

Heterogeneous Convolutional Neural Networks for Visual Recognition

Xiangyang Li, Luis Herranz, and Shuqiang Jiang^(✉)

Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences,
Beijing 100190, China
{xiangyang.li,luis.herranz}@vipl.ict.ac.cn, sqjiang@ict.ac.cn

Abstract. Deep convolutional neural networks (CNNs) have shown impressive performance for image recognition when trained over large scale datasets such as ImageNet. CNNs can extract hierarchical features layer by layer starting from raw pixel values, and representations from the highest layers can be efficiently adapted to other visual recognition tasks. In this paper, we propose heterogeneous deep convolutional neural networks (HCNNs) to learn features from different CNN models. Features obtained from heterogeneous CNNs have different characteristics since each network has a different architecture with different depth and the design of receptive fields. HCNNs use a combination network (i.e. another multi-layer neural network) to learn higher level features combining those obtained from heterogeneous base neural networks. The combination network is also trained and thus can better integrate features obtained from heterogeneous base networks. To better understand the combination mechanism, we backpropagate the optimal output and evaluate how the network selects features from each model. The results show that the combination network can automatically leverage the different descriptive abilities of the original models, achieving comparable performance on many challenging benchmarks.

Keywords: Deep learning · Object classification · Scene recognition · Convolutional neural networks · Heterogeneous convolutional neural networks

1 Introduction

Convolutional neural networks (CNNs) [10] trained using supervised backpropagation were successful in certain controlled tasks such as digit recognition. In recent years, CNNs have shown their real potential in large-scale visual recognition [7, 9, 11, 14, 17, 18, 20, 26]. The first breakthrough was made by Krizhevsky *et al.* [9], showing that CNNs can improve the state-of-the-art accuracy in large scale object recognition with a large margin. The availability of large datasets such as ImageNet [13] or Places [26] and high-performance computing systems such as GPUs are the main factors responsible for this dramatic improvement.

CNNs extract features from the raw pixels in a feed-forward basis, where the output of a layer is the input of the next layer. The convolution operation of the input image with the specific filters introduces local invariance. The filters in CNNs are learned from data and contribute greatly to obtain very discriminative image features. Another key of the success of CNNs is their ability to learn complex high level features by building increasingly abstract representations hierarchically.

A number of works have been focused on how to improve the performance of CNNs. The first strategy is to train and test CNNs with multiple scales rather than using only a fixed size [7, 14]. Sermanet *et al.* [14] proposed OverFeat, which shows how a multi-scale sliding window can be efficiently implemented within CNNs. It uses the same fixed size approach proposed by Krizhevsky *et al.* [9] when training CNNs, but turns to multi-scale during test. He *et al.* [7] proposed spatial pyramid pooling convolutional networks (SPP-net), which adds a spatial pyramid pooling layer between the last convolutional layer and the first fully-connected layer. Another strategy to improve recognition accuracy is to increase the depth and width of the architecture [11, 17, 18]. Lin *et al.* [11] propose Network in Network (NIN), which aims to enhance model discriminability for local patches within the receptive field by building micro neural networks. In NIN, convolutional filters are replaced by micro networks which increase the depth of the global network. Simonyan *et al.* [17] propose models with very deep layers. Their work demonstrated that the representation depth is beneficial for the classification accuracy. Similarly, Szegedy *et al.* [18] propose GoogLeNet which contains 22 layers. In their work, the inception module is the key component, which not only increases the width but also computes multi-scale features.

Previous methods use more complex architectures to learn discriminative features. Another way to increase the performance is averaging the results (i.e. probabilities for each class) obtained from multiple instances of the same network, each of them trained under different settings [3, 6, 9, 16, 17, 21]. In this way the variance of the joint model decreases, with an improvement in the recognition accuracy. However, in this case, since the architecture is fixed, the features extracted from each instance have similar properties.

