Cognitive Computation (2018) 10:908–921
https://doi.org/10.1007/s12559-018-9598-1

Multi-View CNN Feature Aggregation with ELM Auto-Encoder for 3D Shape Recognition

Zhi-Xin Yang[1] · Lulu Tang[1] · Kun Zhang[1] · Pak Kin Wong[1]

Received: 15 November 2017 / Accepted: 20 September 2018 / Published online: 10 October 2018

Abstract

Fast and accurate detection of 3D shapes is a fundamental task of robotic systems for intelligent tracking and automatic control. View-based 3D shape recognition has attracted increasing attention because human perceptions of 3D objects mainly rely on multiple 2D observations from different viewpoints. However, most existing multi-view-based cognitive computation methods use straightforward pairwise comparisons among the projected images then follow with weak aggregation mechanism, which results in heavy computation cost and low recognition accuracy. To address such problems, a novel network structure combining multi-view convolutional neural networks (M-CNNs), extreme learning machine auto-encoder (ELM-AE), and ELM classifer, named as MCEA, is proposed for comprehensive feature learning, effective feature aggregation, and efficient classification of 3D shapes. Such novel framework exploits the advantages of deep CNN architecture with the robust ELM-AE feature representation, as well as the fast ELM classifier for 3D model recognition. Compared with the existing set-to-set image comparison methods, the proposed shape-to-shape matching strategy could convert each high informative 3D model into a single compact feature descriptor via cognitive computation. Moreover, the proposed method runs much faster and obtains a good balance between classification accuracy and computational efficiency. Experimental results on the benchmarking Princeton ModelNet, ShapeNet Core 55, and PSB datasets show that the proposed framework achieves higher classification and retrieval accuracy in much shorter time than the state-of-the-art methods.

Keywords ELM auto-encoder · Convolutional neural networks · 3D shape recognition · Multi-view feature aggregation

Introduction

Fast detection of 3D shapes is a fundamental task in many fields, including computer vision, pattern recognition, and robotic systems. Such task is also of great relevance in modern industry for intelligent part transportation, manufacturing, and 3D printing. For 3D shape recognition, one can directly train recognition algorithms from original 3D representations, such as point clouds, voxel binary occupancies, or surface curvatures. Similar to human visual perception mechanism of 3D objects by means of 2D observations, the projective view-based method has been a basic tool in 3D shape recognition domain. View-based feature describes a 3D model by how it looks with the selected 2D projections. The visual similarity between the selected views of two models is regarded as the index of model difference. Compared to 3D model representations, view-based 2D representations have several desirable properties for shape recognition: (1) being less sensitive to 3D model representation artifacts, such as slightly imperfect polygon meshes and noisy surfaces, and (2) not relying on the explicit virtual model which may not be readily available in physical object detection scenarios. In addition, the rendered 2D views can be directly used for comparison with other images, silhouettes, or even hand-drawn sketches [1], thus leading to relatively low-dimensional and efficient computation. Furthermore, the well-developed image processing techniques could leverage the view-based methods by using the advances in image descriptors and massive image databases (such as ImageNet

✉ Lulu Tang
yb57421@connect.umac.mo

Zhi-Xin Yang
zxyang@umac.mo

Kun Zhang
mb65526@connect.umac.mo

Pak Kin Wong
fstpkw@umac.mo

1 Department of Electromechanical Engineering, Faculty of Science and Technology, University of Macau, Macau SAR, China

[2]) along with the powerful pre-trained deep convolutional neural network (CNN) architectures [3]. Thus, in this work, we adopt the view-based representation for 3D shape recognition.

The straightforward way of view-based 3D shape recognition is to recognize the projected views independently (as shown in Fig. 2), and this method has been well verified as in [4]. However, classifying views independently is a highly time-consuming and complex task owing to the pairwise comparisons between images of a 3D model. For a real-time retrieval system, retrieval accuracy and speed are both important factors that need to be simultaneously considered. However, the traditional 3D shape recognition methods mainly focus on recognition accuracy, with less attention on retrieval efficiency. To alleviate this issue, it is thus natural to synthesize all the information from multiple views of each model into a single feature descriptor. However, the question is how to synthesize all the information from multiple views to generate a compact and high-level representation of a 3D shape. This is a challenging task in multi-view representation. The naive solution, which simply concatenates all selected views into a single input to the CNN architecture, may cause an intractable training effort owing to the potentially infinite large input scale. More recent solution is to concatenate the individual outputs of the CNN into an extremely high-dimensional feature descriptor, which also suffers from complex computation and information redundancy. Thus, efficient feature extraction and dimension reduction techniques are needed to form an aggregated feature extractor for 3D shape recognition.

Traditional dimension reduction algorithms, such as the principal component analysis (PCA), auto-encoder (AE) [5], random projection (RP) [6], and non-negative matrix factorization (NMF) [7], are often used to reduce noise and irrelevant information in source data. However, the traditional AE and NMF are maimed by long training time. The PCA is not able to represent data as parts (e.g., a leg in a chair image), and RP only represents a subspace of the original data. Motivated by the recent work in [8], in this study, we introduce the extreme learning machine auto-encoder (ELM-AE) to 3D shape analysis for multi-view feature aggregation. The ELM-AE can efficiently learn the main features of the input multi-view data and reduce noise or redundant information, obtaining a low-dimensional and high-level feature representation.

Therefore, we propose a new multi-view learning framework (MCEA) by combining deep CNNs with ELM-AE for feature extraction and aggregation, and an ELM classifier for 3D model recognition. The proposed framework exploits the advantage of the deep CNN network and the robust ELM-AE feature representation, which could represent a 3D model as a single compact feature descriptor. By utilizing the advantages of ELM random assignment of input weights without fine-tuning, the ELM-AE algorithm can greatly decrease the computational cost of training. Thus, the framework of MCEA for 3D shape recognition can obtain a good balance between recognition accuracy and computational efficiency. More importantly, it can greatly improve the retrieval efficiency owing to the compact and low-dimensional model feature descriptor.

To summarize, the key technical contributions of this work are as follows:

1. A new hybrid framework of multi-view CNNs and ELM-AE (MCEA) is proposed for feature learning, classification, and retrieval of 3D models. To the best of our knowledge, this framework is the first to combine the advantages of deep CNN architecture with the robust ELM-AE feature representation, along with the fast ELM classifier, for 3D model recognition.
2. In contrast to the traditional multi-view methods that generate 3D-shape feature vectors with a simple concatenation procedure, the proposed MCEA aggregates shape features using the ELM-AE, thus alleviating information loss and information redundancy.
3. The multi-view CNNs compensate the shortcomings of the ELM-AE in direct feature learning from 3D object data. The combination of the deep CNNs architecture with the shallow ELM architecture is ideal for 3D object recognition.
4. The proposed method runs much faster than the existing set-to-set image matching methods and obtains a good balance between classification accuracy and computational efficiency.

The rest of this paper is organized as follows: we discuss the related work in “Related Work” and present the MCAE shape feature-learning algorithm in “Methodology”. The experiments involved in the proposed method are then represented in “Experiments and Evaluation”, followed by the analysis and discussion of the results. Finally, we conclude in “Conclusions”.

Related Work

The proposed method is related to prior work on 3D shape recognition, which has been extensively investigated for a couple of years. The works in this paper are also related to the application of ELM, which is a relatively recent intelligence technique that has been applied in various scenarios with different problems for regression and classifications. Below, we briefly review the relevant works.

3D Shape Representation Various algorithms have been proposed for 3D shape representation, including model- and view-based methods. Model-based methods describe a 3D model with native 3D shape representation, such as point clouds [9, 10], and volumetric[11, 12] or polygon meshes [13]. VoxNet [14], for example, creates object class detectors for 3D point-cloud data by integrating a volumetric occupancy grid representation with a supervised 3D CNN. 3D ShapeNets [15] describes a geometric 3D model as a probabilistic distribution of binary variables on a 3D voxel grid with a convolutional deep belief network. J. Wu et al. [11] later utilized a 3D generative-adversarial network for shape classification, which generates 3D objects from a probabilistic space by leveraging the recent advances in generative-adversarial nets and volumetric convolutional networks. More recently, M. Tatarchenko[12] proposed the octree generating networks, which learn to predict both the structure of the octree and the occupancy values of individual voxel cells and represents volumetric 3D outputs by using an octree. However, model-based methods are often sensitive to 3D model representation artifacts, like noisy surfaces or slightly imperfect polygon meshes.

