Knowledge-Based Systems 247 (2022) 108754

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Multi-view 3D object retrieval leveraging the aggregation of view and instance attentive features

Dongyun Lin *, Yiqun Li, Yi Cheng, Shitala Prasad, Tin Lay Nwe, Sheng Dong, Aiyuan Guo

*Institute for Infocomm Research, A*STAR, Singapore, 1 Fusionopolis Way, #21-01 Connexis (South Tower), Singapore 138632, Singapore*

ARTICLE INFO

Article history:
Received 1 December 2021
Received in revised form 3 April 2022
Accepted 5 April 2022
Available online 11 April 2022

Keywords:
View-based 3D object retrieval
View attention module
Instance attention module
ArcFace loss
Cosine distance triplet-center loss

ABSTRACT

In multi-view 3D object retrieval tasks, it is pivotal to aggregate visual features extracted from multiple view images to generate a discriminative representation for a 3D object. The existing multi-view convolutional neural network employs view pooling for feature aggregation, which ignores the local view-relevant discriminative information within each view image and the global correlative information across all view images. To leverage both types of information, we propose two self-attention modules, namely, **V**iew **A**ttention **M**odule and **I**nstance **A**ttention **M**odule, to learn view and instance attentive features, respectively. The final representation of a 3D object is the aggregation of three features: original, view-attentive, and instance-attentive. Furthermore, we propose employing the ArcFace loss together with the cosine-distance-based triplet-center loss as the metric learning guidance to train our model. As the cosine distance is used to rank the retrieval results, our angular metric learning losses achieve a consistent objective between the training and testing processes, thereby facilitating discriminative feature learning. Extensive experiments and ablation studies are conducted on four publicly available datasets on 3D object retrieval to show the superiority of the proposed method over multiple state-of-the-art methods.

1. Introduction

With the rapid development of computer-vision-based multimedia technologies, large amounts of 3D data are being used to support various applications, such as 3D printing, autonomous driving, augmented reality, and industrial product design and inspection. 3D object retrieval remains a critical and challenging task for the effective management of such large amounts of data. Therefore, numerous studies have been conducted on this topic [1–7]. In general, 3D object retrieval refers to the task of obtaining a ranked list of 3D objects that are semantically similar to a given query object. To address this task, it is pivotal to generate discriminative representations for 3D objects. In addition to deep neural networks for learning discriminative representations for content-based 2D image retrieval [8–10], various deep-learning methods have been proposed to handle 3D object retrieval. These methods can be grouped into two categories: model-based and view-based methods. Model-based methods produce deep representations of 3D objects by converting them into 3D models based on voxels or point clouds [2–4]. View-based methods first capture multiple view images of a 3D object from different viewpoints and then generate feature representations based on 2D view images [1,5–7]. Compared with model-based methods, view-based methods are more flexible and computationally efficient. Owing to the development of deep-learning-based image processing and analysis, view-based methods can produce better retrieval performance than model-based methods. Hence, in this study, we focus on improving view-based 3D object retrieval.

Among the view-based methods for 3D object retrieval, the multi-view convolutional neural network (MVCNN) and its variants [5,6,11,12] are well-known. The general idea of the MVCNN paradigm is to extract visual features from each view image using a weight-shared convolutional neural network (CNN) and then generate the aggregated visual feature by applying view pooling. Subsequently, the aggregated feature is fed through another CNN to output a vectorized feature as the final representation of the 3D object. During testing, the vectorized feature output from the second CNN is used for retrieval. However, in this paradigm, the view pooling aggregation of all visual features ignores two key factors: (i) the local view-relevant discriminative information within each view image and (ii) the global correlative information across all view images as they jointly describe the object. The former is view-level local information, and the latter is instance-level global information. These two levels of information should be jointly investigated to improve the retrieval performance. To leverage these two types of information, we propose two spatial self-attention modules, namely, **V**iew **A**ttention **M**odule (VAM) and **I**nstance **A**ttention **M**odule (IAM), to learn view- and instance-attentive features, respectively. Fig. 1 shows

* Corresponding author.
E-mail address: lind@i2r.a-star.edu.sg (D. Lin).

https://doi.org/10.1016/j.knosys.2022.108754
0950-7051/

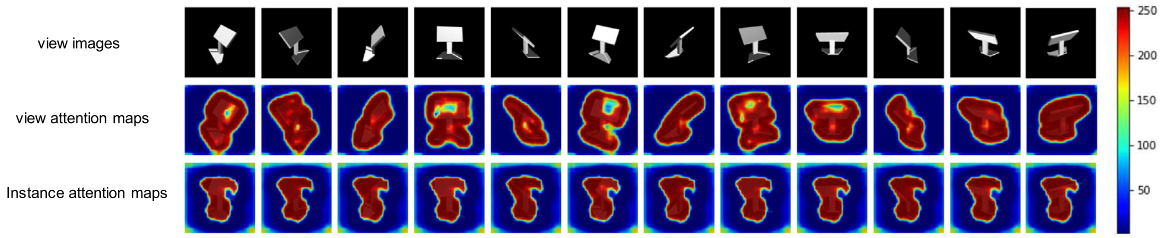

Fig. 1. View and instance attention maps. The first row shows rendered view images of a monitor; the second row shows the view attention maps, which are outputs of the proposed VAM; the bottom row shows instance attention maps, which are outputs of the proposed IAM. The color bar indicates the values of the attention maps.

the example view and instance attention maps generated by the proposed VAM and IAM, respectively. The figure reveals that the learned view attention maps (shown in the second row of Fig. 1) highlight the regions that reflect the view-specific information, for example, the object pose. By contrast, the instance attention maps (shown in the third row) remain robust across various view images and highlight object-specific regions. For each view image, we obtain three vectorized features: (i) the original visual feature generated by CNN1, (ii) the view-attentive features generated by CNN1 and the VAM, and (iii) the instance-attentive features generated by CNN1 and the IAM. Then, we aggregate these three features to generate the final representation of the 3D object.

Motivated by the success of learning discriminative feature embeddings of 2D images in the angular feature space [11,13,14] and metric learning techniques in image retrieval [6,12], we propose to employ ArcFace loss together with the cosine-distance-based triplet-center loss (TCL) as the metric learning guidance to learn discriminative representations of 3D objects in the angular feature space. Since the cosine distance is used to rank the retrieval results during testing, our angular metric learning losses achieve a consistent objective between the training and testing processes, thereby facilitating discriminative feature learning.

The major contributions of this study are summarized as follows:

- To leverage local view-relevant discriminative information within each view image, we propose a View Attention Module (VAM) to learn view-attentive features for each view image.
- To leverage global correlative information across all the view images, we propose an Instance Attention Module (IAM) to learn instance-attentive features for each view image.
- To achieve a consistent objective between the training and testing processes, we propose to employ the ArcFace loss together with cosine-distance-based TCL as the metric learning guidance to learn discriminative representations in the angular feature space.
- Extensive experiments and analysis conducted on four public datasets show the superiority of the proposed method over multiple state-of-the-art methods in 3D object retrieval.

2. Related work

In this section, recent methods for 3D object retrieval are classified into two categories, namely, model-based and view-based methods, and briefly reviewed.

2.1. Model-based methods

Model-based methods learn the representations of 3D objects based on their raw 3D format, such as voxels and point clouds. Voxel-based methods apply 3D CNNs to learn the representations of 3D objects in the format of voxelized dense and regular grids. For example, 3D ShapeNet [2] utilizes a deep belief network to learn the feature representations of 3D voxelized grids. VoxNet [3] exploits a 3D CNN to learn the representations of voxelized grids. In [15], a voxel-based variational autoencoder and Voxception-ResNet (VRN) were proposed for 3D object reconstruction and recognition. In [16], a novel method was proposed for the classification of 3D shapes by exploiting surface and volumetric clues inside a deep learning framework. In general, voxel-based methods directly handle voxelized representations that are capable of exploiting the complete information of 3D objects but suffer from high computational and memory complexity. Point-cloud-based methods learn representations from a set of 3D points sampled from 3D objects. PointNet [17] was first proposed to exploit 3D deep neural networks to learn representations of point clouds of 3D objects. PointNet++ [18] improved PointNet by leveraging multiscale features in a hierarchical manner. Furthermore, KD-Network [4] was proposed to apply the KD-tree structure to learn the hierarchical representation of 3D objects. 3DCapsule [19] was proposed as a 3D extension of capsule networks to function as a drop-in replacement module of the fully connected (FC) layer in PointNet for classifying 3D point clouds. In [20], G3DNet was proposed as a general-purpose feature extractor based on graph convolution networks for 3D point clouds. Exploiting 3D features with a standard classifier can produce good accuracy for 3D point cloud classification. Point-cloud-based methods generally reduce the computational burden compared with voxel-based methods, but their performance is affected by the unstructured nature of point sets. View-based methods represent a 3D object by using a set of 2D view images captured from various viewpoints around the object. Compared with model-based methods, view-based methods are more flexible and computationally efficient. Owing to the advancements in image processing and analysis, view-based methods can produce better retrieval performance than model-based methods. Hence, in this study, we focused on view-based methods.

