BV-CNNs: Binary Volumetric Convolutional Networks for 3D Object Recognition

Conference Paper · July 2017

CITATIONS
0

READS
671

4 authors, including:

Yinjie Lei
Sichuan University
22 PUBLICATIONS **213** CITATIONS
SEE PROFILE

Yulan Guo
National University of Defense Technology
63 PUBLICATIONS **1,008** CITATIONS
SEE PROFILE

BV-CNNs: Binary Volumetric Convolutional Networks for 3D Object Recognition

Chao Ma[1]
machao0408@nudt.edu.cn
Wei An[1]
nudtanwei@tom.com
Yinjie Lei[2]
yinjie@scu.edu.cn
Yulan Guo[13]
yulan.guo@nudt.edu.cn

[1] College of Electronic Science and Engineering, National University of Defense Technology, Changsha, China

[2] College of Electronics and Information Engineering, Sichuan University, Chengdu, China

[3] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China.

Abstract

Though 3D convolutional neural networks (CNNs) have achieved impressive performance for object recognition, they still face challenges in computational and memory cost. In this paper, we propose binary volumetric convolutional neural networks (namely, BV-CNNs) for efficient 3D object recognition. Specially, it transforms the inputs and weights in the network to binary values through binary transformation, then the floating-point arithmetic convolutions are replaced with bitwise operations to reduce the computational and memory cost. Three binary volumetric CNNs are introduced from the traditional CNNs using our BV-CNN approach. Experimental results on the ModelNet and Sydney Urban Objects datasets demonstrated that computational and memory cost can be significantly reduced while achieving a high recognition accuracy comparable to the networks with floating point weights.

1 Introduction

Object recognition is an important topic in 3D computer vision. In recent years, the release of a large amount of 3D data boosts the rapid development of 3D object recognition [6]. A traditional object recognition pipeline usually consists of the extraction and aggregation of hand-crafted features [5, 7, 13, 18, 21, 26] and an off-the-shelf classifier (*e.g.*, SVM). In recent years, deep learning (DL) based approaches have achieved impressive recognition performance (especially 3D CNN) [1, 9, 14]. Therefore, the recent researches on 3D object recognition mainly focus on learning shape representation from a large amount of data using DL methods.

Although great progress has been achieved by DL, there still remain challenges in high computational and memory cost brought by the high dimensionality of data and complicated models. In this paper, we focus on handling this problem by transforming the weights and

inputs of networks to binary values. Thus, the corresponding floating point operands can be estimated by XNOR and bitcounting operations for binary inputs and weights, which is potentially more efficient and memory saving. In 2D computer vision, several existing works [2, 3, 16] have attempted to binarize the weights and the activations in neural networks for recognition tasks, and achieved comparable performance to the state-of-the-arts. However, there is no direct counterpart in 3D object recognition.

We propose a simple yet effective approach to build binary volumetric CNN (BV-CNN) for real-time 3D object recognition. To the best of our knowledge, this work is the first to apply binary network architecture to 3D object recognition. Our approach uses binary transformation to convert the inputs and weights in the networks to binary values and obtains accurate approximations to traditional CNN models. We present the binary variants of several existing 3D CNNs for object recognition, such as VoxNet [14] and LightNet [27]. We also designed two new CNN models, namely VoxNetPlus and its binary variant. Experimental results demonstrate that our approach can achieve comparable recognition performance with less computational and memory cost compared to their floating point counterparts.

This paper is organized as follows. Section 2 gives an overview on 3D object recognition methods, mainly CNN based methods. Section 3 describes the details of our approach. Section 4 evaluates our method. Section 5 finally concludes the paper.

2 Related Work

In this section, we mainly review the methods highly related to our work.