View-based recognition methods describe a 3D model by a collection of 2D projections in order to exploit the advances in image processing techniques. Early works on view-based representation mainly focused on "handcrafted" descriptors, such as the Fisher vectors (FV) [16], which used SIFT features for representing human sketches of shapes. Based on local Gabor filters, Eitz et al. [17] compared human sketches with line drawings of 3D models produced from several different views. The light field descriptor (LFD) [18] extracts a set of geometric and Fourier descriptors from object silhouettes generated from different viewpoints. Other examples, such as the LFD [18], elevation descriptor (ED) [19], and SPH [20], are all representative of the work with "handcrafted" features. The existing view-based methods are labor intensive and it is difficult to extract discriminative information from the input 3D data. In this work, instead, we propose to learn the shape features from 3D models using automatic machine learning algorithms.

More recently, deep-learning-based methods have attracted increasing attention in many areas. Specifically, CNNs have recently been shown to be remarkably successful in image classification [3]. When CNNs are extended to the domain of view-based 3D shape recognition, it is shown that deep CNN-based descriptors have superior performance compared to the handcrafted view-based descriptors and many other model-based descriptors. Taking the 3D shape search engine GIFT [21] as an example, it consists of the following four components: projection rendering, view feature extraction, multi-view matching, and re-ranking. GIFT uses a pre-trained CNN for projected feature extraction and utilizes an inverted file for re-ranking. A multi-view CNN network (MVCNN) was later proposed to learn a 3D shape representation that aggregates information from multiple views and output a compact shape feature vector using the element-wise maximum operation across the selected views [4].

Extreme Learning Machine and Its Variants Compared to the traditional machine learning algorithms, ELM is a relatively recent intelligence technique that has been applied in various scenarios with different problems, such as regression and classifications with very promising results, both in terms of computational performance and accuracy [22, 23]. Different from other neural networks with well-known back-propagation (BP), all the hidden neurons in ELM are initialized randomly (independently from the input data) according to a certain continuous probability distribution and then fixed without iterative fine-tuning. The parameters that only need to be learned are the weights between the hidden layer and the output layer, resulting in a linear-in-the-parameter model, leading to significantly higher computational efficiency compared to the traditional BP neural networks.

During the last several years, the theories and applications of ELM have been extensively studied, such as the kernel ELM, incremental ELM (I-ELM), Bayesian ELM, adaboot ELM [24], multi-layer ELM-LRF [25], hybrid ELMs [26], etc. Especially, the ELM auto-encoder (ELM-AE) [27] has been proposed to perform unsupervised learning, with which the input data could be projected to a different dimensional space. For instance, Kasun et al. [8] proposed a dimension reduction framework with ELM-AE, which can represent data as parts with high learning speed. A complete proof of the principle of ELM-AE was presented, which could reduce the dimensions with the least effect on the Euclidean distance between data points and results in essentially the lowest variance of the dimensions. Owing to the desirable property of ELM-AE in dimension reduction, in this work, we exploit ELM-AE for our 3D shape-learning framework for multi-view feature aggregation.

Methodology

The proposed MCEA multi-view network architecture is composed of four modules: shape rendering, multi-view CNNs, ELM-AE-based feature aggregation, and ELM classifier, as depicted in Fig. 1. The view-based shape representation task starts from multiple views of a 3D model, which is rendered with different virtual cameras. A unified CNN, which is pre-trained as shown in Fig. 2, is applied to generate an image feature for each view separately. *N* number of CNNs are subsequently generated to represent the corresponding multiple image features

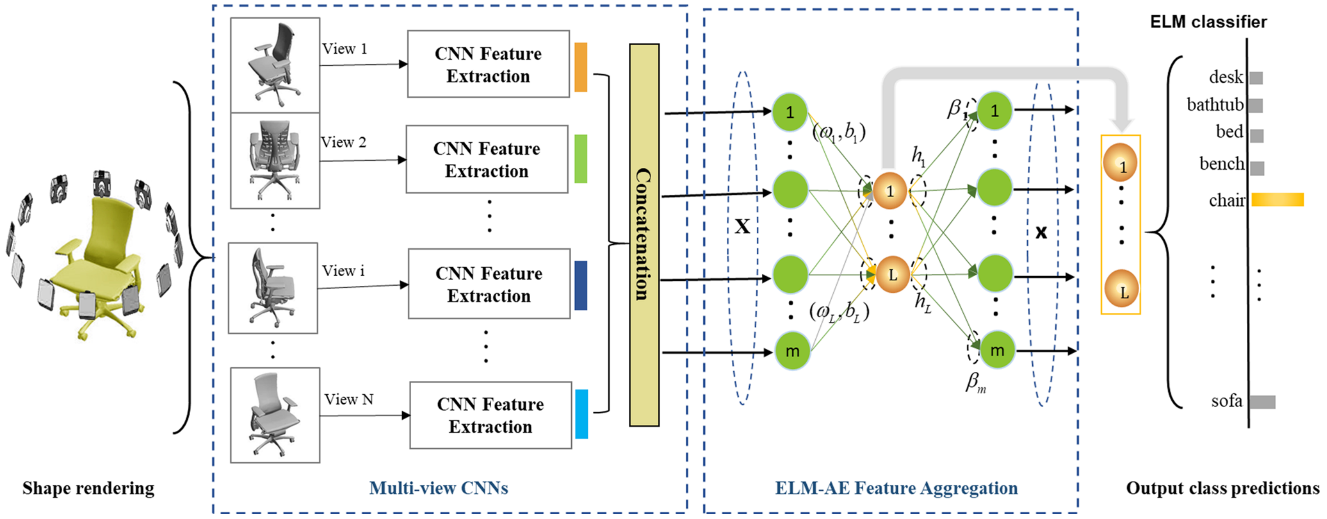


Fig. 1 The proposed multi-view feature-learning pipeline (MCEA). First, a 3D model is rendered from N different viewpoints, generating N images, which are then passed through the multi-view CNNs to extract the individual view-based features. These feature vectors are then concatenated and fed into the ELM-AE module to obtain a compact and low-dimensional shape feature for each model. Finally, an ELM classifier is utilized to predict the output class

(feature vectors) per 3D model, as shown in Fig. 1. For effective recognition, all of the N image features from a 3D model are concatenated as one shape vector X, which is then fed into the ELM-AE for feature aggregation and transformed into a single and compact shape feature. Finally, an ELM classifier is trained on those aggregated shape features. The average category accuracy is used to evaluate the recognition performance. Compared to pairwise comparisons between images of 3D models, the proposed aggregated shape feature can be directly used to compare 3D models, leading to significantly higher computational efficiency. Moreover, the aggregate shape features are more informative and meaningful than the simple combination of multiple projection image features.

For convenience, some important mathematic notations that are employed in this paper are listed in Table 1.

