Neural Processing Letters (2019) 49:1369–1379
https://doi.org/10.1007/s11063-018-9878-5

A Comparison: Different DCNN Models for Intelligent Object Detection in Remote Sensing Images

Peng Ding[1,2,3,4] · Ye Zhang[1,3] · Ping Jia[1,3] · Xu-ling Chang[1,3]

Published online: 30 June 2018

Abstract

In recent years, deep learning especially deep convolutional neural networks (DCNN) has made great progress. Many researchers take advantage of different DCNN models to do object detection in remote sensing. Different DCNN models have different advantages and disadvantages. But in the field of remote sensing, many scholars usually do comparison between DCNN models and traditional machine learning. In this paper, we compare different state-of-the-art DCNN models mainly over two publicly available remote sensing datasets—airplane dataset and car dataset. Such comparison can provide guidance for related researchers. Besides,we provide suggestions for fine-tuning different DCNN models. Moreover, for DCNN models including fully connected layers, we provide a method to save storage space.

Keywords Object detection · Deep convolution neural networks · Deep learning · Remote sensing images

✉ Peng Ding
dingpeng14@mails.ucas.ac.cn

✉ Ye Zhang
zhangye@ciomp.ac.cn

Ping Jia
jiap@ciomp.ac.cn

Xu-ling Chang
xuling.chang@live.com

1 Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China

2 University of Chinese Academy of Sciences, Beijing 100049, China

3 Key Laboratory of Airborne Optical Imaging and Measurement, Chinese Academy of Sciences, Changchun 130033, China

4 Fraunhofer Institute for Computer Graphics Research & TU Darmstadt, 64283 Darmstadt, Germany

1 Introduction

As we all know, the process of hierarchical and multi-stage computing features is more abundant. Convolutional neural networks (CNN), a biologically inspired hierarchical and shift-invariant model for pattern recognition, is a successful model at just such a process. In recent years, deep learning [1–3], particularly deep convolutional neural networks [3, 4] (DCNN) has made great progress. The first CNN model is LeNet which proposed by Lecun in 1986 [5], but then fell out of fashion with the advent of support vector machines (SVMs). However, in recent years, due to the development of large public image repositories, DCNN models have gotten great success in large-scale visual recognition. In 2012, AlexNet won first place in the Imagenet competition. Compared with LeNet, dropout [6] and ReLu [7] was adopt in AlexNet. In 2014, GooleNet [8] became the first in the Imagenet competition. In GoogleNet, $1 * 1$ convolution layers are used to reduce dimension. It adopts more convolutions, deeper layers than AlexNet. In 2015 and 2016, ResNet [9] won first place in the Imagenet competition, its special feature is the design of the "bottleneck" form of the block. Some other famous DCNN models such as VGG-Nets [10], they have all achieved good results than traditional machine learning.

Recently, deep convolutional neural networks have demonstrated excellent performance on various visual tasks. At the same time, in the field of remote sensing, many researchers take advantage of different DCNN models to do object detection. When labeled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. So, many researchers do detection by fine-tuning DCNN models. Chen do vehicle detection in satellite remote sensing images by fine-tuning hybrid deep convolutional neural networks (HDCNN) [11], Zhang do forest Fire Detection by fine-tuning AlexNet [12]. Luiz G do Forest Species Recognition by fine-tuning LeNet [13]. Marco do land use classification by fine-tuning CaffeNet [14]. But in the field of remote sensing, many scholars usually do comparison between DCNN models and traditional machine learning. In fact, comparison between DCNN models has more practical significance. In this paper, we compare different state-of-the-art DCNN models mainly over two publicly available remote sensing datasets—airplane dataset and car dataset. Besides, we take advantages of Faster RCNN (one kind of region-based convolutional network) as the frame.

The main contributions of this paper are as follows: (1) different DCNN models are introduced into intelligent object detection in remote sensing field; (2) we investigate different DCNN models for intelligent object detection in remote sensing field and provides a detailed comparison; (3) we provide suggestions for fine-tuning different DCNN models; (4) for DCNN models including fully connected layers, we provide a method to save storage space.