2.1 3D CNN based Methods

Existing DL based 3D object recognition approaches can be divided into five broad categories [11]: i) DL architectures exploiting descriptors extracted from 3D data; ii) DL architectures exploiting RGB-D data; iii) DL architectures exploiting 3D data directly; iv) DL architectures exploiting 2D projections/views of 3D objects; v) DL architectures exploiting hyperspectral data. Since the third category is closely related to our work, we therefore mainly review the methods belonging to that category.

Wu et al. [25] used a convolutional deep belief network to represent geometric 3D shapes as a probability distribution of binary variables on a 3D voxel grid. Maturana and Cherer [14] also employed volumetric occupancy grid representation with 3D CNN for efficient and accurate object recognition. Sedaghat et al. [19] considered category-level object classification task as a multi-task problem and added orientation prediction as an auxiliary task to achieve improved performance. Qi et al. [15] exploited two new volumetric 3D CNNs to combine auxiliary training, anisotropic probing and Network In Network (NIN) structure, leading to further improved performance. Brock et al. [1] took advantage of recent advancements in DNNs, such as inception-style modules [23], batch normalization [12], residual connections [8], and stochastic network depth [10], to build an ensemble of the proposed VRN models. It achieved the state-of-the-art classification accuracy on both the ModelNet10 and ModelNet40 datasets. In addition, Wu et al. [24] present an unsupervised 3D Generative Adversarial Network (3D-GAN) for object recognition.

Despite the progress has been achieved by various 3D CNNs, their architectures are actually complicated. The state-of-the-art method, Voxception-ResNet, has up to 45 layers with 90M parameters and takes 6 days for training. Furthermore, existing networks can

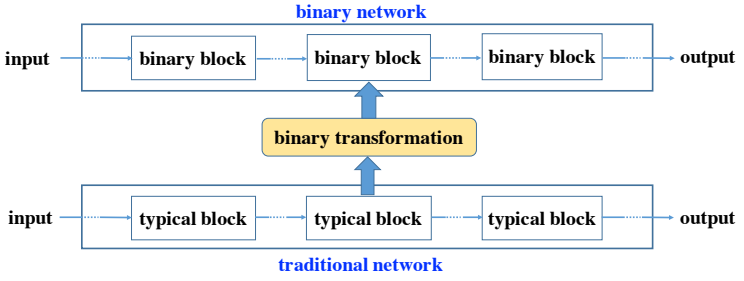

Figure 1: An overview of our approach. We transform the entire network to a binary network by converting the repetitive typical blocks to binary blocks through binary transformation.

only work on low-resolution point clouds, which significantly limits their performance and applications.

Consequently, neural network speedup and model compression are becoming hot topics in the area. Riegler et al. [17] exploited the sparsity of input data to hierarchically partition the space with a set of unbalanced octrees, and built the OctNet with high-resolution inputs. However, it requires special implementation for the network operations on this data structure. Zhi et al. [27] proposed a lightweight 3D CNN (namely, LightNet) to reach a compromise between recognition accuracy and efficiency using multitask learning and a relatively shallower network. Our method is different from these approaches as we do not change the data structure or reduce the depth of a network.

2.2 Network Binarization

BinaryConnect [3] was proposed to restrain the weights to binary values during training while retaining the precision of the stored weights for gradients calculation, which achieved near state-of-the-art performance on the MNIST, CIFAR-10 and SVHN datasets. BinaryNet [2] was proposed to extend BinaryConnect to binarize both weights and activations. XNOR-Net [16] was proposed to find the binary weights and inputs to accurately approximate convolutions and dot product. XNOR-Net was tested on large-scale datasets including ImageNet and outperformed the BinaryNet by a large margin.

Motivated by these approaches above, we propose a binary volumetric network architecture for 3D object recognition. Compared to existing binary networks, several improvements are made to achieve both high compression and high accuracy for volumetric networks. The input binarization method is specifically designed for 3D data and more computationally efficient. Besides, a fully binary network with higher compression rate is proposed, as all inputs and weights are binary. In addition, BN and squared hinge are used for loss calculation, which is suitable for the large variation of variables produced by the last binary layer.