Input: Projection Rendering

To study the view-based 3D shape representation, we first need to generate multiple views of polygon meshes. Following the reflection model introduced in [28], each mesh polygon is rendered under a perspective projection. To create multiple views, virtual cameras (viewpoints) need to be appropriately set up to capture the 2D projections of each mesh. In order to compare the proposed method against the MVCNN in [4], we follow the same experimental setting in that work. Supposing that the input models are

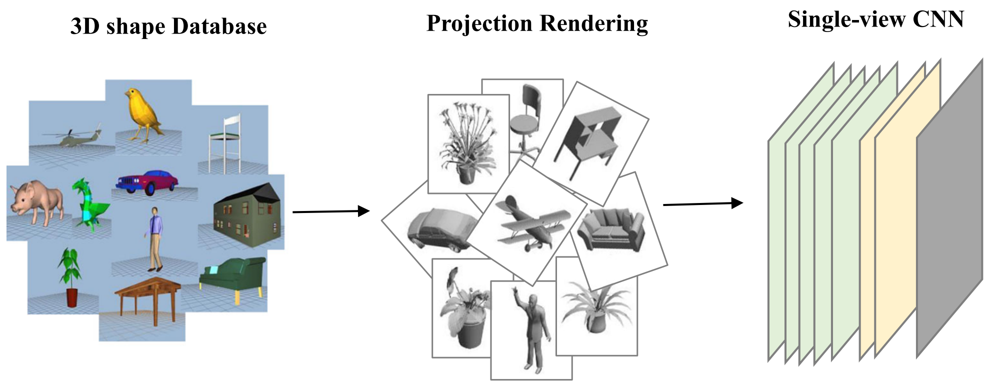


Fig. 2 Single-view feature-learning pipeline. A 3D model is first represented using a 2D projection image via projection rendering. The set of 2D images is then put into a single-view CNN architecture for feature learning. Each image is set with one label depending on its corresponding 3D shape

Table 1 List of some important notations

Notation	Description	Notation	Description
X	Input data	**T**	Target labels corresponding to X
$\mathbf{N_v}$	Number of projection images per model	**d**	Dimensionality of training data in ELM-AE
F(.)	CNN feature extractor for a given shape	**L**	Number of hidden nodes in ELM-AE
P(.)	Image feature learned by CNN	ω	Orthogonal random weights in ELM-AE
H	Hidden nodes	**b**	Orthogonal random biases in ELM-AE
g(.)	Activation function	β_{AE}	Output weights of ELM-AE
N	Number of training data	$\mathbf{Y_L(X)}$	Compressed features for X learned by ELM-AE

upright oriented along a consistent axis (e.g., the Z-axis), 12 different projections can be captured by placing 12 virtual cameras around the mesh, every 30° (as shown in Fig. 1). All the cameras are rotated 30° from the ground level and pointed to the centroid of the mesh, which yields 12 views per 3D model. Although additional views may improve the performance of the feature representation, in this study, we focus on the effect of applying the feature aggregation process. Therefore, for a fair and reasonable comparison, 12 views are eventually selected in this implementation "Experiments and Evaluation".

Multi-View CNN Feature Extraction

Deep-learning-based methods have been widely used as feature extraction techniques [29]. Thanks to the powerful deep CNN, networks, such as VGG, GoogLeNet, and ResNet [30], have been well trained on the ImageNet dataset [2]. Considering similar properties between the images in the ImageNet and the rendered views from Princeton ModelNet, in this study, we initialize our single-view CNN architecture as shown in Fig. 2, with pre-trained weights from the VGG-M network. The single-view CNN is then fine-tuned on all the projection images from the 3D shape dataset. This pre-trained single-view CNN architecture is eventually utilized as the feature extractor $F(.)$ of our multi-view learning framework (shown in Fig. 1).

Specifically, a typical CNN (ConvNet) architecture is stacked with a sequence of layers, and every layer of ConvNet transforms one volume of activations to another through a differentiable function. There are four important ideas in ConvNet: shared weights, local connections, pooling, and the use of multiple layers [29]. A normal CNN architecture mainly includes four types of layers: convolutional, Relu, pool, and fully connected layers (described in Fig. 3). The convolutional layer is the "brain" block in a convolutional network, which does most of the computational effort. A convolutional layer computes the output of neuron units that are connected to the local patches in its input feature space through a set of weights, which is called a filter bank. All the units in a feature space share the same filter bank owing to the high correlation and invariance of the local values in the images, which could dramatically reduce the number of weight parameters. Specifically,

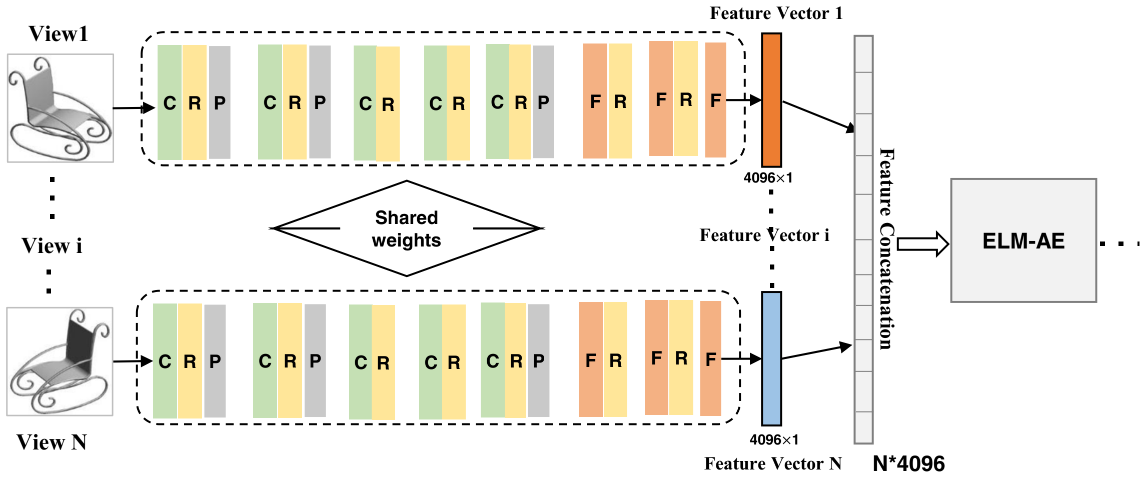


Fig. 3 Multi-view CNN architecture, where C means convolutional layer, R and P are the Relu layer and pool layer separately, and FC denotes fully connected layer. The filter sizes of the five convolutional layers are 7*7*3, 5*5*96, 3*3*256, 3*3*512, and 3*3*512, successively

each convolutional layer performs a dot product between their weights and a small region, which functions like a convolution operation with a certain filter and its input.

$$a = \sigma(f[x,y] * g[x,y]) = \sigma\left(\sum_{n1=-\infty}^{\infty}\sum_{n2=-\infty}^{\infty} f[n1,n2].g[x-n1,y-n2]\right) \quad (1)$$

where $f*g$ is an element-wise multiplication and the sum of a filter f and the input signal g. The result of this weighted sum is then fed to a non-linearity module such as a Relu. The Relu layer applies an element-wise activation function, such as the $max(0,x)$ thresholding at zero, to keep the volume size unchanged. The pool layer performs a down sampling operation along the spatial dimensions, using the maximum or average operations. This merges semantically those similar features into one feature. Pooling can progressively compress the spatial size of the input, and then reduce the large number of parameters and computation in the whole network. In addition, it also alleviates the overfitting, while in the fully connected layer, each neuron unit will be connected to all the activations in the previous volume. It is similar to convolutional layer, except that the neurons in the convolutional layer are connected only to a local patch in the input. Therefore, both layers use the convolution operation (dot product). In this way, a ConvNet architecture transforms the original input image layer-by-layer from the original discrete set of pixel values to the final class scores with certain differentiable functions.

In the single-view CNN architecture, as shown in Fig. 2, each view is set with one label and the network is fine-tuned on all selected views through a back-propagation criterion. Specifically, let $X = \{x_1, x_2, ..., x_N\}$ denote the training dataset with N shapes. For each shape $x_i \in X$ with one label l_i , through shape rendering, we can get its projective image set $P(x_i) = \{x_{i,1}, x_{i,2}, ...x_{i,N_v}\}$, where N_v is the number of projection images for each model. The labels of the projective images are the same as their corresponding 3D shape. Thus, the labeled projective images in the N_c-th category are

$$P_{N_c} = \{x_{i,j} | x_{i,j} \in P(x_i), x_i \in X, l_i = N_c\} \quad (2)$$

The single-view CNN architecture in this paper consists of five convolutional layers, three fully connected layers, and a SoftMax layer. The single-view CNN model is fine-tuned with all the projective images.

