 and ModelNet10 datasets

Dataset	Training config.		Ablation study		
	Rendering	#Views	**w** CNN3 (%)	**w/** CNN3 (%)	Baseline (%)
Modelnet40	Upright	12	92.8	90.0	84.8
	Dodecahedron	20	93.5	86.3	82.2
	Random	12	91.3	83.2	81.1
	Random	20	91.8	82.5	82.3
Modelnet10	Upright	12	96.6	94.5	90.5
	Dodecahedron	20	93.9	92.0	90.2
	Random	12	91.4	87.0	84.3
	Random	20	91.6	86.3	83.2

4.1.3 Ablation Study

Some ablation experiments were carried out to study the influences of the CNNS in LMVCNN. As shown in Table 4, LMVCNN without the third type of CNN (CNN3) performs slightly worse under the upright and dodecahedron viewpoint settings, and much worse under the random viewpoint setting, than LMVCNN with CNN3. These results indicate that CNN3 is able to improve the classification accuracies effectively especially for the random viewpoint settings. In the baseline method, the first type of CNN processed every input view-image to get a category probability distribution, then the sum of these distributions was calculated, and finally the classification scores were obtained by taking the log of the sum. The input object is assigned to the category with the highest classification score. The results in Table 4 show that LMVCNN is superior to the baseline in all of the viewpoint setting, and thus demonstrates the effectiveness of the proposed LMVCNN.

4.1.4 Influence of the CNN Architecture

We investigated the performance of LMVCNN with different CNN architectures: AlexnNet and VggNet(-11,-16,-19). As shown in Table 5, LMVCNN with VggNet-11 performs the best among these CNN architectures. Because VggNet can extract better features of the input view-images than AlexNet, LMVCNN with VggNet-11 naturally performs better than LMVCNN with AlexNet. If we make VggNet more complex, (such as VggNet-16 and VggNet-19), the classification accuracies of LMVCNN will significantly decrease in most of the rendering cases. In addition, we also executed LMVCNN with more advanced CNN architectures, such as ResNet-50. However, LMVCNN with these CNNs were not able to work properly. This may be because these CNNs are too complicated to be trained adequately with our method. Therefore, VggNet-11 is a feasible choice of the CNN architecture for our method.

4.2 3D Shape Retrieval

Some experiments were conducted to evaluate the model's ability for 3D shape retrieval. Note that the proposed neural network was not originally designed for 3D shape retrieval (it was designed for classification). Therefore, we employed the classification results to carry out the 3D shape retrieval. Specially, for each query, we built two retrieval sets: the first set contains all of the items with the same predicted category label with descending scores of the predicted category; the second set contains the rest of items with descending scores of the query's predicted category. We concatenated the second set after the first set to form a complete query set. In Table 6, our method was compared with other methods, including PANORAMA [20], 3D ShapeNets [31], DeepPano [23], MVCNN [27], GIFT [1], PANORAMA-NN [14], GVCNN [9] and NormalNet [29]. The MAPs of LMVCNN under all of the rendering settings are over 80% on ModelNet40, which are superior to the outcome for the classic method MVCNN. In addition, the MAP of LMVCNN reaches 86.7% under the dodecahedron viewpoint setting on ModelNet40, and is 1.0% higher than GVCNN [9]. Moreover, the MAP of LMVCNN reaches 91.8% under the upright viewpoint setting on ModelNet10, higher than all of the competitive methods.

For the retrieval task, we also plotted the PR curves of the other methods and the PR curves of LMVCNN with the 86.7% MAP on ModelNet40 and with 91.8% MAP on ModelNet10. As shown in Fig. 8, LMVCNN is superior to other methods on ModelNet40, while LMVCNN is slightly superior to GIFT and performs better than other methods on ModelNet10.