The rest of the paper is organized as follows: In the next section, we describe basic principle of convolutional neural networks and Faster RCNN. In Sect. 3, describe how to fine-tune DCNN models. Analysis and comparison of experimental results are shown in Sect. 4. Finally, some discussions conclude this paper.

2 Related work

2.1 The Principle of Convolutional Neural Network

Convolutional neural networks including convolution layers and feature pooling layers and fully connected layers [15]. Each component has its own special value and significance.

Convolution Layers At a convolution layer, the previous layer's feature maps X_i^{l-1} are convolved with learnable kernels k_{ij}^l and add a trainable bias parameter b_j, then put through the activation function $f()$ to form the output feature map. The process can be expressed as:

$$X_j^l = f\left(\sum_{i \in M_j} X_i^{l-1} * k_{ij}^l + b_j^l\right) \tag{1}$$

where M_j represents a selection of input maps. Usually, we choose ReLu which called the rectifier activation function as the activation function, for it works better than logistic sigmoid and hyperbolic tangent.

Feature Pooling Layer This layer treats each feature map separately. Usually, we call this layer as subsampling layer which produces down-sampled versions of the input maps. That means the number of input maps and output maps is the same, but the output maps will be smaller. The results are robust to small variations in the location of features in the previous layer. The process can be expressed as:

$$X_j^l = down\left(X_j^{l-1}\right) \tag{2}$$

where $down(\cdot)$ denotes a down-sampling operation. By down-sampling operation we reduce the size of the input by summarizing neurons from a small spatial neighborhood [16].

Fully Connected Layers After several convolutional and subsampling layers, the high-level reasoning in the Neural Networks is done via fully connected layers. Neurons in a fully connected layer have full connections to all activations in the previous layer. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

Training is performed by using back-propagation method [5] to minimize the aberration between the ideal output and the actual output of convolutional neural network. Usually, for detecting, a convolutional neural network is followed by a classification module.

2.2 The Principle of Faster RCNN

Faster RCNN is a particularly successful method for general object detection. It consists of two components: Region Proposal Network (RPN) and Fast RCNN. RPN is a fully convolutional neural network for generating candidate regions (proposals). These proposals would be fed into Fast RCNN. Fast RCNN uses the proposals to do further detection. RPN stage and Fast stage are trained independently, but they share convolutional layers which is helpful to improve accuracy.

2.2.1 Fast RCNN

The whole image first processed by a DCNN model and a conv feature map would be produced. Then, for each object proposals, ROI-pooling layer extracts a fixed-length feature vector from the feature map. Last, each fixed-length feature vector is fed into a sequence of