In contrast to combining decisions or probabilities, we can also combine the mid-level features extracted from CNNs. Agrawal *et al.* [1] suggest that units in CNNs may be classified into grandmother cells (GMC), which have strong response to specific object classes, or part of distributed codes in which a single response is not discriminative, but a group of responses jointly can be very discriminative. Features extracted from different architectures have different characteristics due to different number of layers and filter designs (related with different receptive fields). Thus, these different features can have complementary characteristics. For example, Fig. 5 shows the response of units from two CNNs with very different architectures (corresponding to those in Fig. 2) to different images. Responses from each of these heterogeneous networks focus on different parts of the original images. To recognize a lizard, one unit in one CNN focuses on the middle part of its body, while one of the other CNN is sensitive to its

tail. Features from different models are learned jointly with the rest of its units, so each model has its specific characteristics. The combination of features from heterogeneous networks can make the resulting features more discriminative.

In this paper, we propose heterogeneous convolutional neural networks (HCNNs) as an efficient approach to combine features obtained from heterogeneous architectures using a multi-layer combination network that replaces the top layers of the individual networks and combines them (see Fig. 1). The combination mechanism is also trained from data, learning additional network parameters which can better integrate features obtained from base networks with different architectures. The advantages of HCNNs are:

- First, higher level features resulting from the combination of features from heterogeneous CNNs lead to richer and more discriminative feature representations. HCNN can automatically leverage different descriptive abilities of the original models, achieving better performance on many challenging benchmarks than single network.
- Second, the combination network strategy is flexible and it can benefit from large datasets and fine tuning to specific tasks.

The rest of the paper is organized as follows. Section 2 describes the proposed method. Section 3 describes the experiments and Sect. 4 concludes the paper.

2 Heterogeneous Convolutional Neural Networks

Our motivation is to combine the different capabilities of two (or more) base networks. The architecture and the training process of a HCNNs is illustrated in Fig. 1 for two base networks.

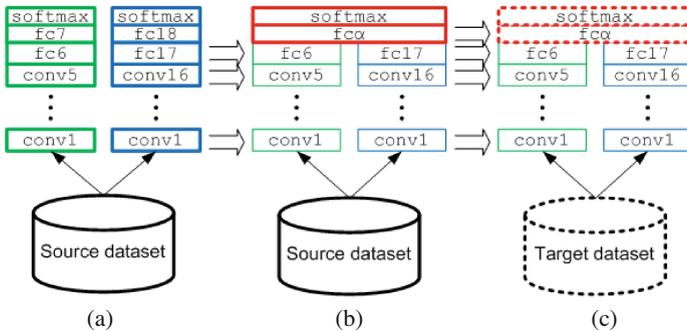


Fig. 1. Heterogeneous convolutional neural networks (HCNNs): (a) pre-training base networks (or reuse available networks), (b) pre-training the combination network, and (c) fine tuning. The example shows two base networks with different architectures, and a two layer combination network.

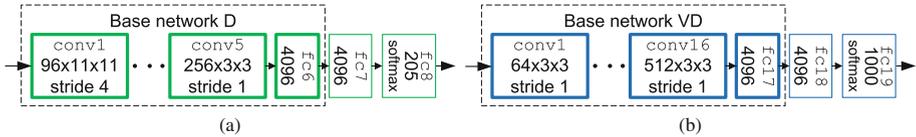


Fig. 2. Architectures used in the experiments: (a) deep network D (8 layers), and (b) very deep network VD (19 layers).

2.1 Base Networks

The base networks are the components we want to combine. Although any number of networks can be combined, we focus on the case of two networks. We prefer that both networks have different architectures, since different architectures may provide different expressive capabilities due to different depth and receptive field sizes.

Figure 2 shows two examples of heterogeneous CNN architectures that we will use as base networks. For simplicity, in the rest of the paper we will refer to this particular two architectures as deep (D) and very deep (VD) networks. D has 8 layers and the filters in D have certain sizes, such as 11×11 , 5×5 and 3×3 , while VD has 19 layers and its convolutional filters have the same size 3×3 . The convolution of the input images or feature maps with the learned kernels can capture local invariance, and pooling can increase the invariance to small deformations as well as small shifts. Following the same process layer by layer, the units in high layers have progressively larger receptive field sizes, so they can extract more complex features.