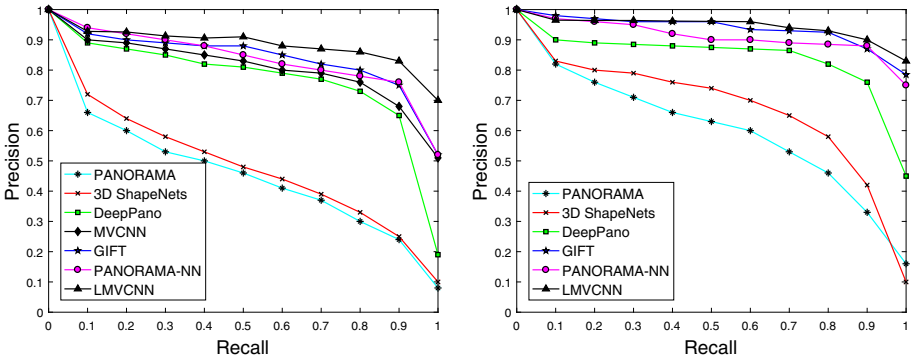
Finally, we investigated the influences of the number of views on the 3D shape retrieval. First, the proposed LMVCNN was performed in 10 trials for each number of view-images. As the final average MAPs shown in Fig. 9, LMVCNN even achieves 85.0% and 79.2% MAPs with only one view under the upright viewpoint setting on ModelNet40 and ModelNet10, respectively. All of the MAPs almost remained stable under the upright viewpoint setting, while the MAPs under other viewpoint settings increases with the number of views. Second, we trained LMVCNN with a large number of view-images from the pre-defined or the random viewpoints, and then retrieved the 3D shapes with a small number of views. As shown in Table 7, LMVCNN even works well for 3D shape retrieval when the number of test view-images is small. Note that when the number of the test view-image is 1, LMVCNN achieves 88.6% MAP on ModelNet10 by using 12 view-images from the upright viewpoint setting. Generally speaking, as shown in Fig. 9 and Table 7, LNVCNN exhibits a promising retrieval performance when the number of the view-images is small.

Table 5 Comparison of the classification accuracies on the ModelNet40 and ModelNet10 datasets with **different CNN architectures**

Dataset	Training config.		CNN architectures			
	Rendering	#Views	AlexNet (%)	VggNet-11 (%)	VggNet-16 (%)	VggNet-19 (%)
Modelnet40	Upright	12	91.8	92.8	**93.0**	91.2
	Dodecahedron	20	92.7	**93.5**	93.2	92.8
	Random	12	90.2	**91.3**	86.6	87.8
	Random	20	91.2	**91.8**	88.2	86.3
Modelnet10	Upright	12	95.2	**96.6**	95.2	93.2
	Dodecahedron	20	92.7	**93.9**	92.8	92.1
	Random	12	**91.7**	91.4	89.5	88.6
	Random	20	91.5	**91.6**	90.2	87.5

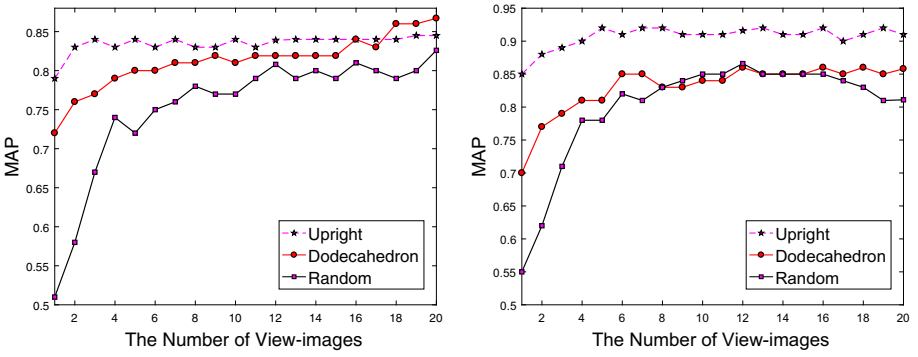
Table 6 Comparison of the retrieval MAPs on the ModelNet40 and ModelNet10 datasets

Method	ModelNet40 Retrieval MAP (%)	ModelNet10 Retrieval MAP (%)
PANORAMA [20]	46.1	60.3
3D ShapeNets [31]	49.2	68.3
MFF [33]	–	72.7
DeepPano [23]	76.8	84.2
MVCNN [27]	79.5	–
GIFT [1]	81.9	91.1
PANORAMA-NN [14]	83.5	87.4
GVCNN [9]	85.7	–
NormalNet [29]	84.4	–
LMVCNN (12 Random Views)	80.8	86.6
LMVCNN (20 Random Views)	82.6	81.4
LMVCNN (Upright)	83.9	**91.8**
LMVCNN (Dodecahedron)	**86.7**	85.8