2.2. View-based methods

View-based methods render multiple view images by projecting a 3D object from various perspectives and learning feature representations by aggregating visual features extracted from multiple view images. In some early works, hand-crafted visual features (e.g., 2D Zernike moments, 2D Fourier descriptors, and 2D Krawtchouk moments) extracted from each view image were exploited and 3D object retrieval was handled as a group matching problem [21,22]. In [21], Zhao et al. proposed a feature fusion method using multimodal graph learning for view-based 3D object retrieval. In [22], the view importance was evaluated at the discriminative level, and the discriminative weights of views were calculated by measuring their shortest distance with views from inter categories. The discriminative weights of views can be integrated into distance-based retrieval and graph-based retrieval, as proposed in [21]. Compared with these early works, our method adopts 2D CNN features that are learnable during training and therefore they are more discriminative than handcrafted features. Second, in [22], the view importance was analyzed by employing an offline classification process that is not directly related to the training of the retrieval model. In contrast, we propose to exploit a spatial self-attention mechanism that can be trained in an end-to-end manner using a CNN. Third, our method leverages not only view-relevant discriminative information but also global correlative information across all views to further improve retrieval performance.

Owing to the recent development of 2D CNNs in feature extraction and learning [23–25], view-based methods using CNN features have produced state-of-the-art performance in 3D object retrieval. Su et al. proposed an MVCNN [5] for 3D object retrieval. The MVCNN first extracts visual features from each view image using a weight-shared CNN and then aggregates these visual features by applying view pooling to obtain the final representation feature. Numerous studies have been conducted on the modification of the MVCNN architecture from two perspectives: view feature aggregation approaches and loss functions. In [26], Wang et al. analyzed the differences between classification and retrieval tasks for 3D objects and concluded that a compact intraclass distribution descriptor in the embedding space is pivotal for improving retrieval performance. They proposed a cube loss that can mine hard positive and hard negative samples and achieve superior performance compared with other triplet losses. He et al. introduced a TCL [12] to replace the softmax loss in the MVCNN to learn more discriminative feature representations. In [6], an improved MVCNN was proposed by employing group-view similarity learning and a novel TCL with adaptive margins. In [27], view images were divided into a set of visual n-grams, and then visual features were extracted based on grams, which produced representations of 3D objects that were robust to rotation. The MLVCNN [28] introduced a hierarchical view-loop-shaped architecture to obtain 3D shape representations from different scales. Wei [29] proposed a view-based graph CNN for 3D shape analysis. The SPNet [30] exploits stereographic transformation to project 3D objects into a 2D plane and then applies a 2D CNN to estimate the object category followed by a view ensemble, which combines the responses from multiple views of the object to further improve the predictions. In [31], inductive multi-hypergraph learning was proposed to address view-based 3D object classification.

Most of the aforementioned MVCNN variants leverage only view-relevant discriminative information for 3D object retrieval. Our method further exploits the instance-attentive features for each view image to improve retrieval performance. A recently proposed network named deep-attention network (DAN) [32] also exploits the concept of self-attention to address 3D object recognition. However, the idea and methodology of the DAN are completely different from those of our method. First, our method proposes two parallel self-attention modules to jointly learn the view- and instance-attentive features. In contrast, the DAN exploits only view-level information without considering instance-level information. Second, our method and the DAN exploit different levels of CNN features. Specifically, our method designs spatial attention maps to handle the intermediate feature maps of CNNs, whereas the DAN utilizes a deep attention layer that stacks several self-attention architectures to handle the global feature vectors generated by pretrained CNN models. Third, our method further employs the ArcFace loss together with the cosine-distance-based TCL as the metric learning guidance to learn discriminative representations in the angular feature space. The DAN employs only the cross-entropy loss for 3D object classification.

In other studies, view images were treated as a sequence, and recurrent neural networks (RNNs) were applied to aggregate the visual features from multiple view images [7,33–36]. For example, SeqViews2SeqLabels [36] consists of an encoder-RNN followed by a decoder-RNN to learn global features by aggregating sequential view images and then performing shape classification and retrieval from the learned global features. The assumption of RNN-based methods is that various view images exhibit temporal correlation. Hence, these methods are sensitive to the sequence of view features fed through the RNNs. Our method exploits the view correlation in a different manner by fusing the view-level and instance-level features via the proposed instance attention module. As the instance-level feature is calculated using all view-level features, our method is robust to the sequence of view images fed through the network.

3. Proposed method

In this section, we first review the MVCNN paradigm for multi-view 3D object retrieval and then analyze its major limitations. Next, we present details of the proposed method.

3.1. MVCNN paradigm

In this subsection, we review the MVCNN paradigm [5] for multi-view 3D object retrieval. First, multiple view images of a 3D object are generated by a rendering engine, e.g., a group of cameras. Subsequently, these view images are fed through CNN1 to obtain convolutional feature maps associated with different views. An aggregated convolutional feature map is then generated by feeding these view-relevant feature maps through a view pooling layer. Finally, the aggregated convolution feature map is vectorized using CNN2. The entire network is trained by optimizing the cross-entropy loss in an end-to-end manner. The high-dimensional feature vector before the last FC layer of CNN2 is used as the descriptor for object retrieval. Regarding CNN1 and CNN2, the widely adopted MVCNN architecture [5,6,11] typically inserts a view pooling layer into a backbone network (e.g., VGG) at an intermediate layer. The subnetwork before the view pooling layer is treated as CNN1, and the one after the view pooling layer is treated as CNN2. In general, the selection of the backbone network is flexible for any CNN, such as AlexNet [23], VGG [24] and ResNet [25]. Although simple and effective, the original MVCNN paradigm has two major limitations: (i) it overlooks the view-relevant discriminative information within each view image and (ii) it does not exploit the correlative information across all view images.

3.2. Overview of the proposed method

Fig. 2 illustrates the overall architecture of the proposed method. During training, it consists of four stages: (i) rendering multiple view images by projecting the 3D object models from varying perspectives; (ii) extracting visual features for each view image using a CNN (CNN1); (iii) generating view- and instance-attentive features using VAM and IAM, respectively; (iv) vectorizing the feature maps using another CNN (CNN2) and simultaneously classifying the three types of features, namely, the original CNN, view-attentive and instance-attentive features, by optimizing the proposed metric learning loss. The details of each stage are specified in the subsequent sections.

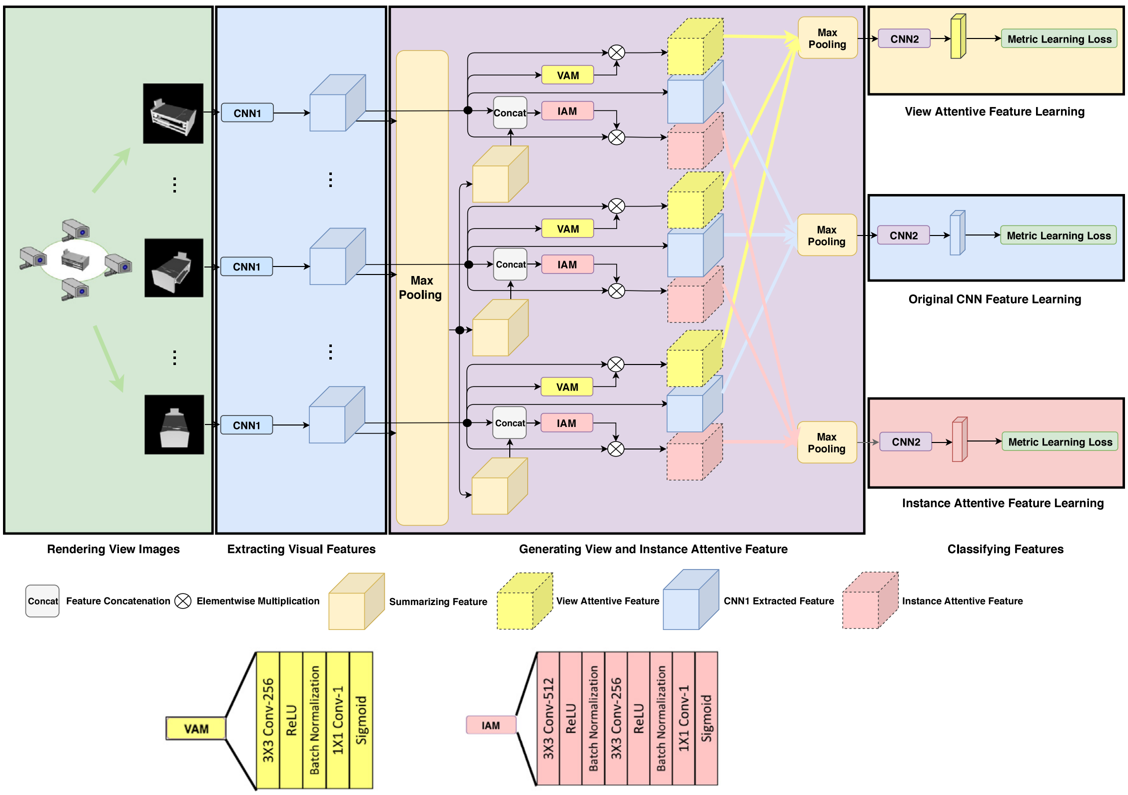

Fig. 2. Overall architecture for VAM and IAM. Notation C×C Conv-N denotes a C-by-C convolution layer with N output channels. ReLU and sigmoid denote ReLU and sigmoid activation functions, respectively.

