Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

# Efficient 3D object recognition in mobile edge environment

Mofei Song[1,2*] and Qi Guo[1,2]

**Abstract**

3D object recognition has great research and application value in the fields of automatic drive, virtual reality, and commercial manufacturing. Although various deep models have been exploited and achieved remarkable results for 3D object recognition, their computational cost is too high for most mobile applications. This paper combines edge computing and 3D object recognition into a powerful and efficient framework. It consists of a cloud-based rendering stage and a terminal-based recognition stage. In the first stage, inspired by the cloud-based rendering technique, we upload the 3D object data from the mobile device to the edge cloud server for multi-view rendering. The rendering stage utilizes the powerful computing resource in the edge cloud server to generate multiple view images of the given 3D object from different views by parallel high-quality rendering. During the terminal-based recognition stage, we integrate a lightweight CNN architecture and a neural network quantization technique into a 3D object recognition model based on the multiple images rendered in the edge cloud server, which can be executed fast in the mobile device. To reduce the cost of network training, we propose a novel semi-supervised 3D deep learning method with fewer labeled samples. Experiments demonstrate that our method achieves competitive performance compared to the state-of-the-art methods with low latency running in the mobile edge environment.

**Keywords:** Cloud computing, Deep learning, 3D object recognition

## Introduction

Recently, with the rapid development of 3D scanning technology, many mobile devices are equipped with portable 3D sensors to realize 3D object reconstruction. These mobile devices make 3D sensing usable anytime and anywhere at a low cost. And the development of 3D sensing ability leads to high-quality 3D object data with more points and geometric details. With the increase of 3D data, it is crucial to realize 3D object recognition for mobile applications, such as automatic drive [1], virtual reality, commercial manufacture, smart city [2, 3], and social network.

Since the size of the 3D data is increasingly large, more computational burden is required to render and analyze the 3D object. However, in practice, the computing power and energy supply of mobile devices [4, 5] are usually too limited to execute such a complex task efficiently. Thus, to realize 3D object recognition for mobile applications, a simple idea is to offload the complex 3D data process tasks from the mobile device to the remote cloud with powerful computing resources [6, 7], and then transmit the recognition results back to the mobile device. However, the amount of 3D object data being transmitted will exceed the capacity of the network. To meet the latency constraint, mobile edge computing is a vital solution by offloading the computation task to the mobile edge servers. Since the mobile edge servers are physically located close to the mobile devices, the timing of the data transmission can be saved.

Nowadays, deep learning technology [8] has achieved great advances in 3D object recognition [9–11]. Among the existing methods, the view-based method [12, 13] achieves the best performance since it does not rely on

*Correspondence: songmf@seu.edu.cn

[2] The Key Lab of Computer Network and Information Integration (Ministry of Education), Southeast University, Nanjing, China
Full list of author information is available at the end of the article

Springer Open

© The Author(s) 2022. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

complex 3D features. This method first renders multiple 2D images of the 3D object from different views and then uses a 2D CNN network to extract the feature from the 2D images for information fusion. Though the view-based methods have shown impressive performances in 3D object recognition, the success of these advances relies on powerful computing resources, such as multi-core CPU and strong graphics processing unit (GPU). Due to the limited computing power and cache space of the mobile device [14], both high-quality 3D rendering and large neural network inference are very expensive tasks, which will lead to high latency. Thus, existing view-based 3D object recognition methods are not suitable to be deployed on mobile device.

To make a good tradeoff between the limited computing resources of the mobile device and the performance of view-based 3D object recognition, we perform a systematic design study to achieve high-performance 3D object recognition for mobile applications based on an edge computing framework. Instead of offloading the whole view-based recognition process in the edge cloud server, we choose to split the whole view-based 3D object recognition process into two subtasks and then distribute these two subtasks on different computing devices in the mobile edge environment. The two separable subtasks are multi-view rendering and recognition model execution, and they are divided based on the following reasons. The first one is that the quality of the rendered images has a significant compact on the accuracy of the recognition model. However, it is impossible to realize photorealistic rendering by mobile devices with limited graphic display capability. The second one is that the recent mobile platform provides some artificial intelligent processor runtime mechanisms to accelerate the execution of deep neural networks. Accordingly, to optimize the computation offloading of our edge computing framework, we distribute the two subtasks on the edge cloud server and the mobile terminal. The edge cloud server is responsible for multi-view rendering, while the mobile terminal executes the deep recognition model. Therefore, the proposed edge computing framework not only achieves low latency and high reliability by maximizing the use of computing power in the mobile cloud environment but also ensures the quality of the rendered image by utilizing the dedicated rendering hardware in the edge cloud server, hence resulting in higher recognition accuracy.

According to the offloading decision, our edge computing framework first uploads the captured 3D data from the mobile terminal to the edge cloud server. Then, the edge cloud server employs the dedicated ray-tracing hardware to realize fast photorealistic rendering of a 3D object. By designing parallel processing in the edge cloud server, multiple photorealistic images are rendered simultaneously to reduce the latency of the whole edge computing framework. Finally, the generated images are sent back from the edge cloud server to the mobile terminal, and a lightweight CNN model is used to realize view-based 3D object recognition on the mobile device. The proposed edge computing framework only offloads the rendering task to the edge cloud server. Such an edge computing structure is very useful when multiple users send 3D object recognition requests at the same edge cloud server with finite resources. If we offload all the tasks on the edge cloud server, it may exceed the cloud capacity.

To support such an edge computing framework, one core technical challenge is to design a lightweight CNN architecture with high recognition accuracy, which can be executed effectively and efficiently on a mobile device. Since most existing view-based 3D object recognition methods rely on very complex network, they are time-consuming for resource-constrained mobile devices. Recently, a few works [15] propose a lightweight volumetric CNN architecture with low resolution, but the recognition accuracy is too limited to satisfy the requirement for our edge computing framework. Another challenge is to reduce the cost of implementing the mobile applications, as the current 3D object recognition methods require a large amount of labeled data, such as ModelNet [16], ShapeNet [17], PartNet [18, 19], and 3DFUTURE [20]. Though some methods are proposed to focus on learning 3D object deep classifiers with fewer data resources [21, 22], the low recognition accuracy and the time-consuming computing of these methods are unable to meet the quality of the cloud services.

To overcome these two problems, we propose several strategies to obtain an accurate 3D object recognition model with the limited computing and data resources in the mobile edge environment. To reduce the latency of executing the recognition model in the mobile device, we design a lightweight CNN architecture by combining the 3D multi-view learning framework [12, 23] and ShuffleNet [24]. Such a method first extracts the feature of each rendered image with an efficient CNN architecture ShuffleNet pre-trained on massive image databases [25] and then fine-tunes on the 3D data by aggregating multi-view features. To improve the efficiency in the mobile edge environment, we employ neural network quantization technology to compress the model without accuracy degradation. To reduce the cost of implementing the whole edge computing framework, we propose a semi-supervised 3D feature learning algorithm based on FixMatch [26] to utilize the massive amount of unlabeled shapes and a few labeled samples for model training. The unlabeled objects can be easily captured by the mobile devices connected to the cloud center. To realize the 3D

feature learning, we first use the FixMatch algorithm to learn a 2D image network for every rendered image of the 3D object derived from the cloud-based rendering stage. A 3D multi-view learning algorithm is then used to learn a 3D object recognition model by integrating the 3D view consistency constraints between different rendered images of the same 3D object. The learned model can be finally deployed on mobile devices with a central processing unit (CPU) or digital signal processor (DSP).

The contributions of this study are summarized as follows: 1) We propose an efficient 3D object recognition method based on the edge computing paradigm for mobile applications, which consists of a cloud-based rendering stage and a terminal-based recognition stage to maximize the use of the computing resources in the mobile edge environment. 2) We propose a lightweight 3D CNN architecture by integrating neural network quantization with high recognition accuracy, which reduces the latency of the edge computing framework. 3) We introduce a semi-supervised learning algorithm for view-based 3D object recognition, which saves the cost of implementing the edge computing framework for mobile applications.

## Related work

This work relates to several research areas. In this section, we briefly review the existing work on 3D object recognition and edge computing.

### 3D object recognition

According to the type of supervision, 3D object recognition methods can be divided into three classes: unsupervised, supervised, and semi-supervised methods. Unsupervised learning uses unlabelled data to train the feature representation. These methods utilize the 3D raw data to learn the high-level features directly by minimizing reconstruction loss [27–30] or self-supervised learning [23, 31, 32]. For example, Afham et al. [32] use a cross-modal contrastive learning approach to align the prediction between the point clouds and the rendered 2D image. However, some labeled objects are still required to obtain a 3D object classifier. Moreover, the recognition accuracy of these methods is lower than that of the supervised ones.

Recently, the supervised method becomes the mainstream technology for 3D object recognition, which learns the feature representation from different 3D raw representations, such as points clouds [33–37] and views [12, 13]. Point-TnT [36] is a two-stage point feature learning approach by fusing local and global attention modules, while Point-Stack [37] uses multi-resolution feature learning and learnable pooling to extract high-semantic point features. Among these methods, the view-based representation usually has the best performance. However, the supervised methods require a large data resource.