3 The Proposed Approach

3.1 Overview

An overview of our approach is illustrated in Fig. 1. The overall strategy is to convert repetitive typical blocks to binary blocks through binary transformation. Typical block is an integration of several basic sequential network layers, including convolutional/full-connected

layer, batch normalization layer, activation layer, pooling layer and dropout layer. It is usually repeated in a CNN, and is considered as the major parts of the network. Binary transformation includes a series of operations to transform the layers in typical blocks, *e.g.*, input binarization and weight binarization. An entire network can be transformed to a binary network by applying binary transformation to each typical block in the network. Since our method deals with typical blocks, it does not rely on any specific CNN architecture. Therefore, a generalized method is proposed in this paper, which can be expanded to various 3D CNN models. Specially, we take VoxNet and LightNet as examples to produce binary volumetric networks.

3.2 Volumetric Representation

3D shape has different types of representation, such as point cloud, mesh, and volumetric data. Occupancy model has been adopted to convert a 3D shape to volumetric representation by several CNN based methods, such as VoxNet [14]. VoxNet introduces three different occupancy grid models, i.e., binary occupancy grid, density grid, and hit grid. In this work, we adopt binary occupancy grid for two considerations. First, it is shown [14, 19] that the three occupancy grid models achieve comparable experimental performance. Second, binary representation exactly satisfies the input requirement (binary input) of a binary network without any need of further process.

3.3 Binary Transformation

Binary transformation is the key of our method, which transforms typical blocks to binary blocks. Binary transformation includes a series of operations listed below:

- Use the binarization function as activation function in activation layers;
- Insert a batch normalization layer before the binary activation layer;
- Replace the convolutional/full-connected layers with the binary convolutional/full-connected layers;
- Place a pooling layer before binary activation layer when the pooling layer is needed.

3.3.1 Input Binarization

To transform convolutional/full-connected layers to binary layers, inputs should be converted into two-values. Sign function is one of the most common functions used for binarization:

$$x^b = \text{sign}(x) = \begin{cases} 1 & if\ x \geq 0, \\ -1 & otherwise \end{cases} \tag{1}$$

where x^b is a binarized value of real-valued x. Since sign function $sign(x)$ is not derivable at the point of zero, we set the gradient function as $\frac{\partial}{\partial x} = \frac{\partial}{\partial x_b} 1_{|x|\leq 1}$ [2]. That is, the gradient is 1 if x is within the range [-1, 1], otherwise it is set to 0. This binarization can be implemented easily in the activation layers as an activation function. However, the binarization results are sensitive to different data distributions. For example, inputs with all negative/positive values result in "-1/1" without any useful information.

In order to avoid this problem, a batch normalization (BN) [12] layer is inserted before the binary activation layer. The BN layer is often used in DL to normalize the inputs in

each mini-batch, which can shift inputs to values with zero-mean and unit variance to fit the sign function. Besides, the BN layer can accelerate the training as the normalization noise is helpful to regularize the models.

In addition, binary outputs of the activation layers may significantly affect the pooling results. For example, most maximum values in the pools are probably "1" for a max-pooling layer. Therefore, we place pooling layers before binary activation layers.

3.3.2 Weight Binarization

Weight binarization is slightly different from input binarization. First, weights can be learned to adjust themselves to fit the sign function during training, so normalization is not needed. Second, weights have significant effects on the output of each layer. Weight binarization with sign function may cause the loss of much useful information of parameters, leading to performance degradation. Therefore, only sign function is not enough for weight binarization. Following [16], together with sign function, a scaling factor $\alpha \in \mathrm{R}^{+}$ for each convolutional/full-connected layer is used to approximate the floating-point weights:

$$W \approx \alpha W_b \tag{2}$$

where $W_b = \mathrm{sign}(W), \alpha = \frac{1}{n}\|W\|_1$.