3.3. View image rendering and feature extraction

Based on the paradigm of view-based 3D object retrieval [5,6,11], given a 3D object, a set of view images are captured by setting virtual cameras from multiple viewpoints. In this study, the Phong shading method based on Blender is adopted to render the view images. The view images are generated by virtual cameras which are set to uniformly surround the object at a viewpoint angle interval of 30 degrees along the Z-axis [5]. Given a 3D object, $\mathcal{O}_i$, multiple view images denoted as $\{\mathcal{I}_i^1, \mathcal{I}_i^2, \ldots, \mathcal{I}_i^M\}$ are generated, where M denotes the number of views. These view images are then fed through CNN1 to extract the original visual features:

$$\mathcal{F}_{\text{ori},i}^j = \text{CNN1}(\mathcal{I}_i^j),\ j = 1, 2, \ldots, M. \tag{1}$$

3.4. View- and instance-attentive feature learning

As discussed in the related work, the MVCNN and its modified variants [5,6,11] directly adopt view pooling to aggregate multiple visual features from different views into a single feature. Such aggregation overlooks two key factors: (i) the local view-relevant discriminative information within each view image and (ii) the global correlative information across all view images, as they jointly describe the 3D object. To leverage these two types of information, we propose the VAM and IAM, to learn the view- and instance-attentive features, respectively.

3.4.1. VAM

To extract view-relevant discriminative information, we propose the VAM. The VAM consists of two convolution stages: a 3×3 convolution layer with 256 output channels, followed by a ReLU activation function and a 2D batch normalization operation; and a 1×1 convolution layer with one output channel, followed by a sigmoid activation function to generate a spatial attention map with its entries within [0, 1] to highlight the attentive region. To generate the view-relevant attentive features, the extracted features from CNN1 are pixel-wise multiplied with the attention map for each channel, as illustrated in Fig. 3. Formally, given the j_{th} visual feature, $\mathcal{F}_{\text{ori},i}^j$, for object $\mathcal{O}_i$, the view-attentive feature is calculated as

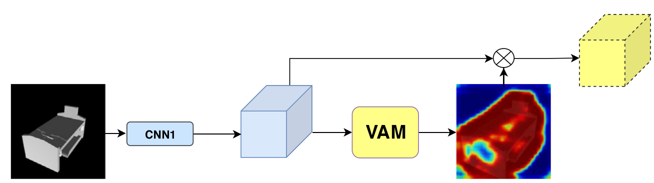

Fig. 3. Generation of view-attentive features.

$$\mathcal{F}_{\text{view},i}^j = \mathcal{F}_{\text{ori},i}^j \otimes \text{VAM}(\mathcal{F}_{\text{ori},i}^j), \tag{2}$$

where $\otimes$ denotes the Hadamard product.

3.4.2. IAM

To exploit correlative information across all view images, we further propose another self-attention module to generate attentive features for each view image by considering the visual features extracted from all view images. As all the view images jointly describe the 3D object instance, we named the module IAM. In particular, as shown in Fig. 4, given object $\mathcal{O}_i$ and its visual features, $\mathcal{F}_{\text{ori},i}^j (j = 1, 2, \ldots, M)$, to leverage the correlative information from all view images, we first obtain the summarizing feature by max pooling the visual features of all view images:

$$\mathcal{F}_{\text{ori},i} = \text{Maxpool}(\mathcal{F}_{\text{ori},i}^1, \mathcal{F}_{\text{ori},i}^2, \ldots, \mathcal{F}_{\text{ori},i}^M). \tag{3}$$

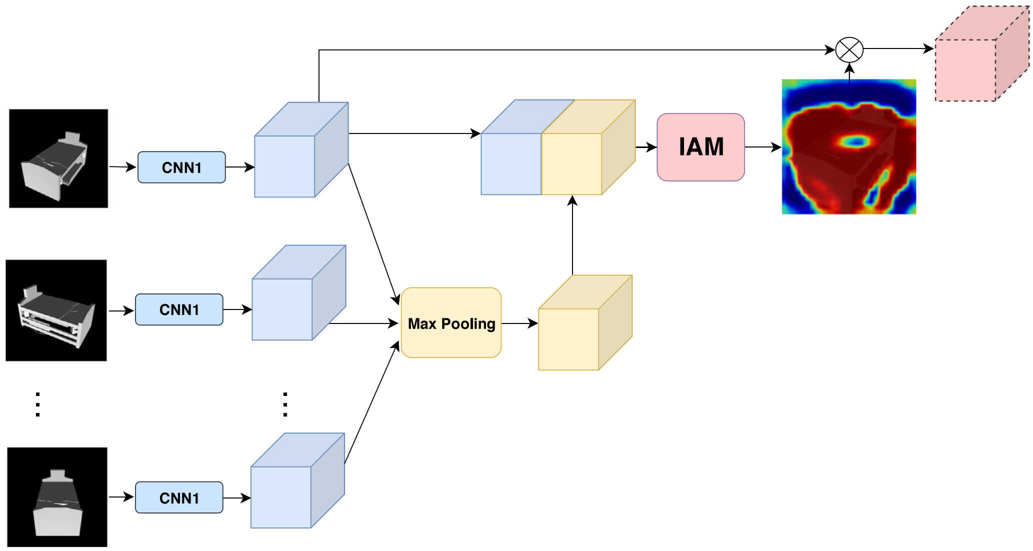

Fig. 4. Generation of instance-attentive features.

Subsequently, to incorporate instance-level information for each view-relevant feature, the summarizing feature, $\mathcal{F}_{\text{ori},i}$, is concatenated in channel with each of the individual visual features, $\mathcal{F}^j_{\text{ori},i}(j = 1, 2, \dots, M)$:

$$\mathcal{F}^j_{\text{Concat},i} = \text{Concat}(\mathcal{F}_{\text{ori},i}, \mathcal{F}^j_{\text{ori},i})\ j = 1, 2, \dots, M, \tag{4}$$

where $\text{Concat}(\cdot, \cdot)$ denotes the concatenation operation in channel dimension. The concatenated feature, $\mathcal{F}^j_{Concat,i}$, is fed through the IAM to generate an instance-level spatial attention map. Similar to the VAM, the instance-attentive feature is calculated as

$$\mathcal{F}^j_{\text{ins},i} = \mathcal{F}^j_{\text{Concat},i} \otimes \text{IAM}(\mathcal{F}^j_{\text{Concat},i}). \tag{5}$$

The IAM consists of three convolution stages. The first is a 3×3 convolution layer with 512 output channels followed by a ReLU activation and a batch normalization operation to reduce the channel dimensionality by fusing across the channels of $\mathcal{F}^j_{\text{Concat},i}$. The remaining two convolution stages of the IAM have the same architecture as the VAM, but do not share the parameters with the VAM. Compared with the VAM, the IAM is more capable of exploiting correlative information across multiple views because $\mathcal{F}_{\text{ori},i}$ represents instance-level visual information by summarizing the visual features of all view images. Hence, IAM provides the instance-level knowledge to generate spatial attention saliency.

After feeding through both the VAM and IAM for view- and instance-level attentive feature learning, for 3D model $\mathcal{O}_i$, three M attentive features denoted as $\{\mathcal{F}^j_{\text{ori},i}, \mathcal{F}^j_{\text{view},i}, \mathcal{F}^j_{\text{ins},i}\}$ $(j = 1, 2, \dots, M)$ are extracted. Subsequently, as shown in Fig. 2, we aggregate the original visual features, $\mathcal{F}^j_{\text{ori},i}$, view-attentive visual features, $\mathcal{F}^j_{\text{view},i}$, and instance-attentive visual features, $\mathcal{F}^j_{\text{ins},i}$, by element-wise max pooling the corresponding group of feature maps across views, that is,

$$\mathcal{F}_{\text{t},i} = \text{Maxpool}(\mathcal{F}^1_{\text{t},i}, \mathcal{F}^2_{\text{t},i}, \dots, \mathcal{F}^M_{\text{t},i}), \tag{6}$$

where $\text{t} \in \{\text{ori}, \text{view}, \text{ins}\}$. The aggregated features are then fed through CNN2 to produce global vectorized features:

$$\boldsymbol{f}_{\text{t},i} = \text{CNN2}(\mathcal{F}_{\text{t},i}), \tag{7}$$

where $\text{t} \in \{\text{ori}, \text{view}, \text{ins}\}$ and l_2 normalization is applied on each of the global vectorized features. During training, these vectorized features were learned through deep metric learning by minimizing the two losses introduced in the subsequent section.