In the proposed multi-view CNNs, each view is passed through the unified CNN architecture (the pre-trained single-view CNN in Fig. 2) with the same parameters (weights and biases) separately. The penultimate (N_l-th) layer is selected as image feature, which generates a 4096-dimensional feature for each view. The N_v view features are then concatenated into a single long vector, resulting in a $4096 * N_v$ dimension feature for each 3D model. Specifically, for each shape $x_i \in X$, which is rendered with N_v projective images, $P(x_i) = \{x_{i,1}, x_{i,2}, ...x_{i,N_v}\}$. By feeding each projective image $x_{i,j} \in P(x_i)$ to the pre-trained CNN in the forward direction, we can obtain its activation with regard to the N_l-th layer of CNN as

$$P_j = F(x_{i,j}, N_l) \quad (3)$$

where P_j is the final feature representation for the j-th view and $F(.)$ denotes the CNN feature extractor. For a given shape $x_i \in X$, the shape feature can then be represented by

$$F(x_i) = \left[P_1, P_2, ...P_j...P_{N_v}\right] \quad (4)$$

ELM-AE-Based Feature Aggregation

Simple and straight forward feature concatenation of the multiple image descriptors, as shown in Eq. 4, leads to a high-dimensional feature representation while with less semantic interrelation. It is prone to inferior classification performance. Therefore, we utilize a novel feature aggregation technique which would synthesize all the feature representations of selected views (generated from the same model), $F(x_i) = [P_1, P_2, ...P_j...P_{N_v}]$, into a single and compressed 3D shape descriptor. Extreme learning machining-based auto-encoder [8, 31] is employed for feature transformation and aggregation. The ELM-AE-based feature representation is as shown in Fig. 4. The output function of classic ELM [22] is

$$\boldsymbol{f}_{\mathrm{L}}(\boldsymbol{x}) = \sum_{\mathrm{i}=1}^{\mathrm{L}} \beta_{\mathrm{i}} \mathrm{h}_{\mathrm{i}}(\boldsymbol{x}) = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta} \quad (5)$$

where L is number of hidden nodes, $\beta = [\beta_1,...,\beta_{\mathrm{L}}]^{\mathrm{T}}$ is the output weight matrix between the hidden layer and the output layer, and $\mathrm{h(x)} = [\mathrm{g}_1(\mathrm{x}),...,\mathrm{g}_{\mathrm{L}}(\mathrm{x})]$ are the random hidden features (hidden node output vector) for input X, where $\mathrm{g}_{\mathrm{i}}(\mathrm{x})$ is the output of the hidden node. Suppose, given N training samples $\{(\mathrm{x}_{\mathrm{i}},\mathrm{t}_{\mathrm{i}})\}_{\mathrm{i}=1}^{\mathrm{N}}$, then ELM learning process can be boiled down to solving the following problem:

$$\boldsymbol{H\beta} = \boldsymbol{T} \quad (6)$$

where $\mathrm{T} = [\mathrm{t}_1,\mathrm{t}_2,...,\mathrm{t}_{\mathrm{N}}]^{\mathrm{T}}$ are the target labels corresponding to the input data $\mathrm{X} = [\mathrm{x}_1,...,\mathrm{x}_{\mathrm{N}}]^{\mathrm{T}}$. The objective function is to minimize the weighted sum of the training error and the norm of output weights:

$$C||\boldsymbol{H\beta} - \boldsymbol{T}||_2^2 + ||\boldsymbol{\beta}||_2^2 \quad (7)$$

By solving the above optimization problem, the weight β can be obtained in two closed-form solutions corresponding to different scales of training samples. If the dimensionality

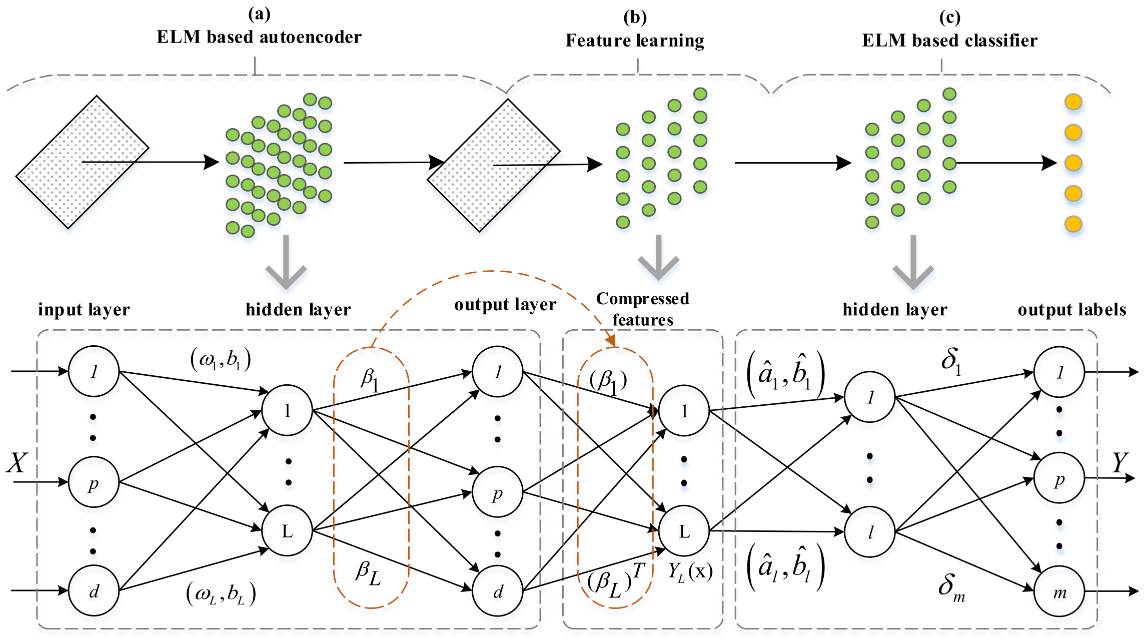


Fig. 4 Shape feature aggregation using the ELM based auto-encoder and classifier. **a** The target output of ELM-AE is the same as input **x**, and the hidden node parameters (ω_i, b_i) are made orthogonal after randomly generated; output weights β are then calculated using the regularized least squares. **b** The output compressed feature of the ELM-AE is given by $\mathbf{y}(\mathbf{x}) = \mathbf{x}\beta$. **c** The ELM classifier is eventually used for the final recognition task

of training samples is larger than the number of hidden nodes, i.e., $d > L$, the solution is:

– Large-scale training case ($d > L$)

$$\boldsymbol{\beta} = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^T\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\boldsymbol{T} \tag{8}$$

– Small-scale training case ($d < L$)

$$\boldsymbol{\beta} = \boldsymbol{H}^T\left(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{T} \tag{9}$$

ELM-AE has the same solution as the classic ELM except for (1) the target output of ELM-AE is the same as input X and (2) parameters (weights and biases) of the hidden nodes are made orthogonal after randomly generated. ELM-AE can transform the representation of the input data into three different coding architectures:

1. Compressed architecture: represent features from a higher dimensional input data space to a lower dimensional feature space
2. Equal dimension architecture: represent features from input data space to an equal dimensional feature space
3. Sparse architecture: represent features from a lower dimensional input data space to a higher dimensional feature space

In this work, we utilize the compressed ELM-AE architecture to learn the shape feature from multiple 2D image features. The random orthogonal weights of the hidden nodes project the input data to a different lower dimensional space, offering a low-dimensional shape descriptor. As shown in Fig. 4, ELM-AE can be calculated as

$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{\omega}.\boldsymbol{x} + \boldsymbol{b}) \tag{10}$$

where $\omega^T\omega = \mathrm{I}$, $\mathrm{b}^T\mathrm{b} = 1$, and $\omega = [\omega_1,...,\omega_L]$ are the orthogonal random weights, $\mathrm{b} = [\mathrm{b}_1,...,\mathrm{b}_L]$ are the orthogonal random biases between the input and hidden nodes and $\mathrm{X} = [\mathrm{x}_1,...,\mathrm{x}_N]^T$ are the input as well as the target output data. Then, the objective function of ELM-AE can be represented as

$$\min_{\beta_{AE}} : ||\boldsymbol{H}\boldsymbol{\beta}_{\boldsymbol{AE}} - \boldsymbol{X}||^2 \tag{11}$$