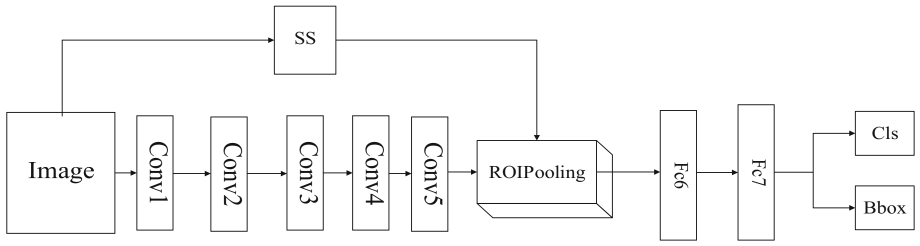

Fig. 1 The flow chart of Fast RCNN

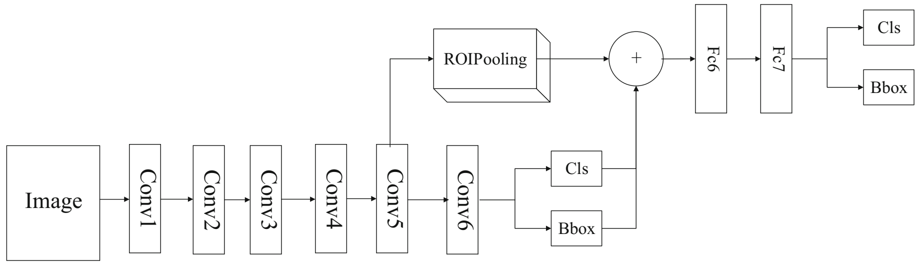

Fig. 2 The flow chart of Faster RCNN

fully connected layers which branch into two sibling output layers: classification layer (Cls) and bbox-regression layer (Bbox). The flow chart of Fast RCNN can be seen in Fig. 1.

Fast RCNN achieves near real time rates, but region proposal remain as the test-time computational bottleneck. Because region proposals usually proposed by selective search algorithm. Selective search algorithm is an order of magnitude slower, about 2 s per image in a CPU implementation. So RPN was proposed in Faster RCNN for computing proposals.

2.2.2 RPN

RPN is a fully convolutional neural network for generating candidate regions. In RPN stage, the convolution layers of a pre-trained DCNN model are followed by a 3*3 convolutional layer. Then, two 1*1 convolutional layers are added to this 3*3 convolutional layer for classification and regression. This 3*3 convolutional layer corresponds to mapping a large spatial window and receptive field in the input image to a low-dimensional feature vector at a center stride. The most important work in RPN is anchors are introduced, at each sliding location, anchors are proposed. For multi-scale predictions, multi-scale anchors are needed. For training RPNs, an anchor that has an IoU overlap higher than 0.7 with any ground-truth box is assigned as a positive label. At the same time, we assign a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes. RPN stage and Fast RCNN stage can be trained independently, but they share convolutional layers, the flow chart of Faster RCNN can be seen in Fig. 2.

Faster RCNN provides us a good way to use our own small database. Because we can fine-tune the whole networks directly according to our database.

3 Comparison of Different DCNN Models in Fast RCNN

3.1 Fine-Tuning Output Parameter for a Particular Dataset

When labeled data is scarce, there are two options for us to choose. One way is to use unsupervised pre-training, followed by supervised fine-tuning. The other one is that supervised pre-training model on a large auxiliary dataset, followed by domain-specific fine-tuning on the dataset [17]. The later one is adopted by many researchers, we also adopt the later one. First of all, the original Faster RCNN model is designed for 21 class of the object. Here, we just detect airplanes or cars. That means we just have two categories. So we should set up categories as two: one is our target, the other one is the background. Every target has a tuple (x, y, w, h), for a ground-truth bounding-box regression, we should set up categories as $4 * (K + 1)$, where, K is the number of category.

3.2 Fine-Tuning Different DCNN Models in Faster RCNN

Different DCNN models are somewhat different when fine-tuning. For some DCNN models including fully connected layers such as ZF-Net [18], CaffeNet,AleNet and so on, ROI-pooling layer should be placed before fully connected layers. For DCNN models without fully connected networks such as NIN [19], GoogleNet and ResNet, ROI-pooling layer placed before last convolutional layers. For example: for ResNet, all layers of conv5_x and up are adopted for each region, playing the roles of fc layers. Besides, for SqueezeNet [20] we should add regularization to conv1 and conv2 for ensuring network convergence. It is worth noting that, for ResNet50 and ResNet101, memory consumption is very big. So the BN layers should be fixed during fine-tuning. We fix the BN layers mainly for reducing memory consumption in Faster R-CNN training. For further reduce memory consumption during training, we should set a smaller batch-size in Fast RCNN stage and choose smaller proposals.

3.3 Convolutional Neural Networks Used in Sub-network

For DCNN models including fully connected layers, the two following layers after ROI-pooling layer are fully connected layers, and the two fully connected layers require a large number of parameters. ROI-pooling layer extracts a fixed-length feature vector from the feature map. So, the output of ROI-pooling layer is fixed. We can see all layers after ROI-pooling as a sub-network. This sub-network consists of fully connected layers. However, the process of hierarchical and multi-stage computing features is more abundant. Thus, we adopt two convolutional networks instead of these two fully connected layers. By our methods, the required memory of the final model would be reduced a lot. The details of our method are illustrated in Fig. 3.