3.3.3 Binary Convolutional/Full-connected Layer

In order to better maintain the capability of the traditional convolutional and full-connected layer, their operations are approximately implemented in the binary layer. With I, W_b and α, operations in the traditional convolutional/full-connected layers can be approximated by

$$I \circ W \approx (I \circ W_b)\alpha \tag{3}$$

where I is the input of the convolutional/full-connected layer, and "$\circ$" indicates the corresponding operation in the convolutional/full-connected layers.

With Eq. 3, we can implement the forward pass and backward propagation in a binary convolutional/full-connected layer, as shown in Fig. 2. Operands in these layers can be accelerated through XNOR and bitcounting operations. Weights W are retained for forward pass and gradient calculation during training, and removed once training is finished.

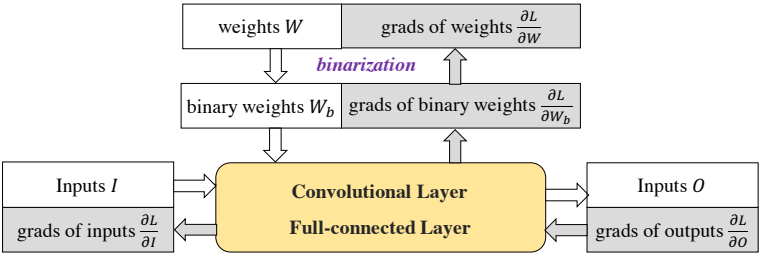

Figure 2: Implementation of the forward pass and backward propagation in the binary convolutional/full-connected layer.

3.4 Binary Volumetric CNNs

Subsequently, the binary variant of an existing 3D network can be obtained based on binary transformation. In this work, we obtained the binary variants of the VoxNet [14] and Light-

Net [27] and designed a deeper modification of VoxNet, namely VoxNetPlus. Compared to VoxNet, an additional convolutional and full-connected layer are added to VoxNetPlus.

3.5 Loss function

The loss function includes both the loss for object recognition and the weight binarization error. We use the squared hinge loss for object recognition, that is:

$$L_c = \sqrt{\max\left(0, 1 - y * \hat{y}\right)} \tag{4}$$

where y and $\hat{y}$ indicate the groundtruth label and the predicted label, respectively.

The approximation error of real-valued weights introduced by weight binarization is defined as

$$J\left(W_b, \alpha\right) = \left\|W - \alpha W_b\right\|^2 \tag{5}$$

With $W_b = \text{sign}\left(W\right), \alpha = \frac{1}{n}\left\|W\right\|_1$, we have

$$J\left(W_b, \alpha\right) = \left\|W\right\|_2^2 - \frac{1}{n}\left\|W\right\|_1^2 \tag{6}$$

Consequently, the overall weight binarization error of the network is

$$L_e = \frac{1}{m}\sum_i^m J_i = \frac{1}{m}\sum_i^m \left\|W\right\|_2^2 - \frac{1}{m}\sum_i^m \frac{1}{n_i}\left\|W\right\|_1^2 \tag{7}$$

where m and n_i are the number of convolutional/full-connected layers and weights in the ith layer, respectively. It can be observed that the two terms of L_e are similar to the $L2$ and $L1$ weight regularization, respectively. Consequently, we can consider error L_e as regularization terms of the network to speedup the trainning process and improve the accuracy.

Finally, the overall loss is

$$L = L_c + \gamma L_e \tag{8}$$

where γ is set to 0.01 in this paper.

4 Experimental Results

Experiments were conducted on the ModelNet and the Sydney Urban Objects datasets to evaluate three original 3D CNN models (i.e., VoxNet, LightNet, and VoxNetPlus) and their binary variants. We also compare our binary VoxNetPlus with the state-of-the-art approaches. Furthermore, we analyzed the improvements of binary 3D CNNs in terms of computational and memory cost.