3.5. Metric learning loss

After feature extraction and aggregation, we obtained three representative feature vectors for each 3D object. During training, inspired by face recognition literature [13,37], we introduced the ArcFace loss [13] as the major guidance for learning discriminative feature representations of 3D objects. Furthermore, we adopted the TCL [12] for feature learning. We observed that the joint optimization using the ArcFace loss together with the cosine-distance-based TCL could significantly improve the retrieval performance compared with the softmax loss combined with the Euclidean-distance-based TCL [6,12]. One reason is that both the ArcFace loss and cosine-distance-based TCL aim to improve the feature discriminative capability in the angular feature space. Since the cosine distance is adopted for 3D object retrieval, training with these two losses strikes a consistent learning objective between the training and testing processes.

3.5.1. ArcFace Loss

Given a 3D model, $\mathcal{O}_i$, belonging to class y_i, we obtain three l_2 normalized d-dimensional feature vectors, $\boldsymbol{f}_{\text{ori},i}$, $\boldsymbol{f}_{\text{view},i}$, and $\boldsymbol{f}_{\text{ins},i}$. Based on the assumption that K classes exist, a weight matrix, $\mathcal{W} \in \mathbb{R}^{d\times K}$, is introduced, with its k_{th} column, $\mathcal{W}_k \in \mathbb{R}^d$, corresponding to class k. $\mathcal{W}_k$ $(k = 1, 2, \dots, K)$ is also normalized by l_2. The ArcFace loss [13] introduces an additive angular margin, m, to the angle between the feature and the weight to increase the within-class compactness while enhancing inter-class discriminative capability, i.e.,:

$$\mathcal{L}^{\text{t}}_{\text{ArcFace}} = -\frac{1}{N}\sum_{i=1}^{N} \log \frac{e^{s\cos(\theta^{\text{t}}_{y_i}+m)}}{e^{s\cos(\theta^{\text{t}}_{y_i}+m)} + \sum_{k=1,k\neq y_i}^{K} e^{s\cos\theta^{\text{t}}_k}}, \tag{8}$$

where $\theta^{\text{t}}_k \triangleq \arccos(\mathcal{W}^{\text{T}}_k \boldsymbol{f}_{\text{t},i})$ and $\text{t} \in \{\text{ori}, \text{view}, \text{ins}\}$. N is the number of 3D objects in one minibatch, m denotes the additive angular margin, and s denotes the scaling factor. The overall ArcFace loss is defined as the summation of the three losses:

$$\mathcal{L}_{\text{ArcFace}} = \mathcal{L}^{\text{ori}}_{\text{ArcFace}} + \mathcal{L}^{\text{view}}_{\text{ArcFace}} + \mathcal{L}^{\text{ins}}_{\text{ArcFace}}. \tag{9}$$

3.5.2. Cosine distance TCL

We further introduce a cosine-distance-based TCL to guide representation learning. Following [12], assuming that a center vector exists for each of the K classes, we obtain a center vector

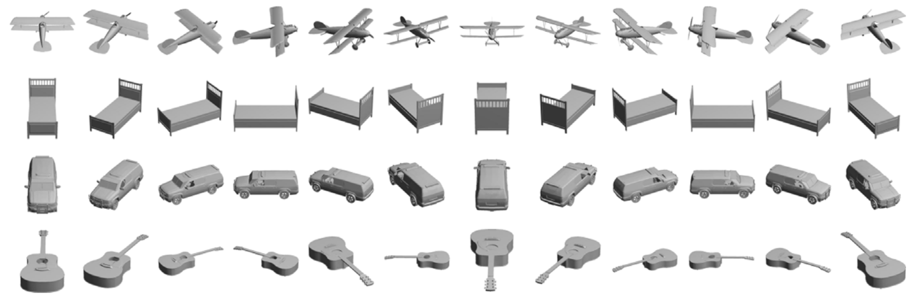

Fig. 5. Examples of view images rendered for 3D models on ModelNet dataset.

set, $\mathcal{C} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_K\}$, where $\boldsymbol{c}_k \in \mathbb{R}^d$. The TCL [12] is formulated to facilitate a feature vector to be pulled to its own class center and pushed away from the nearest negative class center within one minibatch. Specifically, given a minibatch of 3D objects, where object $\mathcal{O}_i$ is represented by $\boldsymbol{f}_{\text{ori},i}$, $\boldsymbol{f}_{\text{view},i}$, and $\boldsymbol{f}_{\text{ins},i}$, TCL is defined as

$$\mathcal{L}_{\text{TC}}^{\text{t}} = -\frac{1}{N}\sum_{i=1}^{N}\max\Big(D\big(\boldsymbol{f}_{\text{t},i}, \boldsymbol{c}_{y_i}\big) + m_{\text{tcl}} - \min_{k \neq y_i} D\big(\boldsymbol{f}_{\text{t},i}, \boldsymbol{c}_k\big), 0\Big), \quad (10)$$

where $\text{t} \in \{\text{ori}, \text{view}, \text{ins}\}$ and m_{tcl} denote the margin. As discussed, in contrast to [12], we adopted the cosine distance for $D(\cdot,\cdot)$ to coordinate with the ArcFace loss to enhance the angular discriminative capability. Here, the cosine distance is defined as

$$D(\boldsymbol{f}, \boldsymbol{c}) \triangleq 1 - \frac{\boldsymbol{f}^{\mathrm{T}}\boldsymbol{c}}{\|\boldsymbol{f}\| \cdot \|\boldsymbol{c}\|}. \quad (11)$$

The overall TCL is defined as the summation of the three TCLs:

$$\mathcal{L}_{\text{TC}} = \mathcal{L}_{\text{TC}}^{\text{ori}} + \mathcal{L}_{\text{TC}}^{\text{view}} + \mathcal{L}_{\text{TC}}^{\text{ins}}. \quad (12)$$

Finally, the total deep metric learning loss is defined as the weighted summation of both the ArcFace loss and TCL.

$$\mathcal{L}_{\text{total}} = \lambda\mathcal{L}_{\text{ArcFace}} + \mathcal{L}_{\text{TC}}, \quad (13)$$

where λ denotes the addition weight. The gradients of the overall loss, $\mathcal{L}_{\text{total}}$, with respect to the network parameters and class centers are backpropagated to tune the parameters using stochastic gradient descent (SGD) in an end-to-end fashion. The specific training procedure is summarized in Algorithm 1.

Algorithm 1 Training Procedure

Input:

- Training set $\mathcal{D}^{\text{train}}$ of 3D objects $\{\mathcal{O}_1, \mathcal{O}_2, \ldots \mathcal{O}_I\}$ belonging to K classes $\{1, 2, \ldots, K\}$, where $\mathcal{O}_i$ is represented by M view images, $\{\mathcal{I}_i^1, \mathcal{I}_i^2, \ldots, \mathcal{I}_i^M\}$ using the rendering setting described in Section 3.3.

for number of iterations **do**

- Sample a minibatch of N training $\langle\text{object}, \text{label}\rangle$ pairs $\{(\mathcal{O}_1, y_1), \ldots, (\mathcal{O}_N, y_N)\}$ from $\mathcal{D}^{\text{train}}$.

View- and Instance-Attentive Feature Learning

- Extract the original vectorized features, $\boldsymbol{f}_{\text{ori},i} (i = 1, 2, \ldots, N)$, using Eqs. (1), (6), and (7).
- Extract the view-level attentive vectorized features, $\boldsymbol{f}_{\text{view},i} (i = 1, 2, \ldots, N)$, using Eqs. (1), (2), (6), and (7).
- Extract the instance-level attentive vectorized features, $\boldsymbol{f}_{\text{ins},i} (i = 1, 2, \ldots, N)$, using Eqs. (1), (5), (6), and (7).

Deep Metric Learning

- Calculate ArcFace loss $\mathcal{L}_{\text{ArcFace}}$ using Eqs. (8) and (9).
- Calculate cosine-distance-based triplet-center loss $\mathcal{L}_{\text{TC}}$ using Eqs. (10)–(12).
- Calculate total loss $\mathcal{L}_{\text{total}}$ using Eq. (13).
- Tune the network and class centers by descending the gradients of $\mathcal{L}_{\text{total}}$ with respect to the network and class center parameters.

end for

4. Experiments

4.1. Datasets

Four public benchmarking datasets were adopted for retrieval performance evaluation, namely ModelNet40 [2], ModelNet10 [2], Mechanical Components Benchmark (MCB) [38], and ShapeNetCore55 [39]. ModelNet40 and ModelNet10 are subsets of the Princeton ModelNet dataset. ModelNet40 contains 12,311 CAD models from 40 object classes. The training and test splits in [5] were used for the ModelNet40. ModelNet10 contains 4899 CAD models from 10 object classes. In ModelNet10, 3991 CAD models were used for training, and 908 models were used for testing. We followed the settings in [6] to apply 12 view images for each 3D model in both datasets for training and testing. Fig. 5 shows examples of view images. Each row shows 12 view images rendered for a 3D object.