Semi-supervised learning utilizes both labeled and unlabeled data to train a recognition model by pseudo-labeling or consistency constraints. Song et al. [38] use the co-training algorithm to realize the end-to-end training by pseudo-labeling the unlabeled samples. Chen et al. [39] utilize the consistency constraints on the point cloud to realize a semi-supervised learning framework. Shi et al. [40] utilize bi-level optimization to solve the semi-supervised learning problem for 3D object classification, which combines a weight predictor network to define the weights of the samples. Different from existing semi-supervised learning methods, our method combines the consistency constraints and pseudo-labeling into a whole framework, which can generate a more accurate recognition rate.

Due to the complexity of 3D data, existing deep 3D object recognition methods usually use cumbersome network architectures to extract effective 3D features. To solve this problem, some real-time 3D object recognition methods are proposed. LightNet [15] is a lightweight volumetric 3D CNN for 3D object recognition, which has fewer training parameters. PVENet [41] is an architecture with small voxel grid sizes for real-time point cloud classification. All of these methods learn the feature representation from voxels with a small resolution ratio to improve efficiency. However, the geometric details are lost due to the small resolution, and the natural inefficiency of 3D convolution operation prevents the improvement of the inference speed. In contrast, our method uses the multi-view 3D representation and each rendered image has sufficient resolution. Through lightweight CNN and the neural network quantization technique, we achieve a high recognition rate and inference efficiency simultaneously.

### Edge computing

There is a close connection between cloud computing and big data analysis [42, 43]. Cloud computing naturally provides distributed computing, parallel computing, and large storage capacity, which is very suitable for a big data model. Besides, cloud computing is helpful for system updating and management, which ensures the stability of mobile applications. However, the delay of communication with mobile cloud servers causes time-sensitive mobile applications unable to meet the requirements. Recently, mobile edge computing has become a promising paradigm to meet the computing resource requirements of mobile devices [44]. Edge computing has been widely used in many mobile application [45] with high load and high concurrencies, such as smart city [2, 3],

smart building [46, 47], intelligent manufacturing [48], and geographical POI recommendation [49]. By sharing configurable computing resources in the mobile edge environment, edge computing can meet the computation demand of various mobile applications and improve user experience.

To design an effective edge computing system, the communication latency and energy cost are the most important optimization objectives to ensure the quality of services [4, 50]. Thus, many works are propose for task offloading optimization. For example, deep reinforcement learning methods [46, 51, 52] are used for task offloading and resource management to balance the processing delay and the power consumption. Bi et al. [53] propose an optimal offloading scheme to maximize the system utility based on throughput and fairness. Our method are inspired with these works, and we design an edge computing framework, which offloads the cumbersome rendering task to the edge cloud server.

A recent trend in academia and industry is to combine artificial intelligent and edge computing into the mobile application. Actually, artificial intelligent and edge computing can benefit from each other to improve the efficiency of the system. Artificial intelligent can be used to optimize the edge service to reduce the consumption of time and energy in data computations and transmissions [1]. Recently, deep learning becomes the mainstreaming method for various applications such as secure healthcare [54], object detection [55] and semantic segmentation [56]. Due to its high computational complexity, deep learning is usually not suitable for mobile applications. And edge computing can provide powerful computing resources to support the efficient training and inference of the artificial intelligent model, which can produce some novel learning paradigms [49]. Our method conforms such a development trend, and we use edge computing to process and analysis the large-scale 3D data. By integrating the computing resources in the edge cloud servers, we alleviate the lack of computing power in the 3D object recognition mobile applications.

## Method

In this section, we introduce the details of our 3D object recognition method.

### Overview

In this paper, we study the efficient 3D object recognition method in the mobile edge environment. The overall idea is that we combine edge computing and 3D object analysis to improve the user experience of mobile applications. By utilizing the computing resources of the edge cloud servers, we can realize high-through complex 3D object analysis tasks with low latency. The process of our method is shown in Fig. 1.

Our edge computing framework consists of many mobile devices. And each mobile device connects to a nearby edge cloud server with high-speed network connectivity based on the location information. We assume that the mobile device is equipped with a portable 3D scanning sensor, which can realize 3D reconstruction of the real object in the physical world. During the 3D reconstruction process, the 3D data is compressed by the distance-based compression method on the mobile device [57]. When users need to obtain the category of the captured 3D object by the mobile device, they can send the request to our edge computing framework. The edge cloud server and the mobile terminal can collaboratively accomplish the recognition task to respond to user

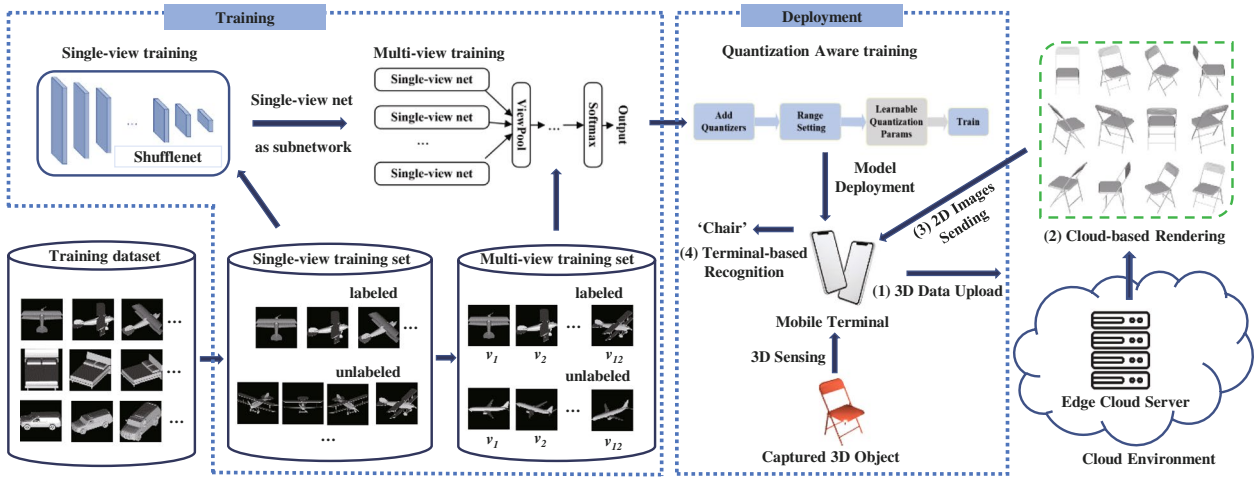


**Fig. 1** The process of our method

requests. The whole process consists of the following four steps.

First, the captured 3D data is uploaded from the mobile device to the edge cloud server. To improve the speed of the data transmission, we discard the normal and color information of the captured 3D data, while transmitting only the compressed geometric data. Since the 3D data has been compressed during the 3D reconstruction process, the timing of data transmission is very short.

Second, the edge cloud server will arrange the recognition tasks for multiple requests. For each request, the edge cloud server decompresses the 3D data after receiving the data and then starts to render 2D images. Our method follows existing view-based methods [12] and renders 12 images according to a fixed viewpoints setting for 3D object recognition. Since the rendering processes of the 12 images are totally unrelated, the edge cloud server performs parallel off-screen rendering by the real-time ray tracing algorithm [58].

Third, the edge cloud server sends the 12 rendered images back to the mobile device via the mobile network. As the resolution of the image is only 224, the timing of data transmission can be negligible.

Fourth, the mobile device takes the 12 rendered images from the edge cloud server as the input of our specific lightweight 3D CNN network, and executes the recognition model to output the prediction result to users. During this step, 3D CNN network can be executed by three modes: CPU, GPU and DSP. The users can choose one of the three modes according to the mobile terminal used by compromising the performance and speed.

Among the four steps, the first and third steps are only responsible for data communication between the mobile device and the edge cloud server. In the following section, we detail the second and fourth steps, which are cloud-based rendering and terminal-based recognition.

#### Cloud-based rendering

The goal of cloud-based rendering is to render multiple 2D photorealistic images by the edge cloud server, which are the input of the recognition model. As shown by previous research [59], the recognition accuracy increases with the improvement of the quality of the rendered images. However, the 3D graphic ability of the mobile device is rather limited, and it requires dedicated hardware to generate most optical effects. Thus, the only choice is to perform the rendering task on the edge cloud server.

To realize the cloud-based rendering, we set up the viewpoint set in advance and save them in the edge cloud server. To ensure the features captured from the different views can be complementary, we put 12 cameras at different positions around the 3D object, which is the same as the pioneering work MVCNN [12]. To put the 3D object into the rendering scene, we use the upright orientation method [60] to adjust the pose of the 3D object and then put the center point at the origin of the 3D coordinate system. The 12 cameras are elevated 30 degrees from the ground plane, pointing toward the centroid of the 3D object, every 30 degrees around the object.