To ensure the hierarchical structure, the receptive fields of new conv1 and new conv2 are set as 3 and 1 respectively. The number of channels is set as 512 for feature calculation effectively.

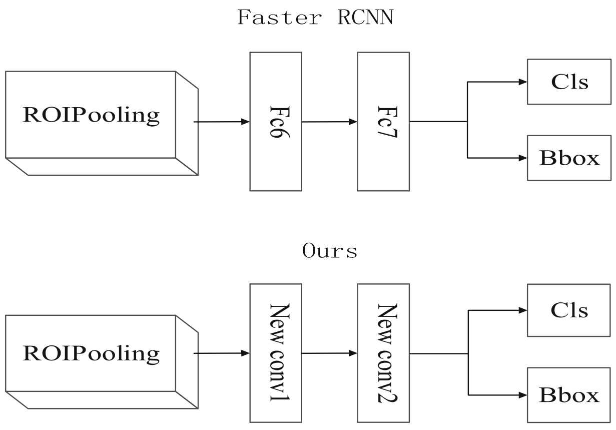

Fig. 3 The flow chart of our approach

4 Experiment and Discussion

All experiments in this section are performed on the airplane dataset and car dataset. These datasets are proposed by the university of Chinese academic of science. All the experiments are performed on the GTX-Tianx (Intel(R) Core(TM) i7-6850K CPU @ 3.6 GHz) with Caffe [21], matlab2014b and some other software such as Opencv3.0. In this paper, we choose different DCNN models as the baseline. We need to compare their performance. Faster RCNN was run for 40 k SGD [22] iteration. The learning rate is set as 0.001 with a momentum [23] as 0.0005. DCNN models including ZF, CaffeNet, Alexnet, GoogleNet, NIN, HDCNN, SqueezeNet, VGG16, VGG_M_1024, VGG19, ResNet18, ResNet50, ResNet101.

4.1 Qualitative Analysis

During this section, we first approve fine-tuning pre-training model on a large auxiliary dataset is feasible. We randomly select three kinds DCNN models for object detection. Test results for aircraft and automobiles are shown below. The aircraft dataset includes 1000 optical remote sensing images, with about 7000 targets. Each image is about $1200 * 600$ pixels, the aircraft occupies a pixel size ranging from $40 * 40$ to $120 * 120$. Some test results of randomly selected DCNN models are shown in Fig. 4.

From these images, it can be noted that by fine-tuning DCNN models, we are able to obtain very effective results. The car dataset includes 500 optical remote sensing images, with about 7000 targets. Each image's sizes is about $1200 * 600$ pixels, the aircraft occupies a pixel size ranging from $20 * 20$ to $60 * 60$. Some test results of randomly selected DCNN models are shown in Fig. 5. Compared with plane, the contrast to the background is less pronounced which increases the difficulty of detection.

We can find that many cars are not detected, that's because for smaller targets, the results of most DCNN models are not very ideal. This is determined by the characteristics of the data set. The size of the aircraft is bigger and the contrast to the background is more pronounced. So, the detection of planes is easier than cars.