MCB [38] is a recently released large-scale dataset of 3D objects with mechanical components. MCB has 58,696 3D models from 68 classes of mechanical components. We adopted dataset A of MCB (MCB-A), which contains aggregated data from TraceParts, 3DWarehouse, and GrabCAD. We followed the training test split from the official dataset. As the official dataset only provides 3D models, we rendered 12 view images for each of the 3D models using Blender software. The example view images are presented in Fig. 6.

ShapeNetCore55 [39] is a subset of ShapeNet and contains approximately 51,300 3D models over 55 common categories, each of which is divided into several subcategories. We followed the official competition setting to use a 70%/10%/20% training/validation/test split from this dataset. We recorded the performance of our method using the normal version of ShapeNetCore55, in which the 3D objects were consistently aligned. The example view images are presented in Fig. 7.

4.2. Evaluation metrics

To evaluate the retrieval performance of ModelNet40 and ModelNet10, we calculated the mean average precision (MAP) by treating each 3D model in the testing set as the query sample to obtain a ranked list of all the testing models based on cosine distance. For the experiments on the MCB-A dataset, similar to the original work [38], which provided the dataset and baseline results, we adopted the F1, MAP, and Normalized Discounted Cumulative Gain (NDCG) as the performance metrics. The "micro" and "macro" averages were adopted to merge the retrieval performance over multiple testing samples. For the experiments on ShapeNetCore55, the F1, MAP, and NDCG under "micro" and "macro" settings were adopted for performance evaluation.

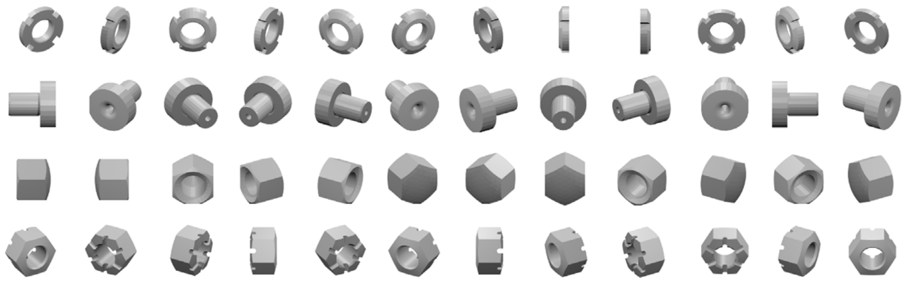

Fig. 6. Examples of view images rendered for 3D models on MCB-A dataset.

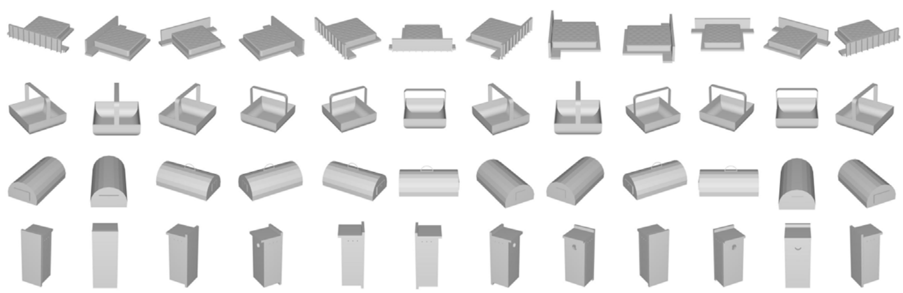

Fig. 7. Examples of view images rendered for 3D models on ShapeNetCore55 dataset.

Table 1
Comparison in terms of the MAP with state-of-the-art methods on ModelNet40 and ModelNet10.

Methods	ModelNet40	ModelNet10
SPH [40]	33.3	44.1
LFD [41]	40.9	49.8
3DShapeNet [2]	49.2	68.3
DLAN [42]	85.0	90.6
DeepPano [43]	76.8	84.2
GIFT [44]	81.9	91.1
MVCNN [5]	80.2	–
RED [45]	86.3	92.2
GVCNN [46]	85.7	–
TCL [12]	88.0	–
ATCL [11]	86.1	92.1
SeqViews [36]	89.1	91.4
VDN [47]	86.6	93.2
Batch-wise [48]	83.8	87.5
MLVCNN [28]	91.2	–
PVNet [49]	89.5	–
MVCNN (cube loss) [26]	88.2	93.4
VNN [27]	88.9	92.8
NCENet [9]	87.1	–
3DViewGraph [50]	90.5	92.4
GSL (ATCL) [6]	90.1	93.0
DAN [32]	90.4	92.3
Ours	**92.8**	**93.6**

4.3. Implementation details

For the network architecture, VGG-M pretrained on ImageNet was adopted as the backbone network. VGG-M contains eight convolution layers and three FC layers. Based on [6], we inserted a view pooling operation after the 6_{th} convolution layer. The subnetwork before the view pooling operation was used as CNN1, whereas the remaining subnetwork with the last FC layer removed was used as CNN2. CNN2 outputs a 4096-dimensional feature representation to be trained using the ArcFace loss and cosine distance TCL. For ArcFace loss, the additive angular margin m was set to 0.5, and the scaling factor s was set to 64. For the TCL, the margin m_{tcl} was set to 0.5 and weight λ was set to 0.1. Sensitive analysis experiments on the hyperparameters λ and m_{tcl} are presented in Section 5.1. We adopted SGD to tune the network parameters and class centers. For the network parameters, the learning rate, momentum, and weight decay were set to 10^{-3}, 0.9, and 10^{-4}, respectively. For the centers, the learning rate was set as 10^{-2} with the gradient clipped to the range of $[-10^{-2}, 10^{-2}]$, as suggested in [6]. During training, the batch size was set to 16, and the epoch number was 100. During the testing, the three output feature representations (shown in Fig. 2) were concatenated for the retrieval. The entire architecture was implemented using PyTorch [51] and run on an NVIDIA Telsa V100 SMX2 GPU.

4.4. Performance comparison on ModelNet datasets

We first compared the proposed method with multiple state-of-the-art methods on ModelNet40 and ModelNet10 datasets. The benchmarking methods included non-learning-based [40,41], model-based [2,40,42], and view-based methods [5,6,9,11,12,27,28,36,43–50]. For 3D object classification methods, the feature representations before the classifiers were used for similarity comparison and retrieval. As shown in Table 1, the proposed method outperforms all benchmarking methods in terms of the retrieval MAP. Specifically, the proposed method yields a MAP of 92.8% and 93.6% for ModelNet40 and ModelNet10, respectively. Compared with the original MVCNN [5], our method achieves a 12.6% performance gain in terms of the MAP on ModelNet40. In addition, compared with the MVCNN with the Euclidean TCL [12], our model achieves a 4.8% performance gain.

The performance gain comes from two major components of our method: (i) the proposed VAM and IAM which leverage the view-relevant discriminative information and the correlated information across multiple views of 3D objects, respectively, and (ii) the proposed ArcFace loss which together with cosine distance-based TCL provides effective guidance for learning discriminative representations in the angular feature space. Fig. 8 shows the qualitative retrieval results obtained using the selected testing samples on the ModelNet10 dataset. The top 10 retrieved samples are shown for each query sample. The mis-retrieved samples represent challenging cases that are semantically similar to the query object. For example, a desk can be retrieved using a table as the query object. Identifying 3D objects with similar semantics remains an open challenge.

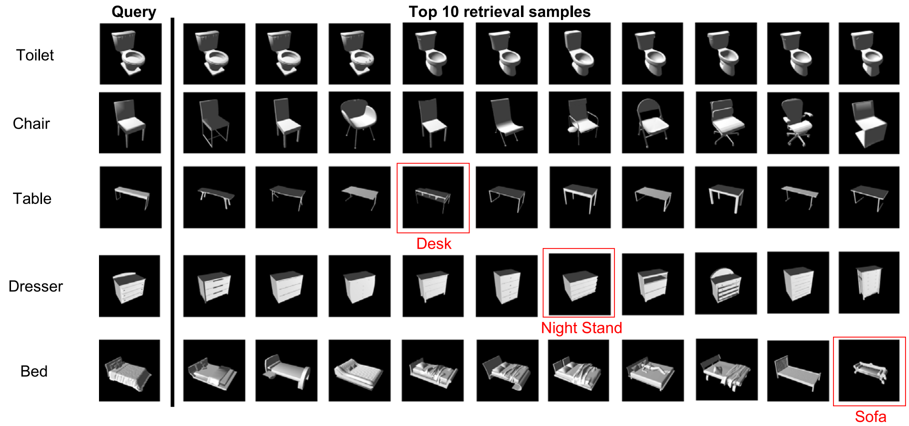

Fig. 8. Retrieval examples on ModelNet10 dataset, with the top 10 retrieval samples and false-positive retrieval objects highlighted in red rectangles.

Table 2
Comparison in terms of the MAP with state-of-the-art methods on MCB-A dataset.