To make the cloud-based rendering result conspicuous, we add some lights into the 3D scene, which shows some optical effects, such as shadow, reflection, and refraction. To handle the illumination, we generate the rendering result by utilizing a real-time ray tracing algorithm [58], instead of the rasterization technology. The ray-tracing algorithm simulates the basic principle of vision, which is the process of shooting rays from the eye toward the pixels of the rendered image. Thus, the algorithm first checks the intersection between the ray and every triangle face of the 3D object and then determines the shading of the corresponding pixel in the rendered image. In practice, the decompressed 3D object usually has more than $100K$ triangles, thus the complexity is too overwhelming due to the inefficiency in handling irregular ray tracing.

As a result, the only viable solution is to resort to powerful dedicated ray-tracing hardware, such as NVIDIA graphics cards. To deal with high concurrency user requests, all the edge cloud serves are equipped with multiple dedicated graphics cards for a high-performance ray tracing process. Due to the separable feature of multi-view rendering, the edge cloud server can parallel run the off-screen rendering subtasks. This reduces the latency of mobile applications significantly.

One challenge of rendering implementation on the edge cloud server is that the edge cloud servers usually have heterogeneous graphics processing units and operating systems. To solve it, instead of using some specific shading languages, we interact with the graphics renderer by OpenGL program as a shared library, which provides unified APIs for 3D graphics rendering. The shared library is then deployed on all the edge cloud servers to realize the ray-tracing rendering process with a small amount of reprogramming. Finally, the 12 2D rendered images of size $224 \times 224$ are generated in the edge cloud server and sent back to the mobile device.

#### Terminal-based recognition

Previous multi-view 3D object recognition methods use a much more complex network to maximize the recognition performance. However, these networks are too large to be deployed on mobile devices. To improve the efficiency of our model in the mobile terminal, we design a lightweight multi-view CNN architecture based on ShuffleNet [24], which has fewer parameters than the other

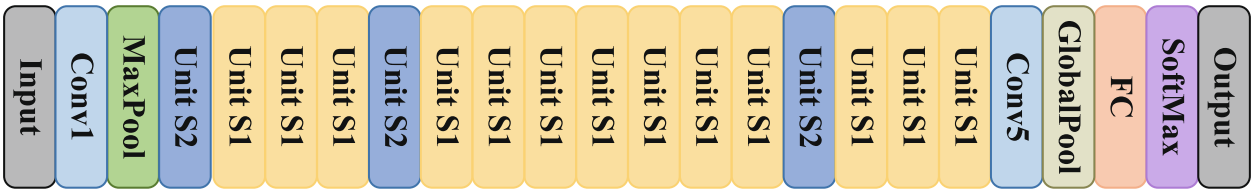


**Fig. 2** The architecture of the ShuffleNet

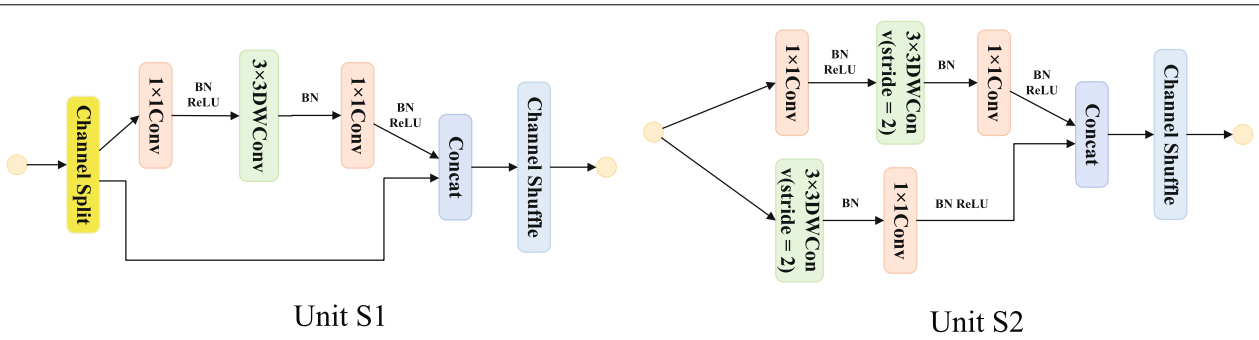


**Fig. 3** The architecture of the unit *S*1 and *S*2 in the ShuffleNet

CNN architecture, such as VGG11 [61] and ResNet [62]. It is worth noting that there are other lightweight networks for 2D images, such as SqueezeNet [63], Xception [64], and MobileNet [65]. SqueezeNet uses a deeper network to reduce the number of parameters, which requires more inference time. Xception proposes depthwise separable convolution to improve the performance, but the amount of the parameters not reduces impressively. MobileNet constructs a lightweight network by combining depthwise convolution and pointwise convolution, which hinders the information transmission. Thus, we choose ShuffleNet for its high information interchange between different channels. The input of our CNN network is the 12 rendered images generated on the edge cloud server. The CNN network outputs the category of the 3D object as the response to the user request.

**Network Architecture.** Given each rendered image, we first use the ShuffleNet network as the basic CNN architecture to compute the image descriptors from each rendered image. The architecture of the ShuffleNet is shown in Fig. 2. The core of the ShuffleNet is the channel shuffle operation which helps the information in different groups flow to other groups randomly. Based on the channel shuffle operation, we define two types of units *S*1 and *S*2, as shown in Fig. 3. Unit *S*1 is the basic unit for feature encoding based on the channel split operation, which splits the input of feature channels into two branches. Meanwhile, unit *S*2 is used for spatial down sampling by removing the channel split operator. The final ShuffleNet architecture is composed of several layers of units *S*1 and units *S*2, which is highly efficient for mobile device.

To aggregate the image descriptors of every rendered image, our multi-view CNN model utilizes a view pooling layer for the fusion of multiple views of the 3D objects in no specific order. The view-pooling layer only considers the view with the maximal activation, thus we simply use a max pooling operation for information fusion. Finally, a SoftMax layer is added as the classification layer, which generates the category prediction results. The whole architecture of our multi-view CNN model is shown in Fig. 4. Our experiment shows that such a lightweight CNN architecture can be executed fast by mobile devices.

**Model Training.** To save the cost of implementing the edge computing framework, we design a two-stage training algorithm by semi-supervised learning algorithm FixMatch [26]. This algorithm reduces the amount of labeled data, which is beneficial for the upgrading of the edge computing framework. There are lots of semi-supervised methods, which can be divided into two categories: pseudo-labeling and consistency regularization. The pseudo-labeling methods first learn a deep model by the labeled examples and then use the learned model to predict the other examples, which are used to incrementally model training. In contrast, the consistency regularization methods utilize the prediction invariance after the random transformation of training data as the regularization term. To integrate the advantages of these two methods, we choose to use FixMatch for model

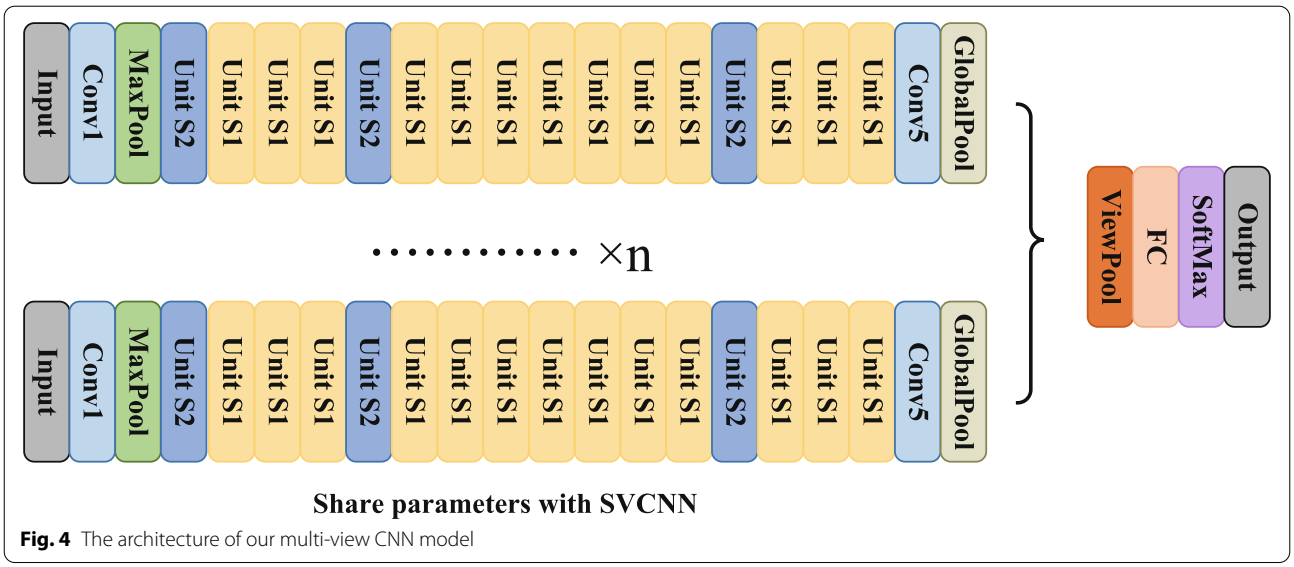


**Fig. 4** The architecture of our multi-view CNN model

training, which generates the labels of unlabeled samples by combining consistency regularization and pseudo-labeling. The input of our model training algorithm is a set of 3D objects $D = L \bigcup U$, where the objects in $L$ are labeled and the ones in $U$ are unlabeled. For each labeled object $s_i \in L$, we attach a category tag $c_i$ as its label, and all the category tags form the label set $C$. Every object is expressed as 12 images $\{m_j\}$, and its category tag is the same as the 3D object.