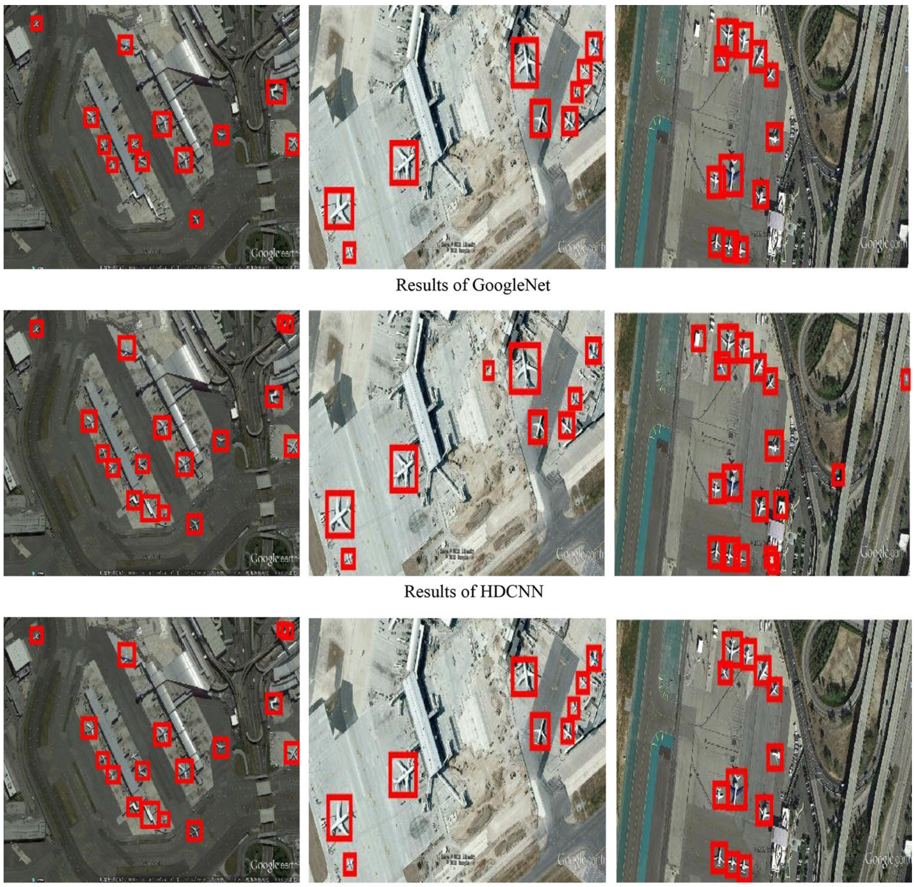

Fig. 4 Some results of randomly selected DCNN models for airplane dataset

4.2 Quantitative Analysis

In Sect. 4.1, we can see directly from pictures that by our approach, we can detect object by fine-tuning DCNN models. But our analysis in 4.1 lacks quantitative evidence. So we would provide quantitative analysis. In this paper, we adapt the two commonly used objective criteria which are average precision (AP) and recall rates to evaluate the performance.

$$average\text{-}precision = \frac{True\ \ positive}{True\ \ positive\ + True\ negative} \tag{3}$$

$$average\text{-}precision = \frac{True\ \ positive}{True\ \ positive\ + True\ negative} \tag{4}$$

Test time is another important indicator for the real-time requirements in engineering. We also adopt it as an index. Moreover, the required storage space is also very important in practical application, we also adopt it as an index. The specific comparison is shown in Table 1.