2.2 Combination Network

In order to learn complementary features from base networks, we combine they (two or more) by concatenating the features from the base networks and stack another network (see Fig. 1b). It is a way to learn new features from base networks and can also retain their strength. This combination network may have multiple layers, typically fully connected. The parameters of the combination network are learned from data, so the joint network can freely combine features from any of the networks. Note that this contrasts with simply averaging the probabilities after the softmax classifier. Our method is more flexible, but also has a large number of parameters that must be learned properly. The architecture of the combination network should be designed carefully to maximize the benefit from the available models.

A first question that arises is where should the features be concatenated. In order to keep as much as possible of the abstract knowledge learned by the pre-trained base networks, we would like to connect them at a high layer. However, high layer units are too specialized to the original task, so reusing them may decrease the generalization to a new task or a new setting [25]. Thus, we drop several top layers (see Fig. 1a), so the base networks can “forget” too specific

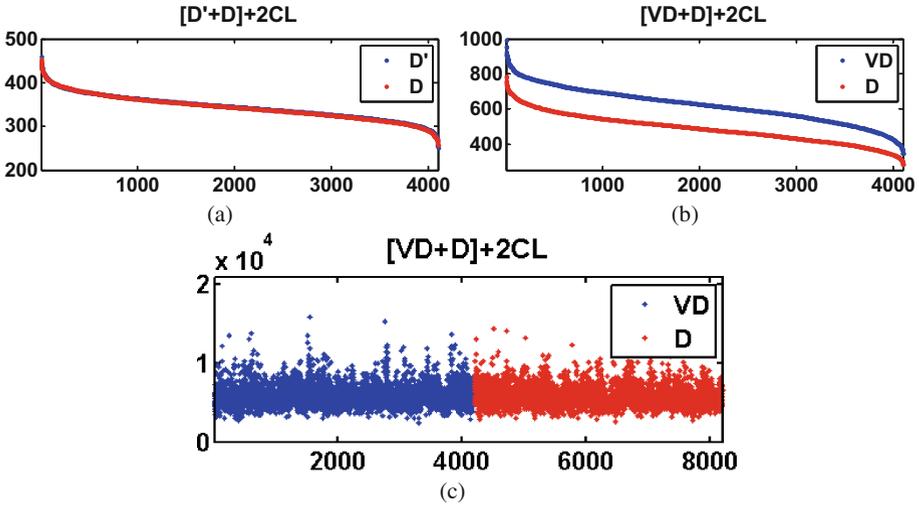


Fig. 3. Average distribution of activations in the concatenation layer, obtained by maximizing the posterior probability (in descending order) when combining (a) homogeneous architectures and (b) heterogeneous architectures, and (c) activations in the concatenated layer for heterogeneous architectures. Best viewed in color.

knowledge, which can be learned again in the combination network from a combination of features from all the base networks. To decide how many layers to remove, a reasonable rule seems to remove as many top layers from each base network as the number of layers in the combination network. In this way, the number of layers from each of the input layers to the output (i.e. the depth) is not changed. In our standard architecture we remove two layers (softmax and top fully connected) and stack two fully connected layers (softmax and fully connected). The number and size of the layers in the combination network is also related with the number of parameters to be learned, and thus has impact over the generalization capability of the network and training cost.

2.3 Training the Network

A HCNN is trained in three steps (see Fig. 1). First the base networks are pre-trained with a suitable large scale dataset such as ImageNet. This step requires far more computational resources and training time than the rest. The training cost can be significantly reduced by reusing models already available, and combining their architectures. In this paper, we reuse publicly available models for fast implementation.

The second step is training the combination network, which typically uses backpropagation over a large scale dataset. The third step is fine tuning the network. This step is optional and only necessary when the target dataset is not the same dataset used from training. In principle we only fine tune the last layer

or the layers of the combination network, although it could be extended to layers in the base networks or even the whole joint network, but that would increase the training cost significantly.

There is another implicit trade-off with the training cost: the more layers to be trained, the higher the computational cost and the training time. Training a combination network is much faster than training a deep network from scratch, because the number of layers and parameters to be learned is significantly small (which is also less prone to overfitting).