The existing multi-view methods usually use the network pre-trained on the ImageNet directly to initialize the weights of the whole network. Since the training samples are all rendered images, the initial weights do not reflect the feature of these samples due to the domain discrepancy. Thus, we finetune the network by our rendered images before the multi-view learning stage. Accordingly, there are two stages in our model training step: SVCNN and MVCNN. The goal of the SVCNN stage is to train the image representation network which is a part of our whole model, while the MVCNN stage is used to train our multi-view CNN model for 3D object recognition.

During the SVCNN stage, we use the FixMatch algorithm to train the image network ShuffleNet $S$, as shown in Fig. 5. The input of the image network is one rendered image, and the output is the category prediction of the image. To utilize the unlabeled rendered images, we define several image augmentation operations and generate their artificial labels by consistent regularization and pseudo-labeling generation. There are two types of image augmentation operations: weak augmentation and strong augmentation. We indicate $\alpha(\cdot)$ and $\beta(\cdot)$ as the weak augmentation and strong augmentation, respectively. The weak augmentation operation takes the standard flip or shift transformation strategy. The images are flipped horizontally with a probability of 50% and translated by up to 12.5%. By contrast, the strong augmentation operation produces a more distortion effect. We first perform no more than 4 augmentation operations from RandAugment [66] and CTAugment [67], and then randomly select a small square from the augmented image. During the strong augmentation process, the gray values of some pixels are changed to a certain value.

For the labeled rendered image $m_L$ with the label $c$, we use the standard cross-entropy loss as the supervised loss $L_s$ on weakly augmented labeled images:

$$L_s(m_L) = H(c, S(\alpha(m_L))) \tag{1}$$

For the unlabeled rendered image $m_U$, we perform the weak augmentation operation and predict the category of the weakly augmented image by the network $S$. If the network $S$ can give a confident result, i.e. $\max S(\alpha(m_U)) \geq \tau$, the label $c' = \arg\max S(\alpha(m_U))$ is taken as the pseudo-label of the image $\alpha(m_U)$. Accordingly, we use the cross-entropy loss as the unsupervised loss $L_u$ on the strong augmented image

$$L_u(m_U) = H(c', S(\beta(m_U))) \tag{2}$$

The final loss function of the SVCNN stage is defined by adding the supervised loss of all the labeled images and the unsupervised loss of all the unlabeled images:

$$L_1 = \frac{1}{B}\sum_{b=1}^{B} L_s(m_{Lb}) + \frac{1}{\mu B}\sum_{b=1}^{\mu B} L_u(m_{Ub}) \tag{3}$$

where $B$ is the number of the labeled images, $\mu$ is the ratio between the number of the labeled and unlabeled images, and the number of all the images is $(1+\mu)B$.

During the MVCNN stage, we combine the FixMatch and view consistency to train the whole MVCNN network $M$. The input of the network is 12 rendered images $\{m_j\}$ which belong to the same 3D object, and the output of the network is the category prediction of the 3D object. The learned network $S$ in the SVCNN stage is part of the MVCNN network, as shown in Fig. 4. The 12 Shufflenet networks share the same parameters and are used to extract the image descriptors. When combining the Shufflenet networks into the MVCNN network, we remove the SoftMax layer and perform the view pooling layer on the penultimate layer of the shufflenet network. As done in the SVCNN stage, we also perform the same image augmentation operations on the rendered images to define the loss function on the unlabeled 3D objects. For the labeled 3D object expressed by $\{m_j^L\}$ with the label $c$, we perform the weakly augmentation operation and use the standard cross-entropy loss as the supervised loss $l_s^M$

$$L_s^M(\{m_j^L\}) = H(c, M(\alpha(m_1^L), ..., \alpha(m_{12}^L))) \tag{4}$$

For the unlabeled 3D object expressed by $\{m_j^U\}$, we perform the weak augmentation operation on all the rendered images and predict the category of the 3D object by the network $M$. If the network $M$ can give a confident result, i.e. $\max M(\alpha(m_1^U), ..., \alpha(m_{12}^U)) \succeq \tau$, the label $c' = \arg\max M(\alpha(m_1^U), ..., \alpha(m_{12}^U))$ is taken as the pseudo-label of the 3D object $\{m_j^U\}$. Accordingly, we use the cross-entropy loss as the unsupervised loss $l_u^M$ on the unlabeled 3D object

$$L_u^M(\{m_j^U\}) = H(c', M(\beta(m_1^U), ..., \beta(m_{12}^U))) \tag{5}$$

Since the rendered images $\{m_j^U\}$ belong to one 3D object, they should have the same category label. According to the observation, we add a view consistency term to boost performance. For a set of rendered images $\{m_j^U\}$, we perform weak augmentation and strong augmentation, and then minimize the divergence between the prediction of these augmented images. To realize such minimization, we compute the standard deviations of the prediction of the weakly augmented and strongly augmented images by the shufflenet $S$ respectively

$$\begin{aligned} L_{std}^{\alpha} &= Std(S(\alpha(m_1^U)), ..., S(\alpha(m_{12}^U))) \\ L_{std}^{\beta} &= Std(S(\beta(m_1^U)), ..., S(\beta(m_{12}^U))) \end{aligned} \tag{6}$$

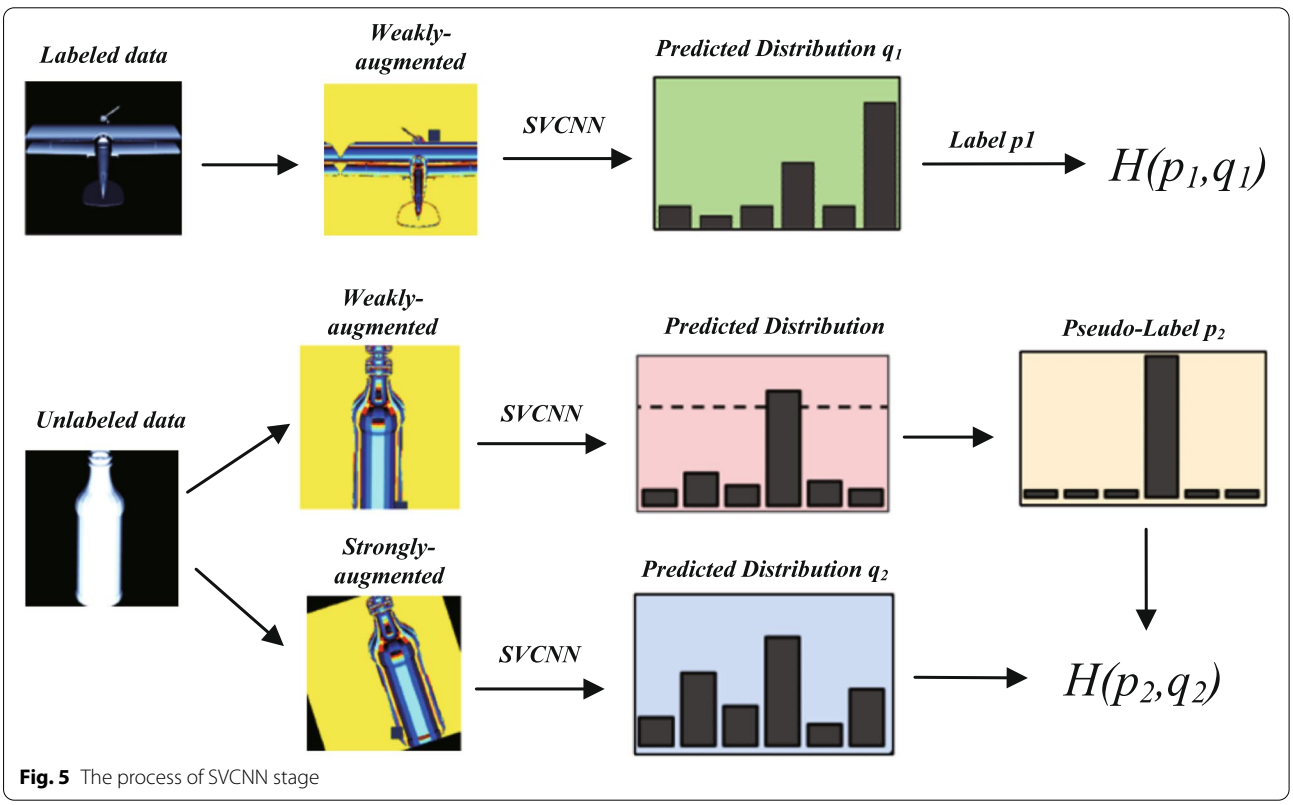


**Fig. 5** The process of SVCNN stage

Based on these two standard deviations, we can measure the consistency degree between the predictions of different views of the 3D object. Thus, we define the view consistency term as follows:

$$L_v = L_{std}^{\alpha} + L_{std}^{\beta} \tag{7}$$

Accordingly, the final loss function of the MVCNN stage is defined as:

$$L_2 = L_s^M + L_u^M + L_v \tag{8}$$

Given the loss function, we optimize the network through back-propagation with stochastic gradient descent [68] with decreasing learning rates.