We trained all the models from scratch. During the training phase, weights were initialized with a uniform distribution within $[-1, 1]$. Since border parts of 3D volumetric data are generally free space or unoccupied, no padding is for binary convolutional layers. All models were trained using Nesterov Momentum [22] with a momentum value of 0.9. Learning rate was initially set to 0.001 and then decreased to 0.0000001 over training epoches. Our method was implemented in Python with Theano[1] and Lasagne[2]. All experiments were conducted on a computer with a Nvidia GTX1080 GPU, an Intel Core i6700k and 16G RAM.

[1] http://deeplearning.net/software/theano
[2] http://lasagne.readthedocs.io

4.1 Dataset

ModelNet [25] contains more than 120,000 CAD models from 662 categories. ModelNet has two subsets with different categories number: ModelNet10 with 10 categories and ModelNet40 with 40 categories. The orientations of all CAD models in ModelNet10 have been aligned along the gravity axis. **Sydney Urban Objects dataset** [4] contains various common urban road objects scanned by LiDAR from a single viewpoint. There are 631 individual scans of objects from 14 categories. As a result, this dataset is more challenging than ModelNet. To evaluate 3D CNNs on these datasets, point clouds were converted to voxel grids with a size of 32^3.

4.2 Evaluation of 3D Volumetric Network Binarization

The results of VoxNet [14], LightNet [27], VoxNetPlus and their binary variants on the ModelNet10, ModelNet40 and Sydney Urban Objects datasets are shown in Table 1. To evaluate the effectiveness of network binarization, we compare each binary 3D CNN with its original network. Similar to [27], we tuned the binary LightNet on ModelNet40 with weights pre-trained on ModelNet10. Similar to [14], the average F1 score weighted by class support on the Sydney Urban Objects dataset is used to evaluate the classification performance.

Network	Size	Classification Accuracy		Average F1 score
		ModelNet10	ModelNet40	Sydney Urban Objects
VoxNet[14]	3.4M	92%	83%	0.72
binary VoxNet	0.1M	90.69%	81.63%	0.68
LightNet[27]	1.1M	93.39%	86.9%	0.76
binary LightNet	40K	92.36%	84.24%	0.72
VoxNetPlus	9M	93.36%	83.91%	0.759
binary VoxNetPlus	0.3M	92.32%	85.47%	0.755

Table 1: Classification results. The overall accuracy on the ModelNet datasets and the weighted average over F1 score on the Sydney Urban Objects dataset are reported. Besides, the size of model parameters on ModelNet40 is also listed.

On the ModelNet10 dataset, the largest performance decrease between binary VoxNet and its original network is 1.31%. However, both binary LightNet and binary VoxNetPlus outperform the traditional VoxNet. The accuracy of the binary VoxNetPlus is only decreased about by 1.04% compared to its original network.

On the ModelNet40 dataset, binary LightNet has the largest decrease of accuracy (about 2.66%) when compared to its original network. This is probably because that training with fine-tuning for binary LightNet does not perform as well as the traditional LightNet. In fact, the binary LightNet achieves better performance than its original network trained from scratch (about 82.9%). Besides, the binary VoxNet has a small decrease of accuracy (about 1.37%) compared to its original network. However, the binary VoxNetPlus even achieves a higher accuracy (about 1.56%) than its original network. That is because, the noise introduced by weight and input binarization during parameters gradient calculation provides a form of regularization. Such regularization can improve the generalization capability of the network like Dropout [20].

Though Sydney Urban Objects dataset is a more challenging dataset for object recognition than the ModelNet datasets, the proposed binary networks still perform well. Specif-

ically, the decreases of the F1 scores are within 0.04. The binary VoxNetPlus achieves a comparable F1 score to its original network, and the difference is almost negligible.

Considering the decline of computational and memory cost, the small decrease in recognition performance is acceptable. It is observed that all of the three binary CNNs need less than 1M memory for model parameters. The binary LightNet just needs 40K memory, which can be applied on devices with limited computational and memory resources. The binary VoxNetPlus achieves comparable accuracy to the traditional LightNet and VoxNetPlus, with 0.3M parameters, which is about 32 times smaller than its original network, and about 4 times smaller than the traditional LightNet.