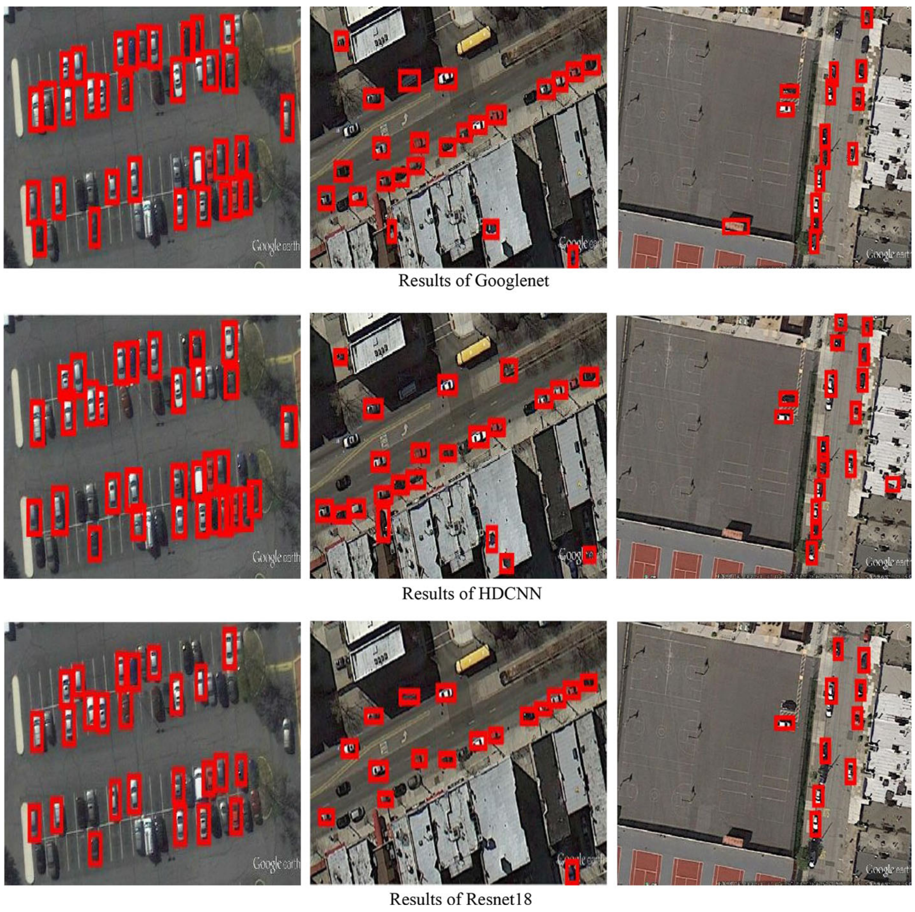

Fig. 5 Some results of randomly selected DCNN models for car dataset

We find that for larger objects, most DCNN models can achieve very good results, for smaller targets, the results of most DCNN models are not very ideal. In terms of model size, models including fully connected networks often require more storage space such as VGG16 and VGG19. Besides, for the accuracy of the test, very deep DCNN models such VGG16 and Resnet50 usually get better results than non-very deep DCNN models such Alexnet and ZF-Net. Sometimes, overfitting will happen in very deep DCNN models such as VGG19. To ensure that it will not be overfitting, the deeper the network, the more data it needs. From the experimental results, we can see that the results of different network structures are different. It is not simply that the deeper the network, the better the results. For a given small data set, simply increasing the depth may not necessarily improve precision. The structure of the network also has a greater impact on precision. For example, VGG16 has only 16 layers, and its detection effect is similar to Resnet50. The P–R curve of different DCNN models can be seen in Fig. 6.

The iteration time used in training is an important index, we adopt the four-stages Faster RCNN. The average iteration time for each stage is as Table 2 (For airplane dataset).

Table 1 Specific comparison of different DCNN models

Base-network	AP(car/plane)	Recall(car/plane)	Test-time	Memory
ZF-Net	64.71/84.12	65.23/86.12	0.070	233.2 M
CaffeNet	55.98/77.54	57.73/81.98	0.056	227.6 M
Alexnet	59.68/78.63	59.77/81.04	0.056	227.6 M
GoogleNet	66.35/87.50	68.65/89.58	0.076	24 M
NIN	68.27.85.28	69.33/87.57	0.062	26.3 M
HDCNN	60.77/84.12	60.92/85.96	0.067	227.6 M
SqueezeNet	64.07/85.49	65.65/87.21	0.055	2.9 M
VGG16	75.21/88.71	78.43/88.74	0.104	512 M
VGG_M_1024	63.05/81.69	64.71/85.57	0.064	344.9 M
VGG19	71.86/88.70	73.16/90.81	0.117	558.4 M
ResNet18	67.24/87.63	71.36/90.12	0.091	42.7 M
ResNet50	75.92/89.74	78.17/93.02	0.416	94.4 M
ResNet101	76.13/89.78	79.23/93.08	0.527	170.6 M