**Model Deployment.** The learned deep model is finally deployed in the mobile terminal. After the terminal receives the 12 images rendered in the edge cloud server, it uses the deep model for 3D object recognition. We provide three running modes on the mobile terminal: CPU, GPU, and DSP. The speed of the CPU mode is the lowest, while the speed of the DSP mode is the fastest. For the recognition accuracy, the CPU and GPU modes are the same, while the DSP mode is the lowest. The reason is that the DSP mode only supports the quantized model, which has a quantization error. The user can choose one of the three modes according to mobile device, and whether to quantize the deep model.

The goal of the neural network quantization technology [69] is to decrease the computational time and energy consumption of the mobile device. After quantization, the weights and parameters are stored in lower bit precision and the computational cost for matrix multiplication reduces quadratically. By network quantization, the latency of our edge computing framework can be reduced impressively. However, quantization without any fine-tuning might degrade the recognition accuracy. To avoid this problem, we use the quantization-aware training method [70] to mitigate the quantization error.

To perform the quantization-aware training, we first introduce the quantization simulation block into every layer of our model. The quantization simulation block will turn the real-valued vector $v$ into the integer vector $v_{int}$ by the rounding and clamping operation. Specially, given a real-valued vector $v$, we first map it to the integer grid $\{0, 1, ..., 2^8 - 1\}$:

$$v_{int} = clamp(\left\lfloor \frac{v}{s} \right\rceil + z; 0, 2^8 - 1) \tag{9}$$

where $\lfloor \cdot \rceil$ is the round-to-nearest operator, $s$ is the factor, $z$ is the zero point. And, $s$ and $z$ are optimized during the quantization-aware training. The clamping is defined as:

$$clamp(x; a, c) = \begin{cases} a, & x < a \\ x, & a \le x \le c \\ c, & x > c \end{cases} \tag{10}$$

To fine-tune such a network, we need to back-propagate through the quantization simulation block. However, the gradient of the round-to-nearest operation is not well defined. To measure the gradient, the straight-through estimator is utilized and the gradient of the round-to-nearest operator is equal to 1. According to this approximation, we can use the standard back-propagation algorithm to fine-tune our MVCNN network with the quantization simulation block. After neural network quantization, the quantitative MVCNN network can be deployed on the mobile device. To use DSP for network inference, we remove the data operations in the MVCNN network that exceed 4-dimension. This is done by converting the 5-dimensional operations involved in the network structure to 4-dimensional operations. For example, we convert a certain operator from the dimension of (3, 12, 1024, 7, 7) to (3, 12, 1024, 7*7). By the DSP environment, the network can be executed on the mobile device quickly, and the speed is close to that of running a normal model on the cloud server with powerful GPU.

## Experiments

### Implementation detail

The experiments of this study use ModelNet40 as the training and testing dataset, containing 9843 training samples and 2468 testing samples, in which 10% of samples from each class are selected to keep the markers during the training process and the rest are disordered as unmarked samples. Each sample consists of 12 views and 12 2D grayscale images of size $224 \times 224$ are transformed into $3 \times 224 \times 224$ RGB images for training in order to facilitate sample enhancement. The sample distribution of all categories is shown in Fig. 6.

The Adam optimizer is used to train the SVCNN and MVCNN model, and there are 60 epochs for each training stage. Every batch includes 1 set of labeled data and 3 sets of unlabeled data. And every epoch traverses all the labeled data once. The initial learning rate is set to $5 \times 10^{-5}$ and the confidence threshold is 0.95.

### The effect of the initial labeled size

In the experiment, we adjust the proportion of labeled data and use 5%, 6%, 7%, 8%, 9%, and 10% of labeled samples separately for training. The goal of this experiment is to observe the influence of the number of labeled samples on the training results. This reflects the cost of implementing our edge computing framework. As shown in Fig. 7, the final accuracy results are improved with the

increase of labeled data. In the following experiment, we choose 10% as the number of labeled samples.

### Comparison with state-of-the-art methods

Figure 8 shows the accuracy variation curve of our proposed method during the training process. We also test the performance of the final model on the unlabeled training samples and the test samples. For the unlabeled training samples, the prediction accuracy is 91.58%. For the test samples of the ModelNet40 dataset, the recognition accuracy is 91.53%.

Moreover, our experiments compare the proposed semi-supervised 3D object recognition method with existing powerful supervised methods, the recent unsupervised 3D shape feature learning methods, and the other semi-supervised methods, respectively. The comparison results are shown in Table 1.

The accuracies of LFD [12], MVCNN [12], MVCNN with high-quality rendering [59], DGCNN [71], PointTnT [36], and PointStack [37] under supervised learning are 75.5%, 89.9%, 95.5%, 92.2%, 92.6% and 93.3% respectively. It can be seen that the 91.53% classification accuracy of our method surpasses even a portion of the supervised learning methods and is comparable to that of DGCNN [71]. It is worth noting that from the report of MVCNN with high-quality rendering [59], the high-quality rendered images can improve the performance significantly. This proves the validity of our offloading decision, which offloads the multi-view rendering to the edge cloud server with the goal of realizing photorealistic rendering.

The unsupervised 3D shape feature learning methods Primitive-GAN [72], FoldingNet (Shapenet) [29] and CrossPoint [32] can achieve classification accuracies of 86.4%, 88.4% and 89.1%. For a fairer comparison, ModelNet40 is used for training instead of the original ShapeNet in the experiments with FoldingNet, and the classification performance dropped to 86.2%. As shown in Table 1, the performance of our method outperforms the state-of-the-art unsupervised methods.

Both FoldingNet [29] and LFD [12] are used as semi-supervised algorithms trained on ModelNet40 using 10% feature data and obtained 76.2% and 60.8% accuracy correspondingly. The most recent semi-supervised methods are the deep co-training method [38] and OSSSL (open-set semi-supervised learning) [40], which achieve the accuracy of 89.0% and 85.5%. The experimental results show that with 10% initial data, our method can obtain better results than the other semi-supervised learning methods.

### Ablation study

#### *Impact of augmentation strategies*

Since augmentation strategies play an important role in the FixMatch algorithm, we conduct an ablation study of data augmentation strategies. We try a combination of other enhancement strategies (e.g., strong enhancement in generating manual labels and weak enhancement in predicting enhancement methods) and find that the test accuracy decreases to some extent. The specific experimental results are shown in Table 2. From the reults, we can see that the strongly-augmented strategy improves the accuracy significantly when we use weakly-augmented strategy during the manual label generation stage. However, when there is no augmentation during the manual label generation stage, the effect of the strongly-augmented strategy is not impressive. This means that the combination of strongly-augmented and weakly–augmented strategy is very important, since this avoids the overfitting of the model.

We use the RandAugment method for image augmentation. For each image, we randomly select some augmentation methods to transform the image. The random augmentation and cropping operations are contained in the strongly-augmented process, and we test the importance of these two operations by varying the predicted augmentation method while maintaining the manual label generation method as weakly-augmented. The results in Table 3 show that the performance of the model is degraded by removing any of these two operations. From the results, we can see that there is no formal distinction between the effect of cropping and random augmentation.

#### *The effect of SVCNN*

To show the effectiveness of the SVCNN stage, we remove the SVCNN stage and compare the accuracy of the learned model with our two-stage training strategy. The comparison results are shown in Table 4, and the prediction accuracy is decreased by 2.15% without the SVCNN stage.

#### *View consistency loss*

To verify the effectiveness of the view consistency loss on the unlabeled samples, we measure the performance of the learned model by removing the corresponding loss. The comparison results are shown in Table 5, and the prediction accuracy is decreased by 0.73% without the view consistent loss.