It is also observed that the binary VoxNetPlus achieves the best approximation performance to its original network, with a slight decrease on the ModelNet10 dataset. It obtains classification accuracy of 82.32% and 85.47% on the ModelNet10 and ModelNet40 datasets, respectively. Its average F1 score on the Sydney Urban Objects dataset is 0.755. It can maintain good performance with less memory requirement on all the three datasets.

4.3 Comparison to the State-of-the-art

We also compared our binary VoxNetPlus with the state-of-the-art approaches, as shown in Table 2. Since we only consider volumetric network frameworks in this work, we do not include multi-view networks. Compared to 3D ShapeNets and VoxNet, our binary VoxNetPlus achieves the best performance on all of the three datasets. On the ModelNet40 dataset, our binary VoxNetPlus improves the accuracy of VoxNet by 2.47% with just one additional convolutional and full-connected layer. However, the model size of our binary VoxNetPlus is about 10 times smaller than VoxNet. Compared to ORION and LightNet, our binary VoxNetPlus obtains comparable accuracy with only 0.3M parameters. Although VRN ensemble achieves the best performance on both ModelNet10 and ModelNet40, it has the most complex network architecture with up to 45 layers and 90M parameters. In contrast, our binary VoxNetPlus is more compact and has a shallower architecture with only 7 layers.

Network	Size	Classification Accuracy		Average F1 score
		ModelNet10	ModelNet40	Sydney Urban Objects
3D ShapeNets [25]	38M	83.54%	77.32%	-
VoxNet [14]	3.4M	92%	83%	0.72
ORION [19]	-	93.8%	-	**0.767**
VRN Ensemble [1]	90M	**97.14%**	**95.54%**	-
LightNet [27]	1.1M	93.39%	86.9%	0.76
binary VoxNetPlus	**0.3M**	92.32%	85.47%	0.755

Table 2: Comparison to the state-of-the-art. "-" indicates that no information is available for the corresponding item in the corresponding paper.

Furthermore, a confusion matrix of category prediction achieved by binary VoxNetPlus on the ModelNet10 dataset is shown in Fig. 3. It is observed that most categories can be accurately predicted. For example, all the bed CAD models were predicted correctly. However, it is challenging for the network to distinguish the dresser from the night stand, the table from the desk. Top three false positive rates are 14%, 13%, 13%, respectively. Examples of false category prediction results are shown in Fig. 4, it is observed that they are difficult to be distinguished in shape even for human eyes.

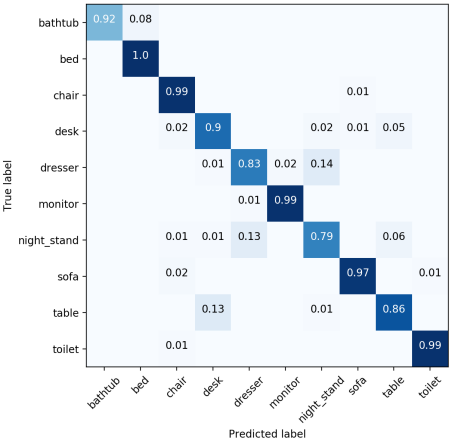

Figure 3: A confusion matrix of category prediction achieved by binary VoxNetPlus on the ModelNet10 dataset.

(a) Dresser (b) Night stand (c) Table (d) Desk

Figure 4: Examples of false category prediction results.

4.4 Computational and Memory Cost

Generally, most computation consumption comes from operations in convolutional layers. Operations in the CNNs consist of convolutions, matrix multiplications and multiplications addition, and the operands units are multiplications and accumulations. For 32-bit CNNs, operands (including multiplications and accumulations) are 32-bit floating point arithmetic. However, For binary CNNs, both the inputs and weights in the convolutional and full-connected layers are constrained to either -1 or 1. As a result, the 32-bit floating point operations are replaced with XNOR and bitcounting operations, which can be accelerated through hardware. Speedup ratio can be determined by the hardware and algorithms (mainly binary-operand kernel) together.