(a) PR curves on ModelNet40 **(b)** PR curves on ModelNet10

Fig. 8 Precision-recall curves of different methods for the 3D shape retrieval on ModelNet40 and ModelNet10



(a) ModelNet40 **(b)** ModelNet10

Fig. 9 Retrieval MAPs vs. number of views used in training and testing on ModelNet40 and ModelNet10

Table 7 Retrieval MAPs on ModelNet40 and ModelNet10 with **a small number of input test views**

Dataset	Training config.		Test #Views (Retrieval)				
	Rendering	#Views	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)
Modelnet40	Upright	12	79.5	82.1	83.3	83.2	83.2
	Dodecahedron	20	75.9	80.7	82.3	83.1	84.2
	Random	12	66.3	73.7	77.8	78.7	79.5
	Random	20	67.4	76.2	79.0	80.1	80.2
Modelnet10	Upright	12	88.6	89.6	90.7	90.8	90.3
	Dodecahedron	20	82.3	83.8	84.3	84.2	84.5
	Random	12	66.9	72.1	75.1	76.7	78.1
	Random	20	70.2	75.9	79.4	79.2	79.3

4.3 Discussion of the Illustration to the Latent Vector

We interpreted the latent vector used in LMVCNN to choose a suitable category probability distribution. For each image, the second type of CNN outputs a latent vector. Some classification index centers are also introduced to correspond to the category probability distributions. If one latent vector is close to one classification index center, then the corresponding category probability distribution is selected. Regardless of whether we use the pre-defined or the random viewpoint, it is always possible for LMVCNN to choose a decent category probability distribution for an input view-image. In this way, the final classification scores of a 3D object are obtained by summing the log likelihoods of the category probability distributions selected by every view-image. The experimental results clearly demonstrate the reasonableness of this interpretation of the latent vector.

5 Conclusions

We proposed a novel problem to recognize 3D object using multiple view-images from the random viewpoints, which is much more challenging than using the pre-defined viewpoints. As shown in the experimental results, LMVCNN performs well and remains competitive with other related methods for the pre-defined and random viewpoints, and achieves a promising performance when the number of view-images is quite small. Based on these experiments, we conclude that the manner of the object rendering has an impact on 3D shape recognition.

In this paper, we have attempted to address the task of 3D shape recognition without background interference. Future studies will address how to detect and recognize 3D objects in a real 3D environment.

Acknowledgements We would like to thank the anonymous reviewers for their helpful suggestions. This work was supported by Natural Science Fund for Colleges and Universities in Jiangsu Province (Grant No. 18KJB520013), the Dual Creative Doctors of Jiangsu Province, National Nature Science Foundation of China (Grant Nos. 61902159 and 61771146), Zhejiang Provincial Natural Science Foundation of China (LQ19F020003) and Qing Lan Project of Jiangsu Province.