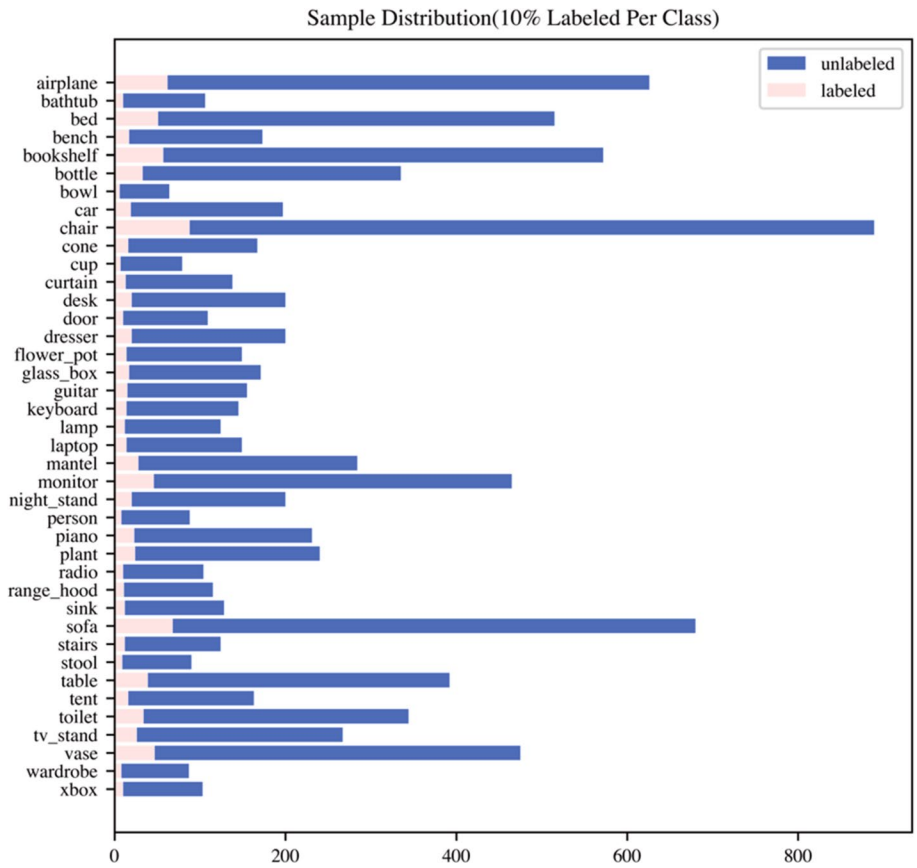


**Fig. 6** The sample distribution of all categories

***The effect of the lightweight architecture***

In contrast to the results of more complex models such as VGG11 [61], we compare the accuracies of different CNN architectures. In this experiment, we set the SVCNN network as VGG11 and ShuffleNet respectively, and the accuracy of the final models is shown in Table 6. The results show that more complex networks lead to higher recognition accuracy.

We then deploy the above two models in a mobile device, which has a Qualcomm Snapdragon 865 Plus mobile platform. In this experiment, we run them in CPU, GPU, and DSP modes respectively. The comparison of the running speed is shown in Table 7 and it can be found that the network Shufflenet runs much faster than the network VGG11 in all three modes. And the former can run about 624 times per second in DSP mode, which is about 6.7 times faster than the latter. The experiment results show that our method is suitable for the edge computing framework, which can be executed efficiently in the mobile terminal.

***The quantization-aware training***

We quantize the final model by quantization aware training and found that the accuracy of the quantized network decreased by 0.48% in comparison as shown in Table 8. The quantitative perception training is trained for 20 epochs, and the initial value of the learning rate is set to $5 \times 10^{-5}$, which is divided by 10 every 5 epochs. Other settings are the same as the training of the MVCNN network. We also compare the performance of the model quantized by the post-training quantization algorithm. The results show that the quantization aware training technology can reduce the quantization error effectively.

**Complexity Analysis.** We measure the timing of the four steps of our edge computing framework. The first step uploads the 3D data to the edge cloud server. After compression, the average size of 3D data in the ModelNet40 is $0.9M$, which takes about 18ms for data transmission under a 5G network. The cloud-based rendering spends around 9ms on the server with 12 NVIDIA GTX 2080Ti graphics cards. To send the 12 rendered images with $224 \times 224$ back to the mobile device (the size is about $0.24M$), it takes around 5ms. Based on our previous experiment, the timings of executing the neural network on the mobile device by CPU, GPU, and DSP mode are 25.4ms, 12.9ms and 1.6ms respectively. Thus,

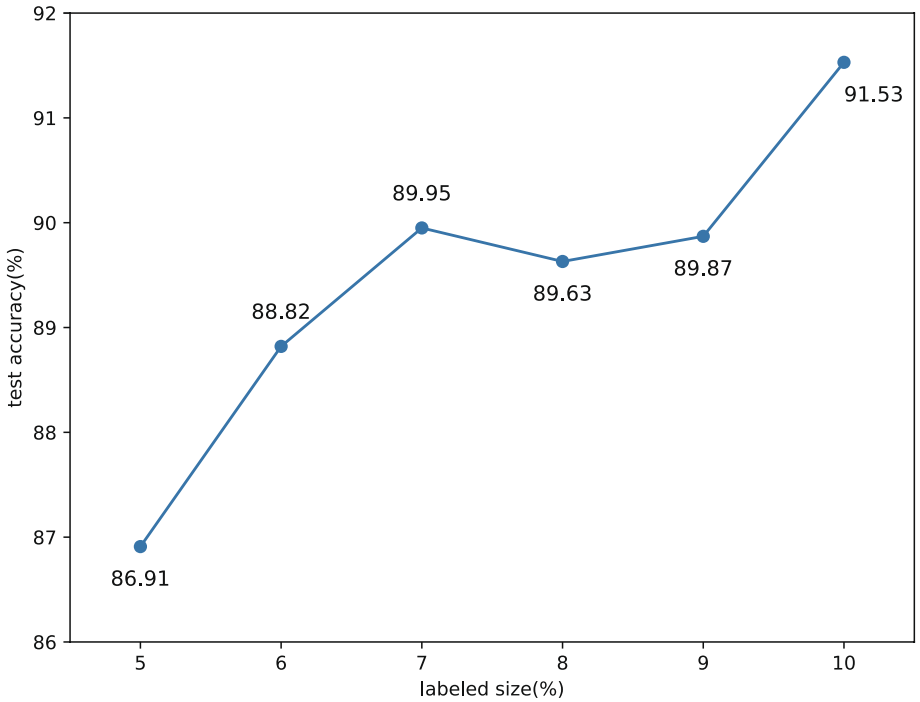


**Fig. 7** The effect of the initial labeled size

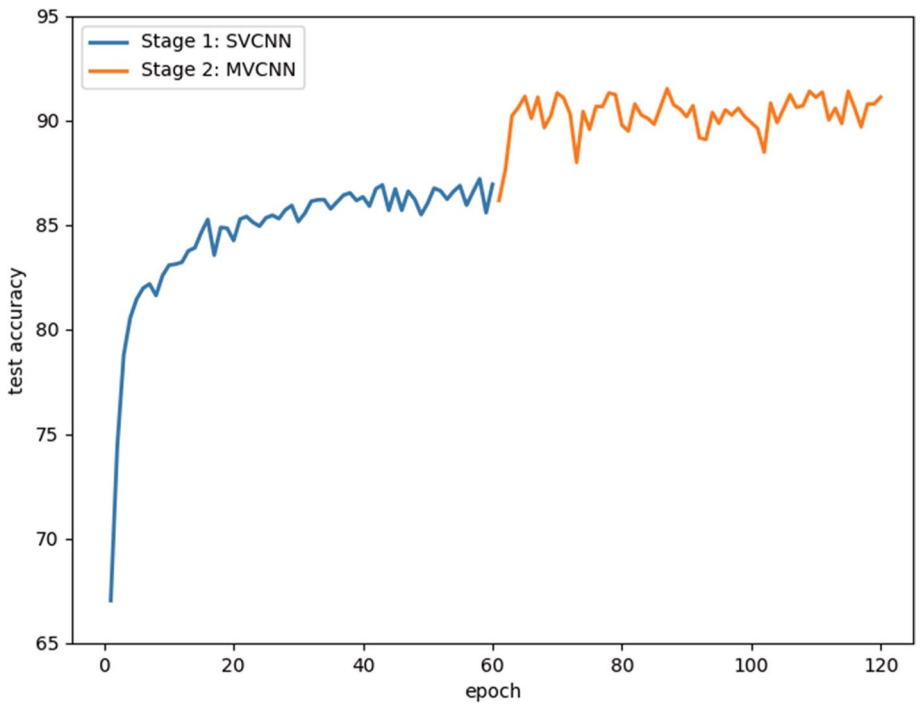


**Fig. 8** The accuracy variation curve

**Table 1** Comparison of Different Methods on ModelNet40

| Type | Methods | Accuracy |
|---|---|---|
| Supervised | LFD [12] | 75.5% |
| | MVCNN [12] | 89.9% |
| | MVCNN (high-quality) [59] | 95.5% |
| | DGCNN [71] | 92.2% |
| | Point-TnT [36] | 92.6% |
| | PointStack [37] | 93.3% |
| Unsupervised | Primitive-GAN [72] | 86.4% |
| | FoldingNet(ModelNet40) [29] | 86.2% |
| | FoldingNet(ShapeNet) [29] | 88.4% |
| | CrossPoint [32] | 89.1% |
| Semi-supervised | LFD [12] | 60.8% |
| | FoldingNet [29] | 76.2% |
| | OSSSL [40] | 85.5% |
| | Co-training [38] | 89.0% |
| | Ours | 91.5% |