2.4 Understanding the Combination Mechanism

In order to understand how the combination network selects features from the base networks we follow the visualization methods proposed in [5, 15], but stopping at the concatenation layer. Instead of computing the input image that maximizes the posterior probability (i.e. softmax) for a given class, we back-propagate the optimal output for each class and check the corresponding input value. Then we average the values across all the classes and sort them in descending order. The results are shown in Fig. 3, where we separately plot the values corresponding to units from each base network.

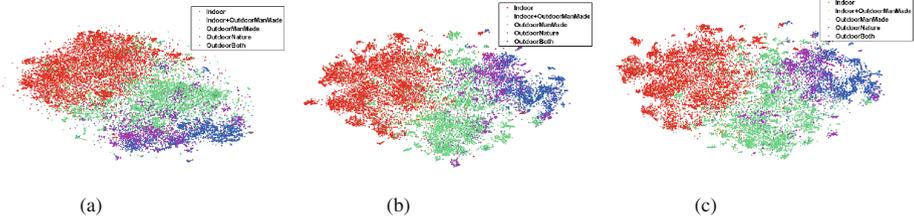


Fig. 4. t-SNE visualization of deep features for SUN397: (a) fc_7 of D, (b) fc_{18} of VD, and (c) fc_α of the joint network [D,VD]+2CL (see Sect. 4 for details). Best viewed in color.

Focusing on SUN397, Fig. 3a shows the case where two homogeneous architectures are combined. As both base networks have the same architecture and learned similar units, the distributions for both networks are similar, showing no particular preference for either of them. In contrast, if we combine two different heterogeneous networks (denoted as D and VD in Fig. 3b) there is a clear dominance of the best performing network. In order to check that this is not caused by different scales in the activations from each base network, we checked the average activations at the same fully connected layer (obtained by forward processing each image and averaging the activations), which shows that both networks are in the same scale (see Fig. 3c). This suggests that the combination network tends to rely more on VD than on D, as expected, because VD is deeper, has better performance and its features are likely to be more discriminative. In

contrast, in the case of base networks with the same architecture, the filters learned by each base network capture similar patterns, so Fig. 3a shows almost the same distribution for each base network.

In contrast to averaging, the combination network can generate deep features by taking the output of a fully connected layer. In order to get some insight about how different groups of classes are represented by these deep features, we visualize the 4096-dimensional features obtained from the last fully connected layer (before the softmax) using a two-dimensional t-SNE embedding [19]. Figure 4 shows the heterogeneous case, comparing features from both individual networks and the combined one for SUN397. The difference between the visualizations of D and VD is relatively clear, with VD visualization showing the groups of classes somewhat more separated. The difference between VD and the HCNNS is more subtle. This agrees with the previous conclusion, suggesting that units from VD are selected with higher weights by the combination network, and thus the deep features are more similar to those obtained from VD. However, some discriminative units from D are also integrated in the joint network, which leads to a more discriminative space with a gain of 7.53% over VD in this particular case (see Table 3).

3 Experiments

3.1 Settings

In our experiments we rely on the Caffe framework [8], and the available models¹. We downloaded the reference BVLC_caffenet [8] and the VGG [17] models (see Fig. 2) trained over ILSVRC2012, which are publicly available in Caffe format. In addition to the network D, we trained from scratch another network D' with the same architecture for comparing with homogeneous architectures. Due to the time limit, we did not train another very deep network VD' from scratch as it will cost a lot of time. So here we only use D and D' to show the performance of homogeneous architectures. In most of the experiments we fine tuned only the combination network for target datasets other than ILSVRC2012.

We compare two combination methods: averaging and the proposed combination network. For convenience we introduce the following notation to compactly describe a particular variation: $[A,B]+k\text{CL}$ denotes a HCNN which concatenates the features from two base networks A and B with a combination network with k layers. Similarly, $[A,B]+\text{avg}$ denotes the model resulting from averaging the predictions of A and B. In our experiments we only used the central crop of size 227×227 of the image previously resized to 256×256 , and compare with other architectures in the same setting.