The output weights β_{AE} of ELM-AE can be then given by

$$\boldsymbol{\beta}_{\boldsymbol{AE}} = \left(\frac{\boldsymbol{I}}{C} + \boldsymbol{H}^{\boldsymbol{T}}\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\boldsymbol{X} \tag{12}$$

The hidden layer of an auto-encoder must preserve information of input data [32]; therefore, ELM-AE retains the Euclidean information and main features of input data through orthogonal random parameters. The output weights β_{AE} of ELM-AE are responsible for the transformation from the input data space to feature space. Then, feature aggregation and dimension reduction are eventually achieved by ELM-AE through projecting input X with weights β_{AE} as

$$Y_L(X) = X\beta_{AE}^T = X\left(\left(\frac{I}{C} + H^T H\right)^{-1} H^T X\right)^T \tag{13}$$

The training routine of the ELM-AE in this work is summarized as Algorithm 1:

Algorithm 1 Training routine of an ELM-AE (given a set of training samples X, activation function g, and hidden node number L.)

```
1: Randomly generate hidden neuron parameters based
   on any continuous distribution.
2: Parameters (weights and biases) are made orthogo-
   nal after randomly generated.
3: Calculate matrix H (Eq. 10).
4: Estimate the output weight $\beta_{AE}$ based on some
   optimization constraints Eq. 12.
```

Experiments and Evaluation

In this section, we demonstrate the performance of MCEA and compare it with state-of-the-art methods for 3D shape recognition on the Princeton ModelNet dataset [15], Princeton Shape Benchmark (PSB)[33], and ShapeNet Core 55 database [34]. The experiments are all performed on a Dell workstation with an Intel(R) Xeon(R) CPU (3.00 GHz), 64 GB RAM memory, and one NVIDIA GPU with GeForce GTX 1080.

Datasets

The Princeton ModelNet currently contains 127,915 CAD models in 662 categories. In the following experiments, we run the proposed algorithm on its two well-annotated subsets: ModelNet10 and ModelNet40, both of which are publicly available on the Princeton ModelNet website. Furthermore, we follow the same training and test split provided by the authors [15] of the dataset (Table 2).

The PSB database is also a public database with 907 polygonal models divided into 92 categories, which are collected from the World Wide Web. We also utilize the ShapeNet Core 55, subset of ShapeNet (ShapeNet55), which is collected from the Trimble 3D warehouse [35]. It contains approximately 51300 3D models from 55 common categories, and it is further divided into 204 sub-categories, which is quite challenging owing to the diversity of categories and large variations within classes. The whole dataset is further split into training/validation/test sets with 70%/10%/20% shares, respectively. Further, the dataset involves two variants (the ShapeNet55 normal dataset and ShapeNet55 perturbed dataset). In the normal setting, the shapes are all aligned, while in the perturbed version, each model is randomly rotated by a certain angle. We particularly conduct experiments on the perturbed dataset, which is more challenging.

Table 2 Statistics of Princeton ModelNet

Dataset	Models	Training	Test
ModelNet10	4899	3991	908
ModelNet40	12311	9843	2468

Experimental Setting

In our implementation, each 3D model is rendered by different virtual cameras, yielding 12 views. Each view is rescaled to 224*224*3 to fit the pre-trained VGG-M network. We use the penultimate layer as image features, generating a 4096-dimensional descriptor for each view. Then, we concatenate the 12 view descriptors into a single long vector, generating a 4096*12 dimension descriptor for each 3D model, which is then fed into ELM-AE for feature aggregation. A single layer ELM-AE is utilized in our framework and the number of selected hidden nodes is set at 5000. Meanwhile, the number of hidden nodes of the ELM classifier is set at 1500. Both are determined using a grid search.

3D Shape Retrieval

In this subsection, we consider two types of retrieval approaches. One is a set-to-set image pairwise comparison, which establishes a correspondence between two sets of image features. We apply the standard Hausdorff distance measurement. Consider a query shape x_q and a matching shape x_m from the dataset. Through shape rendering and multi-view CNNs for feature extraction, we can obtain two feature sets $F(x_q) = [q_1, q_2, ...q_{N_v}]$ and $F(x_m) = [m_1, m_2, ...m_{N_v}]$ respectively. Here, we consider the activation with regard to the second fully connected layer of CNNs as the final feature representation for each view, where N_v is the number of views for each model and q_i (or m_i) denotes the view feature with respect to the i-th view of model x_q (or x_m). Then, the standard Hausdorff distance between two sets of features is defined as

$$D(x_q, x_m) = \max_{q_i \in F(x_q)} \min_{m_j \in F(x_m)} d(q_i, m_j)) \tag{14}$$

Table 3 Analysis of time consumption

Datasets	Off-line	On-line (query) (image-to-image)	On-line (query) (model-to-model)
ModelNet10	≈3.6 h	5.78 s	45.12 ms
ModelNet40	≈6.8 h	14.15 s	100.25 ms
PSB	≈8.5 h	10.71 s	78.38 ms

where function $d(.)$ measures the distance between two features. Another approach is the proposed model-to-model matching strategy, which straightforwardly compares a query model with models from the database. In this strategy, the inputs are model features, so we can simply apply $d(x_Q, x_M)$ to measure the distance between the two models, where x_Q, x_M represent the aggregated model features through the ELM-AE.

As can be seen from Eq. 14, supposing that there are N shapes in the database, then the time complexity of standard Hausdorff matching for a set-to-set image search engine is $\mathbf{O}(N \times N_v^2)$ for a given query model. Model-to-model matching only requires time complexity of $\mathbf{O}(N)$ for each query. The time cost of retrieving for different databases is listed in Table 3.

From Table 3, we can clearly see that, model-to-model search uses much less time than set-to-set image matching, which is consistent with the theoretical analysis of time complexity. The off-line stage includes shape rendering, single-view CNN training, and feature extraction for all models in database, and CNN training is the most time-consuming operation. However, the query time on the ModelNet datasets and PSB database can be controlled within 1 s, which is much more efficient than that in the traditional image-to-image matching strategy. To quantify the retrieval performance and compare with other state-of-the-art methods, we employ the most commonly used evaluation metric, mean average precision (MAP), which represents the average precision with which a positive shape is returned.

Table 4 Performance comparison of retrieval on ModelNet10 and ModelNet40 dataset

Methods	ModelNet10 (mAP)	ModelNet40 (mAP)
(1) SPH [20]	44.05%	33.26%
(2) LFD [18]	49.82%	40.91%
(3) FV [16]	/	43.90%
(4) 3D ShapeNets [15]	68.26%	49.23%
(5) MVCNN [4]	/	70.10%
(6) GIFT [21]	91.12%	81.94%
(7) Single-view CNN	75.26 %	62.80%
(8) MCEA (ours)	91.54 %	82.06%

Table 5 Performance (%) comparison results of the involved methods on the ShapeNet55 perturbed dataset

	Methods	P@N	R@N	F@N	NDCG
1	GIFT [21]	54.6	58.1	54.2	86.6
2	MVCNN [4]	51.9	54.9	51.4	81.8
3	Li [34]	30.1	75.1	35.8	81.6
3	Wang [34]	33.7	26.5	20.5	73.6
3	Tatsuma [34]	24.9	64.0	29.0	79.1
4	MCAE (ours)	52.67	54.31	53.75	82.40

To compare the proposed method with state-of-the-art methods, we collect the retrieval results that are publicly available, which includes the spherical harmonic (SPH) [20], light field descriptor (LFD) [18], Fisher vector (FV) [16], 3D ShapeNets [15], MVCNN [4], and GIFT [21]. Table 4 clearly shows that the proposed MCEA method achieves a result comparable to those obtained by all the state-of-the-art descriptors and the single-view CNN baseline with set-to-set image matching.

We further evaluate the retrieval performance of MCEA on the ShapeNet55 perturbed database, which is extremely challenging for most existing methods. We utilize the evaluation code provided by the organizer of SHREC'16 [34], which calculates several accuracy metrics: Precision (P@N), Recall (R@N), F-score (F@N), and NDCG. We select the following methods from SHREC'16 [34] for the comparisons: GIFT [21], MVCNN [4], Li [34], Wang [34], and Tatsuma [34]. The results of the performance comparison of the involved methods on the ShapeNet55 perturbed dataset are presented in Table 5.