**Table 2** Accuracy under Different Augmentation Strategies

| Manual label generation method | Prediction Method | Accuracy (%) |
|---|---|---|
| No augmented | Weakly-augmented | 88.13 |
| No augmented | Strongly-augmented | 88.86 |
| Weakly-augmented | Weakly-augmented | 89.18 |
| Weakly-augmented | Strongly-augmented | 91.53 |

**Table 3** Accuracy under Different Strong Augmentation Methods

| Manual label generation method | Prediction Method | Accuracy (%) |
|---|---|---|
| Weakly-augmented | cropping | 90.19 |
| Weakly-augmented | random augmentation | 90.03 |
| Weakly-augmented | cropping + random augmentation | 91.53 |

**Table 4** The Effect of SVCNN

| Loss | Accuracy (%) |
|---|---|
| without SVCNN | 89.38 |
| with SVCNN | 91.53 |

the whole processing timing is no more than 60ms. This proves that our edge computing framework can support real-time 3D object recognition.

**Table 5** The Effect of the View Consistent Loss

| Loss | Accuracy (%) |
|---|---|
| without view consistency loss | 90.80 |
| with view consistency loss | 91.53 |

**Table 6** Comparison of Different Architectures

| Architecture | Accuracy (%) |
|---|---|
| VGG11 | 91.53 |
| ShuffleNet | 93.07 |

**Table 7** Speed comparison (infs/s indicates the number of executions per second)

| Model | CPU (infs/s) | GPU (infs/s) | DSP (infs/s) |
|---|---|---|---|
| VGG11 | 7.51069 | 15.9231 | 93.7604 |
| ShuffleNet | 39.402 | 77.8134 | 624.683 |

**Table 8** The Effect of the Quantization-Aware Training

| Quantization Algorithm | Accuracy (%) |
|---|---|
| Without Quantization | 91.53 |
| Post-Training Quantization | 90.48 |
| Quantization-Aware Training | 91.05 |

## Conclusion

In this paper, we propose to use the edge computing technique for high-through 3D object recognition. First, a powerful and efficient framework is created by combining edge computing and 3D object recognition, which consists of a cloud-based rendering stage and a terminal-based recognition stage. Second, a lightweight CNN architecture is integrated into a 3D multi-view learning framework to reduce the complexity of the network. Meanwhile, the quantization-aware training technology is utilized to improve the inference speed on the mobile device further. Third, a novel semi-supervised 3D deep learning method based on Fixmatch is proposed to reduce the cost of implementing the edge computing framework. Experiments show that our method achieves high recognition accuracy and fast inference efficiency, which can be helpful for applications in mobile edge environments.

## Limitation and future work

The success of our method relies on the following assumption: all the testing 3D shapes are shown as isolated objects. However, in practice, the 3D capture data usually have lots of noisy information. Therefore, we need to introduce an attention method to focus on the discriminative region in the future. Another future work will utilize more promising lightweight methods to improve efficiency, such as neural architecture search and knowledge distillation. Moreover, we expect to introduce online learning and lifelong learning into our framework, which can use the captured 3D data from the mobile phone as the training data to enhance the performance of 3D object classifier continuously.

**Acknowledgements**
We sincerely thank the reviewers and the Editor for their valuable suggestions.

**Authors' contributions**
Mofei Song and Qi Guo conceived and designed the study. Mofei Song and Qi Guo performed the simulations. All authors wrote the paper. All authors reviewed and edited the manuscript. All authors read and approved the final manuscript.

**Funding**
This work was supported by National Natural Science Foundation of China (61906036), the Open Research Project of State Key Laboratory of Novel Software Technology (Nanjing University) (KFKT2019B02).

**Availability of data and materials**
The data used to support the finding of this study are available from the corresponding author upon request.

## Declarations

**Competing interests**
The authors declare that they have no competing interests.

**Author details**
[1]The School of Computer Science and Engineering, Southeast University, Nanjing, China. [2]The Key Lab of Computer Network and Information Integration (Ministry of Education), Southeast University, Nanjing, China.

Received: 2 September 2022 Accepted: 2 November 2022
Published online: 10 December 2022

**References**

1. Xu X, Li H, Xu W, Liu Z, Yao L, Dai F (2021) Artificial intelligence for edge service optimization in internet of vehicles: A survey. Tsinghua Sci Technol 27(2):270–287
2. Qi L, Hu C, Zhang X, Khosravi MR, Sharma S, Pang S, Wang T (2020) Privacy-aware data fusion and prediction with spatial-temporal context for smart city industrial environment. IEEE Trans Industr Inform 17(6):4159–4167
3. Tong Z, Ye F, Yan M, Liu H, Basodi S (2021) A survey on algorithms for intelligent computing and smart city applications. Big Data Min Analytics 4(3):155–172
4. Chen Y, Gu W, Xu J, et al (2022) Dynamic task offloading for digital twin-empowered mobile edge computing via deep reinforcement learning. China Commun
5. Huang J, Zhang C, Zhang J (2020) A multi-queue approach of energy efficient task scheduling for sensor hubs. Chin J Electron 29(2):242–247
6. Zhang W, Chen X, Jiang J (2020) A multi-objective optimization method of initial virtual machine fault-tolerant placement for star topological data centers of cloud systems. Tsinghua Sci Technol 26(1):95–111
7. Chen Y, Zhao F, Chen X, Wu Y (2022) Efficient multi-vehicle task offloading for mobile edge computing in 6g networks. IEEE Trans Veh Technol 71(5):4584–4595
8. Hou C, Wu J, Cao B, Fan J (2021) A deep-learning prediction model for imbalanced time series data forecasting. Big Data Min Analytics 4(4):266–278
9. Ioannidou A, Chatzilari E, Nikolopoulos S, Kompatsiaris I (2017) Deep learning advances in computer vision with 3d data: A survey. Acm Comput Surv 50(2):1–38
10. Mirbauer M, Krabec M, Krivanek J, Sikudova E (2022) Survey and evaluation of neural 3d shape classification approaches. IEEE Trans Pattern Anal Mach Intell 44(11):8635–8656
11. Xiao YP, Lai YK, Zhang FL, Li C, Gao L (2020) A survey on deep geometry learning: From a representation perspective. Comput Vis Media 6(2):113–133
12. Su H, Maji S, Kalogerakis E, Learned-Miller EG (2015) Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, Santiago, 7-13 December 2015. IEEE, New York
13. Chen SL, Zheng L, Zhang Y, Sun Z, Xu K (2018) Veram: View-enhanced recurrent attention model for 3d shape classification. IEEE Trans Vis Comput Graph 25:3244–3257
14. Chen Y, Xing H, Ma Z, et al (2022) Cost-efficient edge caching for noma-enabled iot services. China Commun
15. Zhi S, Liu Y, Li X, Guo Y (2018) Toward real-time 3d object recognition: A lightweight volumetric cnn framework using multitask learning. Comput Graph 71:199–207
16. Wu X, Chang J, Lai YK, Yang J, Tian Q (2021) Bispl: Bidirectional self-paced learning for recognition from web data. IEEE Trans Image Process 30:6512–6527
17. Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H, et al (2015) Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1502.03167
18. Mo K, Zhu S, Chang AX, Yi L, Tripathi S, Guibas LJ, Su H (2019) Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, CA, 16-20 June 2019. IEEE, New York
19. Yu F, Liu K, Zhang Y, Zhu C, Xu K (2019) Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, CA, 16-20 June 2019. IEEE, New York
20. Fu H, Jia R, Gao L, Gong M, Zhao B, Maybank S, Tao D (2021) 3d-future: 3d furniture shape with texture. Int J Comput Vision 129(12):3313–3337
21. Cheraghian A, Rahman S, Campbell D, Petersson L (2020) Transductive zero-shot learning for 3d point cloud classification. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision, Colorado, 1-5 March 2020. IEEE, New York
22. Wu Z, Zhang Y, Zeng M, Qin F, Wang Y (2018) Joint analysis of shapes and images via deep domain adaptation. Comput Graph 70:140–147
23. Han Z, Shang M, Liu YS, Zwicker M (2019) View inter-prediction gan: Unsupervised representation learning for 3d shapes by learning global shape memories to support local view predictions. In: Proceedings of the AAAI Conference on artificial intelligence, Hawaii, 27 January–1 February 2019. AAAI, Menlo Park
24. Zhang X, Zhou X, Lin M, Sun J (2018) Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Salt Lake City, UT, 18-22 June 2018. IEEE, New York
25. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Proceedings of advances in neural information processing systems, Nevada, 3-8 December 2012. MIT Press, Cambridge
26. Sohn K, Berthelot D, Carlini N, Zhang Z, Zhang H, Raffel CA, Cubuk ED, Kurakin A, Li CL (2020) Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In: Proceedings of advances in neural information processing systems, virtual, 6-10 December 2020. MIT Press, Cambridge