Method	microALL			macroALL			microALL + macroALL		
	F1@N	MAP	NDCG@N	F1@N	MAP	NDCG@N	F1@N	MAP	NDCG@N
PointCNN [17]	0.690	0.889	0.898	**0.833**	0.886	0.854	0.762	0.883	0.876
PointNet++ [18]	0.613	0.794	0.754	0.712	0.803	0.746	0.663	0.799	0.750
SpiderCNN [52]	0.669	0.867	0.793	0.776	0.877	0.812	0.723	0.872	0.803
MVCNN [5]	0.488	0.657	0.487	0.585	0.735	0.641	0.537	0.696	0.564
RotationNet [53]	0.508	0.805	0.683	0.683	0.815	0.735	0.560	0.810	0.709
DLAN [54]	0.568	0.879	0.828	0.820	0.880	0.845	0.694	0.880	0.837
VRN [15]	0.402	0.653	0.519	0.507	0.664	0.576	0.455	0.659	0.548
Ours	**0.915**	**0.974**	**0.989**	0.822	**0.948**	**0.972**	**0.867**	**0.961**	**0.980**

4.5. Performance comparison on MCB-A dataset

We evaluated the performance of the proposed method on the MCB-A dataset. Based on [6,38], we determined "microALL" and "macroALL" to calculate the mean performance metrics over the testing samples. We also calculated the average of "microALL" and "macroALL" (denoted as "microALL + macroALL") to demonstrate the overall performance. Table 2 indicates that the proposed method significantly outperforms the benchmarking methods in terms of all the metrics except for F1@N on "macroALL" averaging, where our method still produces a comparable performance with the best benchmarking method, PointCNN. For "microALL + macroALL", compared with the best benchmarking method, the proposed method yields a performance gain of 10.5%, 7.8%, and 10.4% in terms of the F1@N, MAP, and NDCG@N, respectively. The significant improvements show the effectiveness of the proposed VAM and IAM and the metric learning losses. Fig. 9 qualitatively shows several examples with the top 10 retrieval components for each query component. The figure reveals that our method can produce effective retrieval results. The mismatched samples, such as "thrust washes" and "spacers," are from different subcategories of the same class of mechanical components. How fine-grained 3D object retrieval can be handled should be further explored in the future.

4.6. Performance comparison on ShapeNetCore55 dataset

We also evaluated the performance of the proposed method on the ShapeNetCore55 dataset. Our method was trained using the training set, and the best model was selected as the one that produced the best MAP on the validation set. Performance was evaluated on the testing set using the performance metrics provided by the official competition. We compared our method with benchmarking methods reported in the SHREC17 competition.

Table 3
Comparison in terms of the MAP with state-of-the-art methods on ShapeNetCore55 normal dataset.

Method	Micro			Macro		
	F1@N	MAP	NDCG	F1@N	MAP	NDCG
RotationNet	0.798	0.772	0.865	0.590	0.583	0.656
GIFT	0.767	0.722	0.827	0.581	0.575	0.657
ReVGG	0.772	0.749	0.828	0.519	0.496	0.559
DLAN	0.712	0.663	0.762	0.505	0.477	0.563
MVFusionNet	0.692	0.622	0.732	0.484	0.418	0.502
CM-CNN	0.479	0.540	0.654	0.166	0.339	0.404
ZFDR	0.282	0.199	0.33	0.197	0.255	0.377
VoxNet	0.253	0.192	0.277	0.258	0.232	0.337
MVCNN	0.764	0.735	0.815	0.575	0.566	0.640
Ours	**0.799**	**0.809**	**0.867**	**0.593**	**0.630**	**0.667**

Table 3 shows the retrieval performance. From the table, it can be observed that our method outperforms all benchmarking methods using all evaluation metrics. Compared with the previous best solution, RotationNet [53], the proposed method produces 3.7% and 4.7% performance gains in terms of the MAP under micro and macro settings, respectively. Compared with the baseline MVCNN model, our method can produce 7.4% and 6.4% performance gains in terms of the MAP under micro and macro settings, respectively. The experimental results show the effectiveness of the proposed attention modules and loss functions.

5. Discussion

In this section, we first analyze the hyperparameters of the proposed method. Then, we conduct ablation experiments on the attention modules and deep metric learning losses. Finally, we provide a visualization of the attention regions.

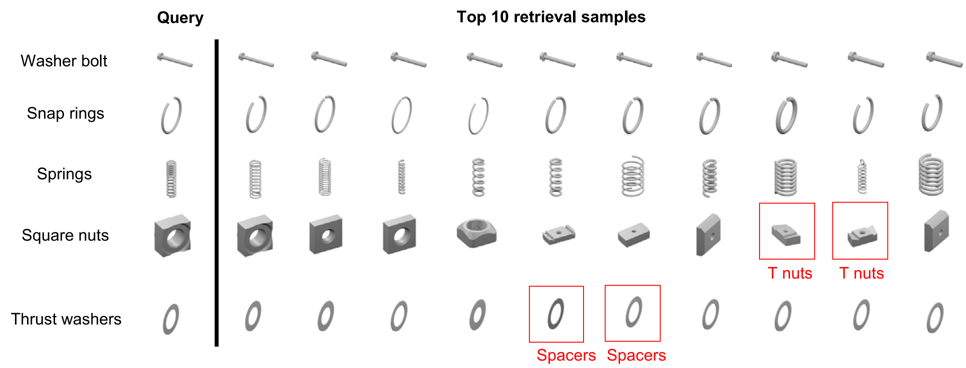

Fig. 9. Retrieval examples on MCB-A dataset, with the top 10 retrieval samples and the false-positive retrieval objects highlighted in red rectangles.

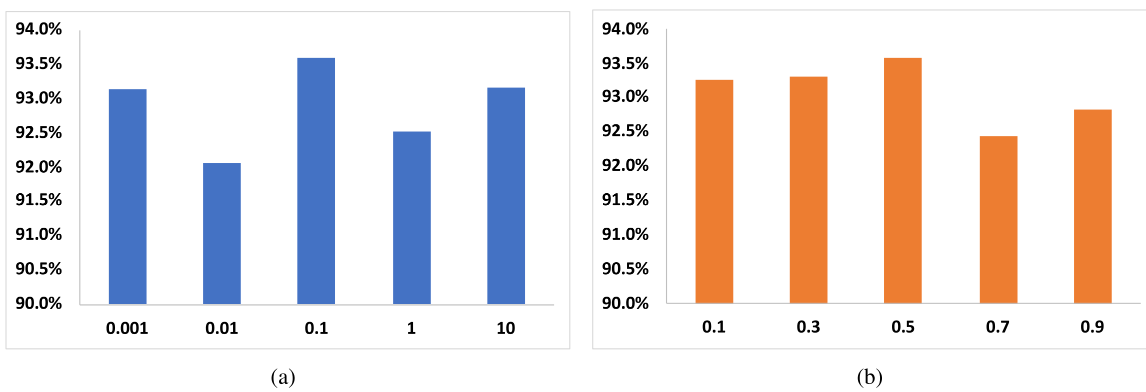

Fig. 10. Sensitive analysis in terms of the MAP with respect to (a) λ and (b) m_{tcl}.

5.1. Sensitive analysis on λ and m_{tcl}

In this subsection, we conduct a sensitivity analysis of two hyperparameters λ and m_{tcl} using the ModelNet10 dataset. For λ, we calculated the MAP values by setting λ in the range of 10^{-3}–10^{1}. Fig. 10(a) depicts that the retrieval performance deteriorates when the λ values are extremely high or low and the best performance is achieved when λ is set to 0.1. For m_{tcl}, we set λ as 0.1 and m_{tcl} in the range of 0.1–0.9. Fig. 10(b) shows that our method is robust to the values of m_{tcl} and the best performance is achieved when m_{tcl} is set to 0.5.

5.2. Ablation study on attention modules

We considered four ablation models with different attention modules: (i) the baseline model without any attention module; (ii) our method with only the VAM; (iii) our method with only the IAM; (iv) our method with both the VAM and IAM. We calculated the MAPs for the four ablation models on ModelNet40 and ModelNet10 in Table 4. The table indicates that both the IAM and VAM can work complementarily to improve the retrieval performance. Specifically, incorporating the global instance-attentive information introduced by the IAM can improve the MAP by 1.0% and 0.7% compared with the baseline model for ModelNet10 and ModelNet40, respectively. Moreover, incorporating the local view-attentive information introduced by the VAM can produce performance gains of 0.4% and 0.7% for ModelNet10 and ModelNet40, respectively. The performance gain produced by exploiting the VAM is lower than that by using the IAM on ModelNet10, because ModelNet10 is a relatively easier dataset for 3D object retrieval than ModelNet40 as it has only 10 categories. However, on ModelNet40, both the IAM and VAM can achieve comparable performance gains in terms of the MAP, thereby verifying the effectiveness of the proposed attention modules. In summary, both attention modules can be exploited jointly and complementarily to generate discriminative features for object retrieval (see Table 4).

Table 4
Ablation study in terms of the MAP with respect to attention modules.