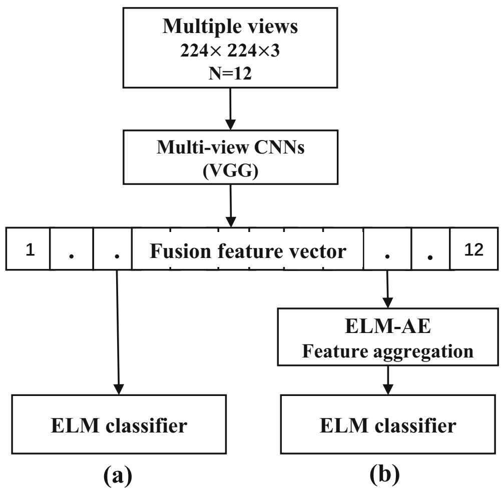


Fig. 5 Architecture of involved algorithms. (a) CNN-ELM. (b) MCEA

As listed in Table 5, the proposed MECA method can achieve superior results than most of the above methods and obtain marginal advantages over MVCNN [4]. It confirms that our MECA method has an advantage to explore the efficient information of 3D models. When compared with GIFT [21], our method shows less advantage in retrieval task. The reason is that GIFT includes an efficient re-ranking mechanism, particularly designed for retrieval tasks after feature learning. We believe that combined with a metric learning algorithm, such as the Mahalanobis metric, our method could obtain better retrieval results.

3D Shape Classification

To evaluate the efficiency of the proposed method in classification tasks, we conduct two different experiments (depicted in Fig. 5). The details are as follows: (a) Firstly, the pre-trained CNN (shown in Fig. 2) is utilized as the feature extractor of the projective images, and the output of the penultimate layer of each CNN is used as image feature. Twelve image features are then concatenated into a single fusion vector as the feature representation for each 3D model. Finally, an ELM classifier is applied on those fusion feature vectors. We call this method as CNN-ELM. (b) To verify the performance of the ELM-AE, in the second experiment, we introduce the ELM-AE to the process of shape recognition. As shown in Fig. 5b, an ELM-AE module is added between the fusion layer and the last ELM classifier based on the first experiment, which is denoted as MCEA.

The main results of the experiments are included in Table 6. With an ELM-AE module, the classification accuracy can be increased by approximately 2.2% in relation to the baseline model with CNN-ELM on ModelNet40. It indicates that the introduced ELM-AE module can obtain the principal components of the original input, as an auto-encoder does, and yield a more meaningful shape representation.

In order to compare the proposed method with the state-of-the-art methods for 3D shape classification on the ModelNet10 and ModelNet40, we collect the classification results publicly available on Princeton ModelNet website, which include 3D ShapeNets by Wu et al. [15], GIFT by Bai et al. [21], VoxNet by Daniel Maturana et al. [14], DeepPano by Shi et al. [36], and MVCNN by Su et al. [4]. We also choose three handcrafted methods: the spherical harmonics (SPH) by Kazhdan et al. [20], light field descriptor (LFD) by Chen et al. [18], as well as Fisher vector (FV) by Snchez et al. [16]. Finally, we select an ELM-based method named MVD-ELM by Xie et al. [37] for comparison. The classification results are summarized in Table 7.

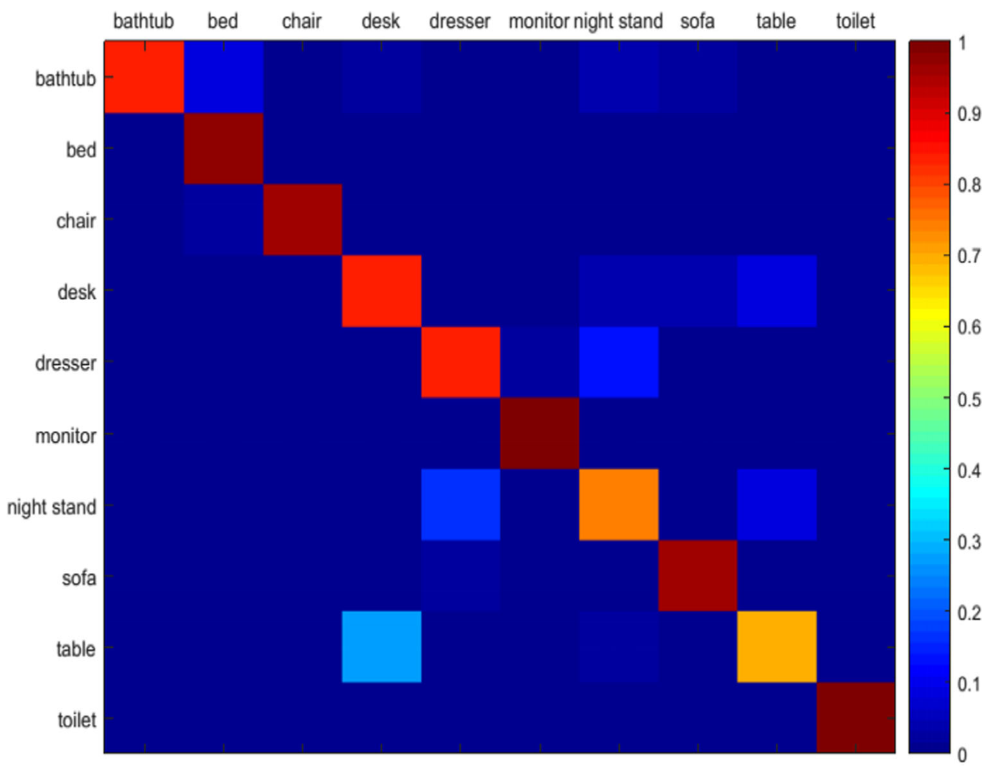
Table 7 lists the classification accuracies and training times for individual methods. It is observed that the recognition performance of MCEA is superior to three handcrafted methods, which is mainly due to its powerful machine learning algorithms. To further evaluate MCEA, we compare the proposed method against several deep learning methods, such as 3D ShapeNets, GIFT, VoxNet, and MVCNN. It is important to note that most of the deep learning methods need several hours or several days during the training stage. In contrast, our method could consume less than 700 s (except for the pretraining time of the single-view CNN) for training in ModelNet10 or ModelNet40, attaining comparable high classification accuracies. When we pay more attention to the feature aggregation process, then the feature extraction can be implemented by other descriptors. Even using the original VGG-M network as feature extractor without fine-tuning (Table 7. (11)), with ELM-AE we can obtain approximately 88.67% of classification accuracy on ModelNet40. It confirms the validity of MCEA for feature representation. Furthermore, when the fine-tuned single-view network is used as feature

Table 6 Statistics of classification accuracy and time on Princeton ModelNet

	Classification accuracy and time	
Network	ModelNet10	ModelNet40
CNN-ELM	90.97% (20.63s)	88.43% (60.14s)
MCEA	92.18% (298.00s)	90.65% (614.60s)

Table 7 Classification accuracies and training times for individual methods

	Classification accuracy		
Method	ModelNet10	ModelNet40	Time
(1) SPH [20] +SVM	79.97%	68.23%	–
(2) LFD [18] +SVM	79.87%	75.47%	–
(3) FV [16] +SVM	–	78.8%	–
(4) 3DShapeNet [15]	83.50%	77.30%	≈2days
(5) DeepPano [36]	85.45%	77.63%	≈4h
(6) VoxNet [14]	92.00%	83.00%	6∼12 h
(7) MVCNN [4]	–	90.10%	–
(8) GIFT [21]	91.50%	89.50%	0.3∼0.7h
(9) MVD-ELM [37]	88.99%	81.39%	≤700 s
(10) Single-view CNN	–	85.10%	–
(11) VGG+ELM-AE	90.46%	88.67%	≤700 s
(12) MCEA (ours)	92.18%	90.65%	≤700 s