27. Sharma A, Grau O, Fritz M (2016) Vconv-dae: Deep volumetric shape learning without object labels. In: Proceedings of geometry meets deep learning workshop at european conference on computer vision, Amsterdam, 9 October 2016. Springer, Berlin
28. Wu J, Zhang C, Xue T, Freeman B, Tenenbaum JB (2016) Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In: Proceedings of advances in neural information processing systems, Barcelona, 5-10 December 2016. MIT Press, Cambridge
29. Yang Y, Feng C, Shen Y, Tian D (2018) Foldingnet: Point cloud auto-encoder via deep grid deformation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Salt Lake City, UT, 18-22 June 2018. IEEE, New York
30. Achlioptas P, Diamanti O, Mitliagkas I, Guibas L (2018) Learning representations and generative models for 3d point clouds. In: Proceedings of international conference on machine learning, Stockholm Sweden, 10-15 July 2018. ACM, New York
31. Eckart B, Yuan W, Liu C, Kautz J (2021) Self-supervised learning on 3d point clouds by learning discrete generative models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, virtual, 19-25 June 2021. IEEE, New York
32. Afham M, Dissanayake I, Dissanayake D, Dharmasiri A, Thilakarathna K, Rodrigo R (2022) Crosspoint: Self-supervised cross-modal contrastive learning for 3d point cloud understanding. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Louisiana, 19-24 June 2022. IEEE, New York
33. Qi CR, Su H, Mo K, Guibas LJ (2017a) Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Honolulu, HI, 22-25 July 2017. IEEE, New York
34. Qi CR, Yi L, Su H, Guibas LJ (2017b) Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Proceedings of advances in neural information processing systems, Long Beach, 4 December 2017. MIT Press, Cambridge
35. Liu Y, Fan B, Xiang S, Pan C (2019) Relation-shape convolutional neural network for point cloud analysis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, 16-20 June 2019. IEEE, New York
36. Berg A, Oskarsson M, O'Connor M (2022) Points to patches: Enabling the use of self-attention for 3d shape recognition. arXiv preprint arXiv:2204.03957
37. Wijaya KT, Paek DH, Kong SH (2022) Advanced feature learning on point clouds using multi-resolution features and learnable pooling. arXiv preprint arXiv:2205.09962
38. Song M, Liu Y, Liu XF (2020) Semi-supervised 3d shape recognition via multimodal deep co-training. Comput Graph Forum 39(7):279–289
39. Chen L, Zhang Y, Lin Y, Jiang M, Huang Y, Lei Y (2021) Consistency-based semi-supervised learning for point cloud classification. In: Proceedings of international conference on pattern recognition and artificial intelligence, virtual, 20-22 August 2021. Springer, Berlin
40. Shi X, Xu X, Zhang W, Zhu X, Foo CS, Jia K (2022) Open-set semi-supervised learning for 3d point cloud understanding. arXiv preprint arXiv:2205.01006
41. Bader C, Dingler S, Schwieger V (2021) Pvenet: Point voxel encoder network for real-time classification of lidar point cloud segments. In: Proceedings of international conference on advanced robotics, virtual, 6-10 December 2021. IEEE, New York
42. Li F, Yu X, Ge R, Wang Y, Cui Y, Zhou H (2021) Bcse: Blockchain-based trusted service evaluation model over big data. Big Data Min Analytics 5(1):1–14
43. Sandhu AK (2021) Big data with cloud computing: Discussions and challenges. Big Data Min Analytics 5(1):32–40
44. Huang J, Lv B, Wu Y, Chen Y, Shen X (2022) Dynamic admission control and resource allocation for mobile edge computing enabled small cell network. IEEE Trans Veh Technol 71(2):1964–1973
45. Qi L, Lin W, Zhang X, Dou W, Xu X, Chen J (2022) A correlation graph based approach for personalized and compatible web apis recommendation in mobile app development. IEEE Trans Knowl Data Eng. https://doi.org/10.1109/TKDE.2022.3168611
46. Xu J, Li D, Gu W et al (2022) Uav-assisted task offloading for iot in smart buildings and environment via deep reinforcement learning. Build Environ 222. https://doi.org/10.1016/j.buildenv.2022.109218
47. Li K, Zhao J, Hu J, Chen Y (2022) Dynamic energy efficient task offloading and resource allocation for noma-enabled iot in smart buildings and environment. Build Environ. https://doi.org/10.1016/j.buildenv.2022.109513
48. Wang K (2020) Migration strategy of cloud collaborative computing for delay-sensitive industrial iot applications in the context of intelligent manufacturing. Comput Commu 150:413–420
49. Huang J, Tong Z, Feng Z (2022) Geographical poi recommendation for internet of things: A federated learning approach using matrix factorization. Int J Commun Syst. https://doi.org/10.1002/dac.5161
50. Huang J, Gao H, Wan S et al (2023) Aoi-aware energy control and computation offloading for industrial iot. Future Gener Comput Syst 139:29–37
51. Chen Y, Gu W, Li K (2022) Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning. Int J Commun Syst. https://doi.org/10.1002/dac.5154
52. Chen Y, Liu Z, Zhang Y et al (2021) Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things. IEEE Trans Industr Inform 17(7):4925–4934
53. Bi R, Liu Q, Ren J, Tan G (2020) Utility aware offloading for mobile-edge computing. Tsinghua Sci Technol 26(2):239–250
54. Wu Y, Zhang L, Berretti S, Wan S (2022) Medical image encryption by content-aware dna computing for secure healthcare. IEEE Trans Industr Inform. https://doi.org/10.1109/TII.2022.3194590
55. Wu Y, Guo H, Chakraborty C, Khosravi M, Berretti S, Wan S (2022) Edge computing driven low-light image dynamic enhancement for object detection. IEEE Trans Netw Sci Eng. https://doi.org/10.1109/TNSE.2022.3151502
56. Shi G, Wu Y, Liu J, Wan S, Wang W, Lu T (2022) Incremental few-shot semantic segmentation via embedding adaptive-update and hyper-class representation. arXiv preprint arXiv:2207.12964
57. Kim D, Lee S, Lee H, Cho S (2008) A distance-based compression of 3d meshes for mobile devices. IEEE Trans Consum Electron 54(3):1398–1405
58. Deng Y, Ni Y, Li Z, Mu S, Zhang W (2017) Toward real-time ray tracing: A survey on hardware acceleration and microarchitecture techniques. ACM Comput Surv 50(4):1–41
59. Su JC, Gadelha M, Wang R, Maji S (2018) A deeper look at 3d shape classifiers. In: Proceedings of the european conference on computer vision, Munich, 8–14 September 2018. Springer, Berlin
60. Fu H, Cohen-Or D, Dror G, Sheffer A (2008) Upright orientation of man-made objects. In: Proceedings of special interest group on computer graphics and interactive techniques conference, Los Angeles, 11-15 August 2008. ACM, New York
61. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. Ithaca, NY. arXiv preprint https://arxiv.org/abs/1409.1556
62. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Las Vegas, 27-30 June 2016
63. Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. arXiv preprint arXiv:1602.07360
64. Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Honolulu, HI, 22-25 July 2017. IEEE, New York
65. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861
66. Cubuk ED, Zoph B, Shlens J, Le QV (2020) Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, virtual, 14-19 June 2020. IEEE, New York
67. Kurakin A, Raffel C, Berthelot D, Cubuk ED, Zhang H, Sohn K, Carlini N (2020) Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In: Proceedings of international conference on learning representations, virtual, 26 April-1 May 2020. Ithaca, NY
68. Johnson R, Zhang T (2013) Accelerating stochastic gradient descent using predictive variance reduction. In: Proceedings of advances in neural

information processing systems, Nevada, 5-10 December 2013. MIT Press, Cambridge
69. Yang J, Shen X, Xing J, Tian X, Li H, Deng B, Huang J, Hua Xs (2019) Quantization networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, 16-20 June 2019. IEEE, New York
70. Tailor SA, Fernandez-Marques J, Lane ND (2020) Degree-quant: Quantization-aware training for graph neural networks. arXiv preprint arXiv:2008.05000
71. Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM (2019) Dynamic graph cnn for learning on point clouds. ACM Trans Graph 38(5):1–12
72. Khan SH, Guo Y, Hayat M, Barnes N (2019) Unsupervised primitive discovery for improved 3d generative modeling. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, Long Beach, 16-20 June 2019. IEEE, New York

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Submit your manuscript to a SpringerOpen journal and benefit from:**

► Convenient online submission
► Rigorous peer review
► Open access: articles freely available online
► High visibility within the field
► Retaining the copyright to your article

**Submit your next manuscript at ► springeropen.com**