References

1. Bai S, Bai X, Zhou Z, Zhang Z, Latecki LJ (2016) Gift: a real-time and scalable 3d shape search engine. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 5023–5032
2. Bronstein MM, Bruna J, Lecun Y, Szlam A, Vandergheynst P (2017) Geometric deep learning: going beyond euclidean data. IEEE Signal Process Mag 34(4):18–42
3. Bruna J, Zaremba W, Szlam A, Lecun Y (2014) Spectral networks and locally connected networks on graphs. In: international conference on learning representations
4. Bu S, Wang L, Han P, Liu Z, Lib K (2017) 3d shape recognition and retrieval based on multi-modality deep learning. Neurocomputing 259:183–193
5. Charles RQ, Su H, Mo K, Guibas LJ (2016) Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 77–85
6. Chatfield K, Simonyan K, Vedaldi A, Zisserman A (2014) Return of the devil in the details: delving deep into convolutional nets. In: British Machine Vision Conference
7. Chen X, Chen Y, Gupta K, Zhou J, Najjaran H (2018) Slicenet: a proficient model for real-time 3d shape-based recognition. Neurocomputing 316:144–155
8. Cohen TS, Geiger M, Koehler J, Welling M (2018) Spherical cnns. In: Proceedings of international conference on learning representations
9. Feng Y, Zhang Z, Zhao X, Ji R, Gao Y (2018) Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In: Proceedings of IEEE international conference on computer vision, pp 264–272
10. Ghodrati H, Luciano L, Hamza AB (2019) Convolutional shape-aware representation for 3d object classification. Neural Process Lett 49(2):797–817
11. Hamza AB (2016) A graph-theoretic approach to 3d shape classification. Neurocomputing 211:11–21
12. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 770–778
13. He X, Yang Z, Zhou Z, Song B, Xiang B (2018) Triplet-center loss for multi-view 3d object retrieval. In: 2018 IEEE/CVF conference on computer vision and pattern recognition (CVPR)
14. Sfikas K, Pratikakis TTI (2017) Exploiting the panorama representation for convolutional neural network classification and retrieval. In: 3DOR2017
15. Kanezaki A, Matsushita Y, Nishida Y (2018) Rotationnet: joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: Proceedings of IEEE international conference on computer vision
16. Klokov R, Lempitsky V (2017) Escape from cells: deep kd-networks for the recognition of 3d point cloud models. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 863–872
17. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: International conference on neural information processing systems, pp 1097–1105
18. Monti F, Boscaini D, Masci J, Rodolà E, Svoboda J, Bronstein MM (2017) Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 5425–5434
19. Nie W, Liu A, Hao Y, Su Y (2018) View-based 3d model retrieval via multi-graph matching. Neural Process Lett 48(3):1395–1404
20. Papadakis P, Pratikakis I, Theoharis T, Perantonis S (2010) Panorama: a 3d shape descriptor based on panoramic views for unsupervised 3d object retrieval. Int J Comput Vision 89(2–3):177–192
21. Qi CR, Su H, Nießner M, Dai A, Yan M, Guibas LJ (2016) Volumetric and multi-view cnns for object classification on 3d data. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 5648–5656
22. Qi CR, Yi L, Su H, Guibas LJ (2017) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Proceedings of neural information processing systems
23. Shi B, Song B, Zhou Z, Xiang B (2015) Deeppano: deep panoramic representation for 3-d shape recognition. IEEE Signal Process Lett 22(12):2339–2343
24. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Proceedings of international conference on learning representations
25. Sinha A, Bai J, Ramani K (2016) Deep learning 3d shape surfaces using geometry images. In: proceedings of European conference on computer vision, pp 223–240,
26. Su H, Jampani V, Sun D, Maji S, Kalogerakis E, Yang MH, Kautz J (2018) Splatnet: sparse lattice networks for point cloud processing. In: Proceedings of IEEE international conference on computer vision
27. Su H, Maji S, Kalogerakis E (2015) Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of IEEE international conference on computer vision, pp 945–953

28. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of IEEE conference on computer vision and pattern recognition
29. Wang C, Cheng M, Sohelb F, Bennamounc M, Li J (2019) Normalnet: a voxel-based cnn for 3d object classification and retrieval. Neurocomputing 323:139–147
30. Wu J, Zhang C, Xue T, Freeman WT, Tenenbaum JB (2016) Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Proceedings of neural information processing systems
31. Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J (2015) 3d shapenets: a deep representation for volumetric shapes. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 1912–1920
32. Xie J, Zheng Z, Gao R, Wang W, Zhu SC, Wu YN (2018) Learning descriptor networks for 3d shape synthesis and analysis. In: Proceedings of IEEE conference on computer vision and pattern recognition
33. Yan Z, Zeng F (2017) 2d compressive sensing and multi-feature fusion for effective 3d shape retrieval. Inf Sci 409–410:101–120
34. Yi L, Su H, Guo X, Guibas L (2017) Syncspeccnn: synchronized spectral cnn for 3d shape segmentation. In: Proceedings of IEEE international conference on computer vision, pp 6584–6592
35. Yu T, Meng J, Yuan J (2018) Multi-view harmonized bilinear network for 3d object recognition. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 186–194
36. Zhou Y, Zeng F, Qian J, Han X (2019) 3d shape classification and retrieval based on polar view. Inf Sci 474:205–220