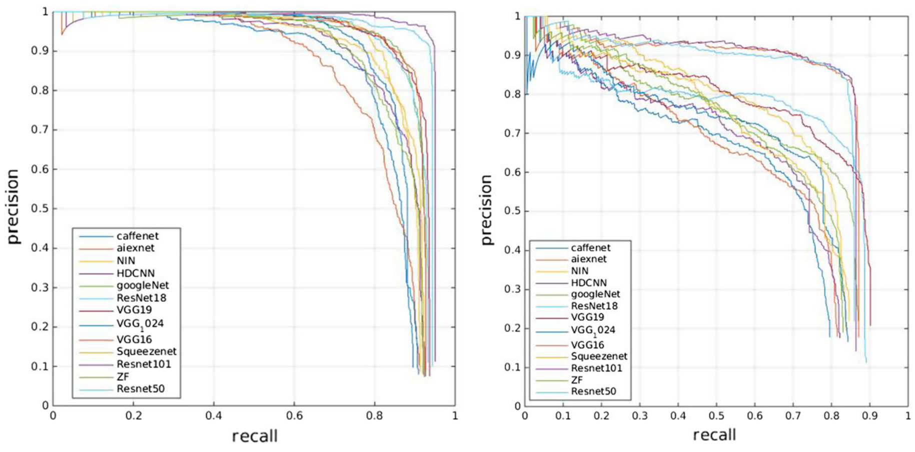

Fig. 6 The P–R curve of different DCNN models for airplane datasets (left) and car datasets (right)

Through Table 2, we can know that very deep DCNN models need more time to train a good result, but for Resnet101, training time in Fast RCNN stage is not particularly long compared with ResNet18 and ResNet50. That is because we just choose 300 proposals to do further detection which is suggest in original paper.

4.3 Analysis Convolutional Neural Networks Used in Sub-network

The purpose of using CNNs in sub-network is not to improve AP and Recall but to save memory. As we analysis in Sects. 4.1 and 4.2, DCNN models including fully connectted layers need much more storage. By using our method, the memory needed reduced a lot which increases its portability. The detailed comparation can be seen in Table 3. (Memory1 and Memory2 respectively represent fully connected layers used in sub-network and convolutional neural networks used in sub-network.)

Table 2 Average iteration time for each stage of different DCNN models

Base-Network	RPN stage1	Fast stage1	RPN stage2	Fast stage2
ZF	0.089 s	0.191 s	0.071 s	0.140 s
CaffeNet	0.064 s	0.130 s	0.058 s	0.118 s
Alexnet	0.064 s	0.130 s	0.056 s	0.117 s
GoogleNet	0.094 s	0.182 s	0.072 s	0.131 s
NIN	0.068 s	0.146 s	0.058 s	0.124 s
HDCNN	0.074 s	0.151 s	0.061 s	0.124 s
SqueezeNet	0.083 s	0.183 s	0.072 s	0.131 s
VGG16	0.256 s	0.568 s	0.161 s	0.329 s
VGG_M_1024	0.074 s	0.161 s	0.065 s	0.133 s
VGG19	0.304 s	0.665 s	0.170 s	0.361 s
ResNet18	0.081	0.226	0.078	0.198
ResNet50	0.172	0.421	0.169	0.385
ResNet101	0.202	0.206	0.194	0.131

Table 3 Detailed comparison about convolutional neural networks used in sub-network

Base-network	Memory1	Memory2
ZF	233.2 M	29.1 M
CaffeNet	227.6 M	23.5 M
Alexnet	227.6 M	23.5 M
HDCNN	227.6 M	23.5 M
VGG16	512 M	77.8 M
VGG_M_1024	344.9 M	45 M
VGG19	558.4 M	124.2 M

5 Conclusions

DCNNs has been applied to object detection in the field of remote sensing.But in the field of remote sensing, many scholars usually do comparison between DCNNs and traditional machine learning. In this paper, we compare different state-of-the-art DCNN models mainly over two publicly available remote sensing datasets. We compare them in many aspects such as accuracy, recall, the required storage space and so on. Besides, we provide some suggestion about how to fine-tune different deep models. Moreover, we provide a method to save storage space for DCNN models including fully connected layers.