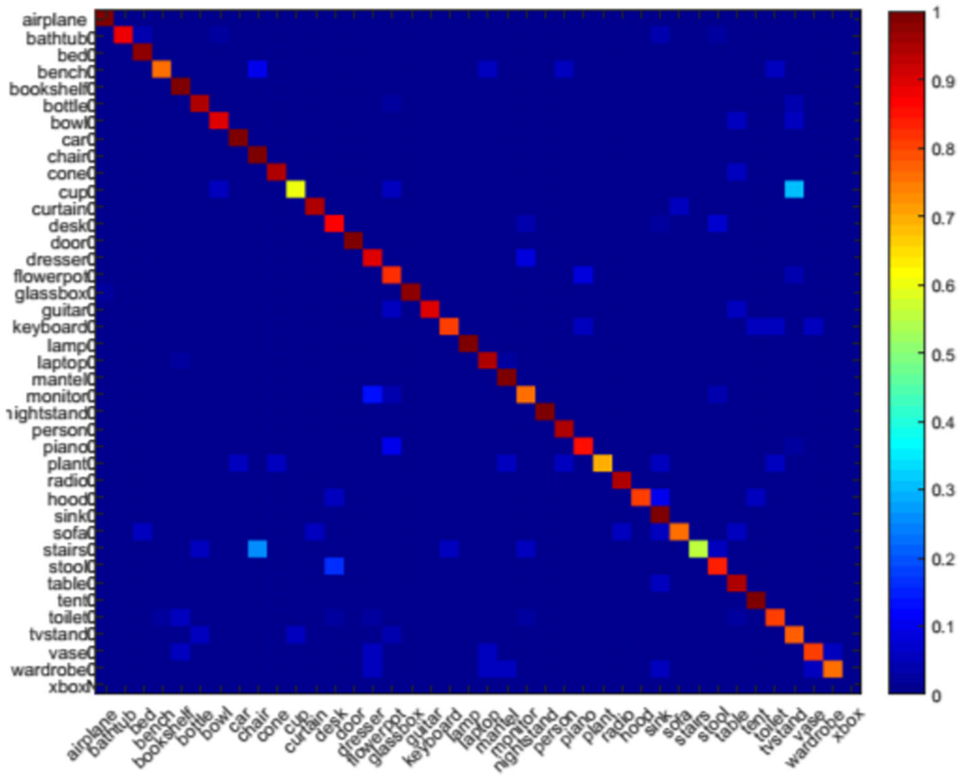


Fig. 6 Confusion matrices of MCEA on ModelNet10 (top) and ModelNet40 (bottom). Ground truth labels are on the vertical axis and predicted labels are on the horizontal axis

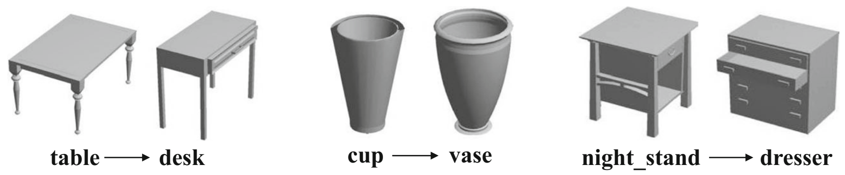


Fig. 7 Some misclassification instances of the proposed method

extractor, our MCEA method could achieve better results (90.65%). In contrast to MVD-ELM [37], which adopts the multi-view depth image representation and deep extreme learning machine to achieve fast feature learning for 3D shapes, our method obtains superior accuracy because of the powerful deep network and informative aggregated ELM-AE module.

Discussion

The confusion matrices of our final results on Princeton ModelNet10 and ModelNet40 are shown in Fig. 6. In the matrices, the diagonal elements describe the classification accuracy and the off-diagonal elements show the misclassification proportion. As shown in Fig. 6, most 3D models can be correctly classified except for some categories with similar appearance, such as table and desk. Some misclassification models are displayed in Fig. 7. Those errors are mainly due to the similar appearance between different categories.

We apply 2D t-SNE embedding [38] to the feature vectors learned from MCEA, and a different color (or number) encodes a different model category. The serial numbers from 1 to 10 in Fig. 8 represent the categories of bathtub, bed, chair, desk, dresser, monitor, night_stand, sofa, table, and toilet respectively. The result shows that 3D models can be well-separated with our MCEA method.

Figure 9 shows the classification accuracy curves obtained by the proposed MCEA method. It can be seen that the accuracy curves have tiny fluctuation as parameter L varies, and when $L = 1500$ there is a peak of recognition accuracy. Thus, in the proposed MCEA framework, the number of hidden nodes in the ELM classifier is set as 1500. We also explore the performance with a different number of hidden nodes in the ELM-AE, the results are clearly demonstrated in Fig. 10. It indicates that the ELM-AE is not sensitive to the number of hidden nodes as long as it is assigned as a large number (e.g., $L > 5000$). Additional hidden nodes could not benefit the improvement in performance but consume much more training time. Therefore, we set the number of hidden node of the ELM-AE as 5000, which means that the input multi-view features are eventually reduced to 5000-dimension features.

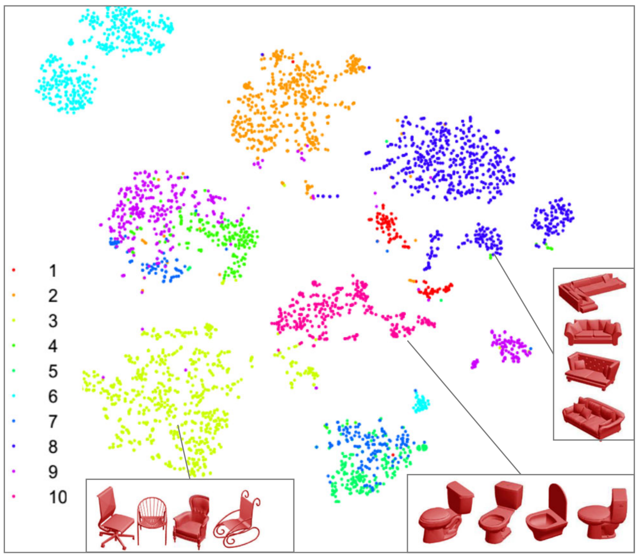


Fig. 8 t-SNE feature visualization on ModelNet10 with the proposed MCEA (best viewed in color)