Reducing memory requirement has always been a prior consideration for the algorithms to applications. For the binary 3D CNNs, only a single bit is required for storing a weight parameter. Therefore, a memory smaller than the 32-bit CNN models by 32 times is required to store the weight parameter. In addition, smaller memory size can reduce the memory access, which significantly reduces energy consumption.

5 Conclusion

In this paper, a simple yet effective method is introduced to build binary volumetric CNNs for 3D object recognition. The inputs and weights in the CNNs are transformed to binary values through binary transformation. The binary variants of three traditional 3D CNNs are introduced, including VoxNet, LightNet and VoxNetPlus. Experiments on the ModelNet10,

ModelNet40, and Sydney Urban Objects datasets show that the binary CNNs significantly reduce computational and memory cost without obvious accuracy loss in object recognition.

6 Acknowledgement

This work is supported by the National Natural Science Foundation of China (Grant Nos. 61403265, 61602499, and 61471371), the Science and Technology Plan of Sichuan Province (Grant No. 2015SZ0226), the National Postdoctoral Program for Innovative Talents (No. BX201600172), and China Postdoctoral Science Foundation.

References

[1] Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.

[2] Matthieu Courbariaux and Yoshua Bengio. BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

[3] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. BinaryConnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems*, pages 3123–3131, 2015.

[4] Mark De Deuge, Alastair Quadros, Calvin Hung, and Bertrand Douillard. Unsupervised feature learning for classification of outdoor 3D scans. In *Australasian Conference on Robitics and Automation*, volume 2, 2013.

[5] Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. Rotational projection statistics for 3D local surface description and object recognition. *International Journal of Computer Vision*, 105(1):63–86, 2013.

[6] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, and Jianwei Wan. 3D object recognition in cluttered scenes with local surface features: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014.

[7] Yulan Guo, Mohammed Bennamoun, Ferdous Sohel, Min Lu, Jianwei Wan, and Ngai Ming Kwok. A comprehensive performance evaluation of 3D local feature descriptors. *International Journal of Computer Vision*, 116(1):66, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[9] Vishakh Hegde and Reza Zadeh. FusionNet: 3D object classification using multiple data representations. *arXiv preprint arXiv:1607.05695*, 2016.

[10] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pages 646–661, 2016.

[11] Anastasia Ioannidou, Elisavet Chatzilari, Spiros Nikolopoulos, and Ioannis Kompatsiaris. Deep learning advances in computer vision with 3D data: A survey. *ACM Computing Surveys*, 50(2):20, 2017.

[12] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[13] Andrew Johnson and Martial Hebert. Surface matching for object recognition in complex 3D scenes. *Image and Vision Computing*, 1998.

[14] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 922–928, 2015.

[15] Charles R Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5648–5656, 2016.

[16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. XNOR-Net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542, 2016.

[17] Gernot Riegler, Ali Osman Ulusoys, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. *arXiv preprint arXiv:1611.05009*, 2016.

[18] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[19] Nima Sedaghat, Mohammadreza Zolfaghari, and Thomas Brox. Orientation-boosted voxel nets for 3D object recognition. *CoRR, vol. abs/1604.03351*, 2016.

[20] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[21] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer Graphics Forum*, volume 28, pages 1383–1392, 2009.

[22] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning*, pages 1139–1147, 2013.

[23] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[24] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016.

[25] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[26] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 373–380, 2009.

[27] Shuaifeng Zhi, Yongxiang Liu, Xiang Li, and Yulan Guo. LightNet: A lightweight 3D convolutional neural network for real-time 3D object recognition. *Eurographics Workshop on 3D Object Retrieval*, 2017.