Acknowledgements This work was supported by the project of National Science Fund for Distinguished Young Scholars of China (Grant No. 60902067) and the Key Science-Technology Project of Jilin Province (Grant No. 11ZDGG001).

References

1. Hong C, Yu J, Wan J, Tao D, Wang M (2015) multimodal deep autoencoder for human pose recovery. IEEE Trans Image Process 24(12):5659–5670

2. Hong C, Yu J, Chen X (2013) Image-based 3D human pose recovery with locality sensitive sparse retrieval. In: Systems, man and cybernetics, pp 2103–2108
3. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. https://doi.org/10.1038/nature14539
4. Schmidhuber J (2015) Deep learning in neural networks: an overview. Neural Netw 61:85–117. https://doi.org/10.1016/j.neunet.2014.09.003
5. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
6. Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
7. Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: AISTATS, pp 315–323
8. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich (2015) A Going deeper with convolutions. In: Computer vision and pattern recognition, pp 1–9
9. He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. In: Computer vision and pattern recognition, pp 770–778
10. Zisserman KSAA (2015) Very deep convolutional networks for large-scale image recognition. In: International conference on learning representations (ICLR)
11. Chen X, Xiang S, Liu CL, Pan CH (2014) Vehicle detection in satellite images by hybrid deep convolutional neural networks. IEEE Geosci Remote Sens Lett 11(10):1797–1801. https://doi.org/10.1109/LGRS.2014.2309695
12. Zhang QJ, Xu JL, Xu L, Guo HF (2016) Deep convolutional neural networks for forest fire detection. In: Kim YH (ed) Proceedings of the 2016 international forum on management, education and information technology application, vol 47. Advances in social science education and humanities research, pp 568–575. Atlantis Press, Paris
13. Hafemann LG, Oliveira LS, Cavalin PR (2014) Forest species recognition using deep convolutional neural networks. In: International conference on pattern recognition, pp 1103–1107
14. Castelluccio M, Poggi G, Sansone C, Verdoliva L (2015) Land use classification in remote sensing images by convolutional neural networks. In: Computer science
15. Lecun Y, Kavukcuoglu K, Farabet C (2010) Convolutional networks and applications in vision. In: International symposium on circuits and systems, pp 253–256
16. Scherer D, Muller A, Behnke S (2010) Evaluation of pooling operations in convolutional architectures for object recognition. In: International conference on artificial neural networks
17. Girshick R, Donahue J, Darrell T, Malik J, Ieee (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE conference on computer vision and pattern recognition, pp 580–587. IEEE, New York. https://doi.org/10.1109/cvpr.2014.81
18. Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Computer vision-ECCV 2014: 13th European conference, Zurich, Switzerland, September 6–12, 2014, proceedings, part I. Springer, Cham, pp 818–833. https://doi.org/10.1007/978-3-319-10590-1_53
19. Lin M, Chen Q, Yan S (2013) Network in network. arxiv:1312.4400
20. 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:1602.07360
21. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: convolutional architecture for fast feature embedding. In: ACM multimedia, pp 675–678
22. Bottou L (2010) Large-scale machine learning with stochastic gradient descent. In: Lechevallier Y, Saporta G (eds) Proceedings of COMPSTAT'2010: 19th International Conference on computational statistics, Paris France, August 22–27, 2010 Keynote, invited and contributed papers, pp 177–186. Physica-Verlag HD, Heidelberg. https://doi.org/10.1007/978-3-7908-2604-3_16
23. Bottou L (2012) Stochastic gradient descent tricks. In: Montavon G, Orr GB, Müller K-R (eds) Neural networks: tricks of the trade, 2nd edn. Springer, Berlin, pp 421–436. https://doi.org/10.1007/978-3-642-35289-8_25