Ablation models	ModelNet40	ModelNet10
Baseline model w/o attention module	91.4	92.2
Ablation model with VAM	91.9	92.5
Ablation model with IAM	92.1	93.2
Our model with both VAM and IAM	**92.8**	**93.6**

5.3. Ablation study on metric learning losses

We conducted two ablation experiments on the metric learning losses using ModelNet10. The first was to study different combinations of classification losses (Softamx loss or ArcFace loss) and TCLs (using the Euclidean distance or cosine distance). Specifically, we trained six models using (a) Softmax, (b) Softmax + TCL (Euclidean), (c) Softmax + TCL (cosine), (d) ArcFace, (e) ArcFace + TCL (Euclidean), and (f) Our ArcFace + TCL (cosine). Table 5 lists the retrieval MAPs obtained by these models. Table 5 also shows the MAP on ModelNet10 for our model trained using different losses. Based on the table, we make three key observations: (i) the retrieval performance is improved from 91.7% to 93.3% by replacing the Softmax loss with the ArcFace loss; (ii) for the models trained using the Softmax loss, incorporating the Euclidean distance TCL outperforms incorporating the cosine distance TCL; (iii) for the models trained using the ArcFace loss, incorporating the cosine distance TCL outperforms incorporating the Euclidean distance TCL. Although incorporating the cosine-distance-based TCL produces a relatively small MAP gain in the

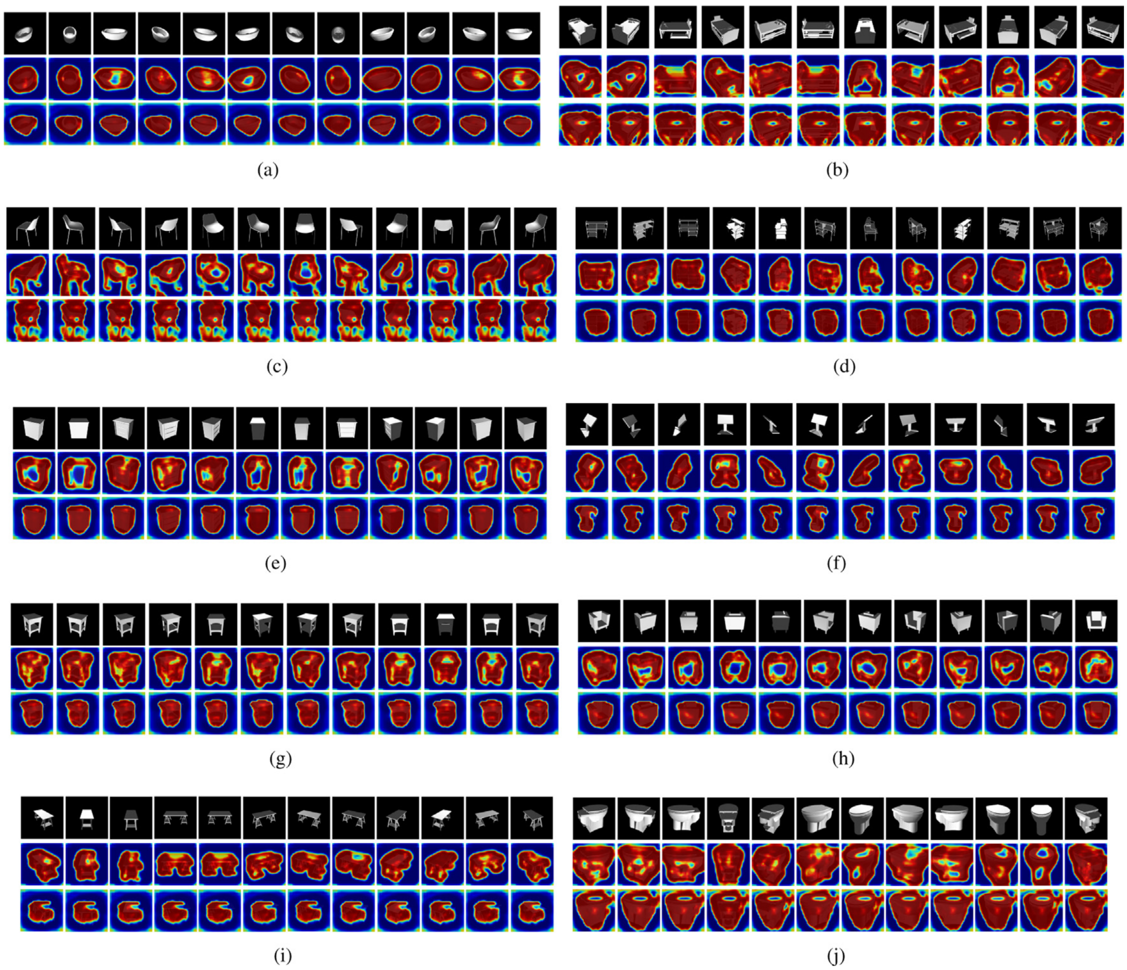

Fig. 11. Attention maps of samples from various classes in ModelNet10: (a) bathtub, (b) bed, (c) chair, (d) desk, (e) dresser, (f) monitor, (g) night stand, (h) sofa, (i) table, and (j) toilet. For each sample, the top row shows the view images rendered from different perspectives, second row shows the view attention regions, and third row shows the instance attention regions.

Table 5
Ablation study with different combinations between the classification losses and TCL losses.

Loss	MAP
Softmax	91.7
Softmax + TCL (Euclidean)	92.6
Softmax + TCL (Cosine)	92.2
ArcFace	93.3
ArcFace + TCL (Euclidean)	91.8
Our ArcFace + TCL (Cosine)	**93.6**

Table 6
Ablation study in terms of the MAP with respect to various angular space losses.

Loss	MAP
CosFace + TCL (Cosine)	92.5%
AdaCos + TCL (Cosine)	92.1%
Our ArcFace + TCL (Cosine)	**93.6**%

model trained using the AracFace loss, compared with exploiting the Euclidean-distance-based TCL, which causes a significant performance degradation (1.5%), the proposed ArcFace loss together with the cosine distance TCL represents the best loss setting of our method. This is because both the ArcFace loss and cosine distance TCL aim to improve the feature discriminative capability in the angular feature space. Since the cosine distance is adopted for 3D object retrieval, training with these two losses strikes a consistent learning objective between the training and testing processes.

The second experiment was conducted to study various angular space losses: (i) CosFace loss [14], (ii) AdaCos loss [55], and (iii) ArcFace loss [13]. Table 6 lists the retrieval MAPs obtained by employing our models trained using these angular space losses. The table indicates that the proposed model with the ArcFace loss and cosine distance TCL exhibits the best performance among the three ablation models.

5.4. Visualization of attention regions

In this section, ten testing samples (one from each category) from ModelNet10 were selected to show the attention regions highlighted by the proposed attention modules. Fig. 11 reveals that the view attention regions highlight the view-relevant local information, such as the postures, outlines, and edges, whereas the instance attention regions represent the object-relevant global information that is invariant and robust across different view perspectives.

6. Conclusion

In this paper, we proposed leveraging the aggregation of view and instance attentive features for multi-view 3D object retrieval.

The VAM and IAM are proposed to exploit the local view-relevant discriminative and global instance-relevant information, respectively. Moreover, to strike a consistent objective between the training and testing processes, we propose employing the ArcFace loss together with the cosine-distance-based TCL as the metric learning guidance to learn discriminative representations of 3D objects in the angular feature space. Extensive experiments and ablation studies conducted on four publicly available datasets confirmed the superiority of the proposed method over multiple state-of-the-art benchmarking methods for 3D object retrieval tasks. One future research direction is to exploit sequence models, such as long short-term memory or gated recurrent units, to sequentially aggregate the view- and instance-attentive features for 3D object retrieval. Moreover, it is also intriguing to further explore fine-grained 3D object retrieval in the future.

CRediT authorship contribution statement

Dongyun Lin: Conceptualization, Methodology, Software, Formal analysis, Validation, Investigation, Writing – original draft. **Yiqun Li:** Supervision, Project administration. **Yi Cheng:** Writing – review & editing. **Shitala Prasad:** Writing – review & editing. **Tin Lay Nwe:** Writing – review & editing. **Sheng Dong:** Data curation, Resources. **Aiyuan Guo:** Data curation, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research is supported by A*STAR, Singapore under its RIE2020 INDUSTRY ALIGNMENT FUND-INDUSTRY COLLABORATION PROJECTS (IAF-ICP) Grant No. I2001E0073.