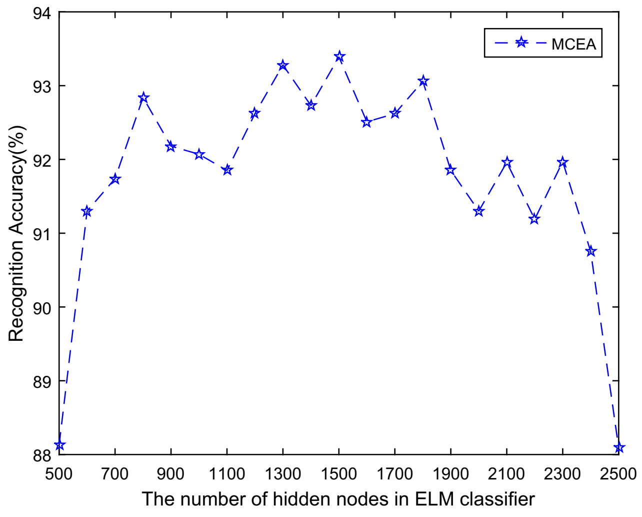


Fig. 9 Effect of hidden nodes' number L in ELM classifier

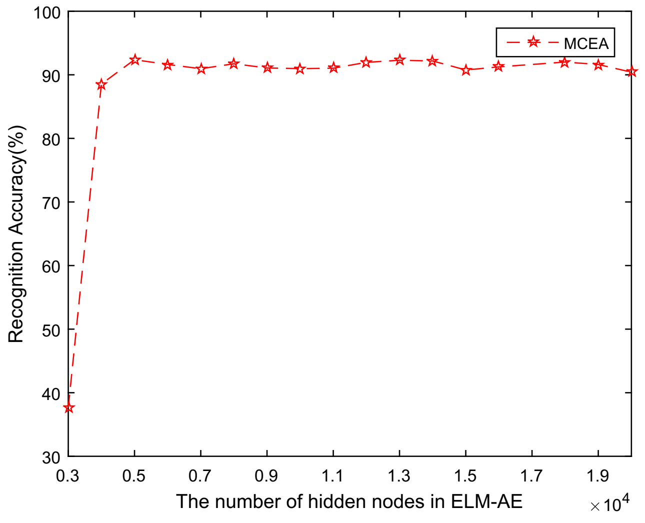


Fig. 10 The performance ELM-AE under different number of hidden nodes

Conclusions

In this work, we proposed a computationally efficient method for view-based 3D shape recognition. A new framework of multi-view CNNs and ELM-AE was developed to recognize 3D models, which utilizes the composited advantages of deep CNN architecture with the robust ELM-AE feature representation, as well as the fast ELM classifier. The experimental results demonstrate the effectiveness of MCEA in solving 3D shape recognition problems. In contrast to existing 3D shape recognition methods that face difficulties for fast shape retrieval or learning without information loss or redundancy, the proposed method runs much faster while maintaining a very high recognition accuracy. The novel MCEA framework successfully achieves a good balance between recognition accuracy and computational efficiency. Therefore, the proposed combination of deep CNN architecture with shallow ELM architecture is feasible for leveraging the performance of view-based 3D object recognition, especially when the 3D geometric features are not available in certain physical scenarios. In addition, the proposed framework could be readily extended to other multi-view representational cognitive areas, such as RGB-D object recognition, and face recognition.

Funding Information This study was supported in part by the Science and Technology Development Fund of Macao S.A.R (FDCT) under grant FDCT/121/2016/A3 and MoST-FDCT Joint Grant 015/2015/AMJ, in part by University of Macau under grant MYRG2016-00160-FST.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Wang F, Kang L, Li Y. Sketch-based 3D shape retrieval using convolutional neural networks, Computer Science. 2015.
2. Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L. Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009 IEEE, pp. 248–255; 2009.
3. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. Adv Neural Inf Proces Syst. 2012;25(2):2012.
4. Su H, Maji S, Kalogerakis E, et al. Multi-view convolutional neural networks for 3D shape recognition. In: IEEE International Conference on Computer Vision. IEEE Computer Society; 2015. p. 945–953.
5. Rifai S, Vincent P, Muller X, Glorot X, Bengio Y. Contractive auto-encoders: explicit invariance during feature extraction. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 833–840; 2011.
6. Achlioptas D. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. Journal of computer and System Sciences. 2003;66(4):671–687.
7. Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature. 1999;401(6755):788–791.
8. Kasun LLC, Yang Y, Huang G-B, Zhang Z. Dimension reduction with extreme learning machine. IEEE Trans Image Process. 2016;25(8):3906–3918.
9. Qi CR, Su H, Mo K, Guibas LJ. Pointnet: deep learning on point sets for 3D classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE. 2017;1(2):4.
10. Li Y, Bu R, Sun M, Chen B. Pointcnn. 2018. arXiv:1801.07791.
11. Wu J, Zhang C, Xue T, Freeman B, Tenenbaum J. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: Advances in Neural Information Processing Systems, pp. 82–90; 2016.
12. Tatarchenko M, Dosovitskiy A, Brox T. Octree generating networks: efficient convolutional architectures for high-resolution 3d outputs. 2017. arXiv:1703.09438.
13. Yi L, Kim VG, Ceylan D, Shen I, Yan M, Su H, Lu C, Huang Q, Sheffer A, Guibas L, et al. A scalable active framework for region annotation in 3D shape collections. ACM Trans Graph (TOG). 2016;35(6):210.
14. Maturana D, Scherer S. Voxnet: a 3D convolutional neural network for real-time object recognition. In: International Conference on Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ. IEEE, pp. 922–928; 2015.
15. Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, Xiao J. 3D shapenets: a deep representation for volumetric shapes, Eprint Arxiv, pp. 1912–1920. 2015.
16. Nchez J, Perronnin F, Mensink T, Verbeek J. Image classification with the fisher vector: theory and practice. Int J Comput Vis. 2013;105(3):222–245.
17. Eitz M, Richter R, Boubekeur T, Hildebrand K, Alexa M. Sketch-based shape retrieval. ACM Trans Graph. 2012;31(4):31–1.
18. Chen D-Y, Tian X-P, Shen Y-T, Ouhyoung M. On visual similarity based 3d model retrieval. In: Computer Graphics Forum, vol. 122, no. 3. Wiley Online Library, pp. 223–232; 2003.
19. Shih J-L, Lee C-H, Wang JT. A new 3D model retrieval approach based on the elevation descriptor. Pattern Recogn. 2007;40(1):283–295.

20. Kazhdan M, Funkhouser T, Rusinkiewicz S. Rotation invariant spherical harmonic representation of 3D shape descriptors. In: Eurographics/acm SIGGRAPH Symposium on Geometry Processing, pp. 156–164; 2003.
21. Bai S, Bai X, Zhou Z, Zhang Z, Latecki LJ. Gift: a real-time and scalable 3D shape search engine. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. IEEE, pp. 5023–5032; 2016.
22. Huang GB, Zhou H, Ding X, Zhang R. Extreme learning machine for regression and multiclass classification. IEEE Trans Syst Man Cybern Part B Cybern. 2012;42(2):513–529.
23. Huang GB, Cambria E, Toh KA, Widrow B, Xu Z. New trends of learning in computational intelligence. IEEE Comput Intell Mag. 2015;10(2):16–17.
24. Zhang PB, Yang ZX. A novel adaboost framework with robust threshold and structural optimization. IEEE Transactions on Cybernetics. 2018;48(1):64–76.
25. Huang G, Huang GB, Song S, You K. Trends in extreme learning machines: a review. Neural Netw. 2015;61:32–48.
26. Yang ZX, Wang XB, Wong PK. Single and simultaneous fault diagnosis with application to a multistage gearbox: a versatile dual-elm network approach. IEEE Trans Ind Inf. 2018;PP(99):1–1.
27. Zhou H, Huang GB, Lin Z, Wang H, Soh YC. Stacked extreme learning machines. IEEE Transactions on Cybernetics. 2015;45(9):2013–2025.
28. Phong BT. Illumination for computer generated pictures. Commun ACM. 1975;18(6):311–317.
29. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–444.
30. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M. Imagenet large scale visual recognition challenge. Int J Comput Vis. 2015;115(3):211–252.
31. Tang J, Deng C, Huang GB. Extreme learning machine for multilayer perceptron. IEEE Transactions on Neural Networks and Learning Systems. 2016;27(4):809–821.
32. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol P-A. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J Mach Learn Res. 2010;11(Dec):3371–3408.
33. Shilane P, Min P, Kazhdan M, Funkhouser T. The princeton shape benchmark. In: Shape modeling applications, 2004. Proceedings. IEEE, pp. 167–178; 2004.
34. Savva M, Yu F, Su H, Kanezaki A, Furuya T, Ohbuchi R, Zhou Z, Yu R, Bai S, Bai X, Aono M, Tatsuma A, Thermos S, Axenopoulos A, Papadopoulos GT, Daras P, Deng X, Lian Z, Li B, Johan H, Lu Y, Mk S. Large-scale 3D shape retrieval from shapenet Core55. Eurographics Workshop on 3D Object Retrieval. In: Pratikakis I, Dupont F, and Ovsjanikov M, editors; 2017. The Eurographics Association.
35. Trimble. 3D warehouse. 2012. https://3dwarehouse.sketchup.com/.
36. Shi B, Bai S, Zhou Z, Bai X. Deeppano: Deep panoramic representation for 3-D shape recognition. IEEE Signal Process Lett. 2015;22(12):2339–2343.
37. Xie Z, Xu K, Shan W, Liu L, Xiong Y, Huang H. Projective feature learning for 3D shapes with multi-view depth images. Comput Graphics Forum. 2015;7:34.
38. Maaten Lvd, Hinton G. Visualizing data using t-sne. J Mach Learn Res. 2008;9(Nov):2579–2605.