References

[1] W. Nie, Y. Zhao, A.-A. Liu, Z. Gao, Y. Su, Multi-graph convolutional network for unsupervised 3D shape retrieval, in: Proceedings of the 28th ACM International Conference on Multimedia, 2020, pp. 3395–3403.
[2] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3D ShapeNets: A deep representation for volumetric shapes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1912–1920.
[3] D. Maturana, S. Scherer, VoxNet: A 3D convolutional neural network for real-time object recognition, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2015, pp. 922–928.
[4] R. Klokov, V. Lempitsky, Escape from cells: Deep kd-networks for the recognition of 3D point cloud models, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 863–872.
[5] H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3D shape recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 945–953.
[6] X. He, S. Bai, J. Chu, X. Bai, An improved multi-view convolutional neural network for 3D object retrieval, IEEE Trans. Image Process. 29 (2020) 7917–7930.
[7] H.-Y. Zhou, A.-A. Liu, W.-Z. Nie, J. Nie, Multi-view saliency guided deep neural network for 3D object retrieval and classification, IEEE Trans. Multimed. 22 (6) (2019) 1496–1506.
[8] X. Dong, H. Zhang, X. Dong, X. Lu, Iterative graph attention memory network for cross-modal retrieval, Knowl.-Based Syst. 226 (2021) 107138.
[9] H. Xu, C. Huang, D. Wang, Enhancing semantic image retrieval with limited labeled examples via deep learning, Knowl.-Based Syst. 163 (2019) 252–266.
[10] X. Shi, X. Qian, Exploring spatial and channel contribution for object based image retrieval, Knowl.-Based Syst. 186 (2019) 104955.
[11] Z. Li, C. Xu, B. Leng, Angular triplet-center loss for multi-view 3D shape retrieval, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, no. 01, 2019, pp. 8682–8689.
[12] X. He, Y. Zhou, Z. Zhou, S. Bai, X. Bai, Triplet-center loss for multi-view 3D object retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1945–1954.
[13] J. Deng, J. Guo, N. Xue, S. Zafeiriou, Arcface: Additive angular margin loss for deep face recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4690–4699.
[14] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, W. Liu, Cosface: Large margin cosine loss for deep face recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5265–5274.
[15] A. Brock, T. Lim, J.M. Ritchie, N. Weston, Generative and discriminative voxel modeling with convolutional neural networks, 2016, arXiv preprint arXiv:1608.04236.
[16] L. Minto, P. Zanuttigh, G. Pagnutti, Deep learning for 3D shape classification based on volumetric density and surface approximation clues., in: VISIGRAPP (5: VISAPP), 2018, pp. 317–324.
[17] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, B. Chen, PointCNN: Convolution on X-transformed points, Adv. Neural Inf. Process. Syst. 31 (2018) 820–830.
[18] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: Deep hierarchical feature learning on point sets in a metric space, Adv. Neural Inf. Process. Syst. (2017) 5105–5114.
[19] A. Cheraghian, L. Petersson, 3DCapsule: Extending the capsule architecture to classify 3D point clouds, in: 2019 IEEE Winter Conference on Applications of Computer Vision, WACV, IEEE, 2019, pp. 1194–1202.
[20] M. Dominguez, R. Dhamdhere, A. Petkar, S. Jain, S. Sah, R. Ptucha, General-purpose deep point cloud feature extractor, in: 2018 IEEE Winter Conference on Applications of Computer Vision, WACV, IEEE, 2018, pp. 1972–1981.
[21] S. Zhao, H. Yao, Y. Zhang, Y. Wang, S. Liu, View-based 3D object retrieval via multi-modal graph learning, Signal Process. 112 (2015) 110–118.
[22] D. Wang, B. Wang, S. Zhao, H. Yao, et al., View-based 3D object retrieval with discriminative views, Neurocomputing 252 (2017) 58–66.
[23] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. 25 (2012) 1097–1105.
[24] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[25] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[26] D. Wang, H. Yao, F. Tombari, S. Zhao, B. Wang, H. Liu, Learning descriptors with cube loss for view-based 3-D object retrieval, IEEE Trans. Multimed. 21 (8) (2019) 2071–2082.
[27] X. He, T. Huang, S. Bai, X. Bai, View n-gram network for 3D object retrieval, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7515–7524.
[28] J. Jiang, D. Bao, Z. Chen, X. Zhao, Y. Gao, MLVCNN: Multi-loop-view convolutional neural network for 3D shape retrieval, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, no. 01, 2019, pp. 8513–8520.
[29] X. Wei, R. Yu, J. Sun, View-GCN: View-based graph convolutional network for 3D shape analysis, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1850–1859.
[30] M. Yavartanoo, E.Y. Kim, K.M. Lee, SPNet: Deep 3D object classification and retrieval using stereographic projection, in: Asian Conference on Computer Vision, Springer, 2018, pp. 691–706.
[31] Z. Zhang, H. Lin, X. Zhao, R. Ji, Y. Gao, Inductive multi-hypergraph learning and its application on view-based 3D object classification, IEEE Trans. Image Process. 27 (12) (2018) 5957–5968.
[32] W. Nie, Y. Zhao, D. Song, Y. Gao, DAN: Deep-attention network for 3D shape recognition, IEEE Trans. Image Process. 30 (2021) 4371–4383.
[33] Z. Han, H. Lu, Z. Liu, C.-M. Vong, Y.-S. Liu, M. Zwicker, J. Han, C.P. Chen, 3D2SeqViews: Aggregating sequential views for 3D global feature learning by CNN with hierarchical attention aggregation, IEEE Trans. Image Process. 28 (8) (2019) 3986–3999.
[34] C. Ma, Y. Guo, J. Yang, W. An, Learning multi-view representation with LSTM for 3D shape recognition and retrieval, IEEE Trans. Multimed. 21 (5) (2018) 1169–1182.
[35] A.-A. Liu, H. Zhou, W. Nie, Z. Liu, W. Liu, H. Xie, Z. Mao, X. Li, D. Song, Hierarchical multi-view context modelling for 3D object classification and retrieval, Inform. Sci. 547 (2021) 984–995.
[36] Z. Han, M. Shang, Z. Liu, C.-M. Vong, Y.-S. Liu, M. Zwicker, J. Han, C.P. Chen, SeqViews2SeqLabels: Learning 3D global features via aggregating sequential views by RNN with attention, IEEE Trans. Image Process. 28 (2) (2018) 658–672.
[37] J. Deng, S. Zafeiriou, Arcface for disguised face recognition, in: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, 2019.

[38] S. Kim, H.-g. Chi, X. Hu, Q. Huang, K. Ramani, A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks, in: Proceedings of 16th European Conference on Computer Vision, ECCV, 2020.
[39] M. Savva, F. Yu, H. Su, M. Aono, B. Chen, D. Cohen-Or, W. Deng, H. Su, S. Bai, X. Bai, et al., Shrec16 track: largescale 3D shape retrieval from shapenet core55, in: Proceedings of the Eurographics Workshop on 3D Object Retrieval, Vol. 10, 2016.
[40] M. Kazhdan, T. Funkhouser, S. Rusinkiewicz, Rotation invariant spherical harmonic representation of 3D shape descriptors, in: Symposium on Geometry Processing, Vol. 6, 2003, pp. 156–164.
[41] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, M. Ouhyoung, On visual similarity based 3D model retrieval, in: Computer Graphics Forum, Vol. 22, no. 3, Wiley Online Library, 2003, pp. 223–232.
[42] T. Furuya, R. Ohbuchi, Deep aggregation of local 3D geometric features for 3D model retrieval., in: BMVC, Vol. 7, 2016, p. 8.
[43] B. Shi, S. Bai, Z. Zhou, X. Bai, Deeppano: Deep panoramic representation for 3D shape recognition, IEEE Signal Process. Lett. 22 (12) (2015) 2339–2343.
[44] S. Bai, X. Bai, Z. Zhou, Z. Zhang, L. Jan Latecki, GIFT: A real-time and scalable 3D shape search engine, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5023–5032.
[45] S. Bai, Z. Zhou, J. Wang, X. Bai, L. Jan Latecki, Q. Tian, Ensemble diffusion for retrieval, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 774–783.
[46] Y. Feng, Z. Zhang, X. Zhao, R. Ji, Y. Gao, GVCNN: Group-view convolutional neural networks for 3D shape recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 264–272.
[47] B. Leng, C. Zhang, X. Zhou, C. Xu, K. Xu, Learning discriminative 3D shape representations by view discerning networks, IEEE Trans. Vis. Comput. Graphics 25 (10) (2018) 2896–2909.
[48] L. Xu, H. Sun, Y. Liu, Learning with batch-wise optimal transport loss for 3D shape recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 3333–3342.
[49] H. You, Y. Feng, R. Ji, Y. Gao, Pvnet: A joint convolutional network of point cloud and multi-view for 3D shape recognition, in: Proceedings of the 26th ACM International Conference on Multimedia, 2018, pp. 1310–1318.
[50] Z. Han, X. Wang, C.-M. Vong, Y.-S. Liu, M. Zwicker, C. Chen, 3DViewGraph: Learning global features for 3D shapes from a graph of unordered views with attention, 2019, arXiv preprint arXiv:1905.07503.
[51] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, 2017.
[52] Y. Xu, T. Fan, M. Xu, L. Zeng, Y. Qiao, Spidercnn: Deep learning on point sets with parameterized convolutional filters, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 87–102.
[53] A. Kanezaki, Y. Matsushita, Y. Nishida, Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5010–5019.
[54] T. Furuya, R. Ohbuchi, Diffusion-on-manifold aggregation of local features for shape-based 3D model retrieval, in: Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, 2015, pp. 171–178.
[55] X. Zhang, R. Zhao, Y. Qiao, X. Wang, H. Li, Adacos: Adaptively scaling cosine logits for effectively learning deep face representations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 10823–10832.