3.2 Combination Network Architecture

We evaluated combination networks consisting of stacked fully connected layers on top of the concatenated features of the two base networks. We refer to them

¹ <http://caffe.berkeleyvision.org>.

as combination layers (CLs). The most important parameters are the number of CLs and the size. In our standard architecture (see Fig. 5a), there are two CLs, where the top one is the softmax. The only free parameter is the size of the other layer $fc\alpha$. We compared three sizes for the number, and the performance is very similar (see Table. 1), so we keep the two layer network with a size of 4096 for the rest of the experiments.

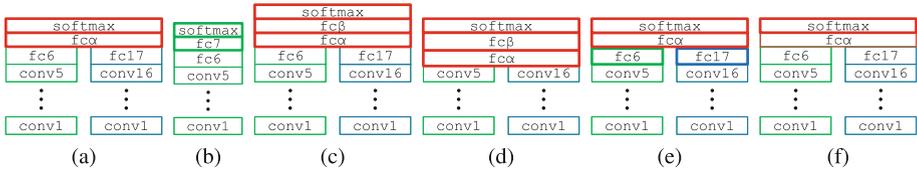


Fig. 5. Variations of architectures and fine tuning strategies in the experiments: (a) standard architecture [V,VD]+2CL with 2 CL fine tuned, (b) single network D with the two top-layers fine tuned, (c) three-layer CL, (d) three-layer CL with one additional top-layer removed from the base networks, (e) fine tuning down to the first top layer of the base networks, and (f) only fine tune the top layer (i.e. retrain the softmax).

We also compared with a deeper combination network with three layers. In this case we considered two variations. The first one adds one additional layer between $fc\alpha$ and the softmax (see Fig. 5c). This increases the total depth of the network, and increases the parameters to be learned and the training cost. However, the accuracy decreases slightly. We also compared with a second variation with three layers, but with another fully connected layer from the base networks removed, so the two base networks are concatenated after the last convolutional layer (the total depth is the same as in the standard architecture).

Table 1. Comparison of different architectures for the combination network.

Architecture	Layer size		Accuracy (%)	
	$fc\alpha$	$fc\beta$	Top 1	Top 5
2CL (Fig. 5a)	4096	-	69.87	89.67
	5192	-	70.10	89.50
	6144	-	70.08	89.44
3 CL (Fig. 5c)	4096	6144	69.43	89.02
3 CL (Fig. 5d)	4096	6144	66.73	87.24

3.3 Object Recognition Performance

We first evaluate the performance for object recognition, and compare the proposed method with averaging the outputs of the two networks. The results are shown in Table 2. In general we can observe that averaging performs well for homogeneous base networks, but the gain is more limited for heterogeneous networks (except for Caltech 101). In contrast, HCNNs seem to achieve higher gains when the architectures are heterogeneous, while achieving marginal or even negative gains for homogeneous ones.

Table 2. Accuracy (%) for object recognition.

Method	ILSVRC 2012 (top 1)		ILSVRC 2012 (top 5)		Caltech101		Caltech256	
	Acc	Gain	Acc	Gain	Acc	Gain	Acc	Gain
D	57.47	-	80.41	-	88.38	-	66.33	-
D'	57.45	-	80.34	-	88.31	-	66.06	-
VD ^a	68.56	-	88.53	-	90.32	-	76.87	-
[D,D'] ₊ avg	59.25	1.78	81.68	1.27	89.18	0.80	68.39	2.06
[D,VD] ₊ avg	69.10	0.54	88.15	-0.38	92.55	2.23	77.32	0.45
[D,D'] ₊ 2CL	58.25	0.78	80.75	0.34	88.55	0.17	65.81	-0.52
[D,VD] ₊ 2CL	70.10	1.54	89.50	0.97	92.82	2.50	79.07	2.20
VGG ^b	75.6		92.5		-		-	
GoogLeNet ^c	-		89.93 (92.11)		-		-	
GoogLeNet ^d	-		91.91 (93.33)		-		-	
GoogLeNet ^e	67.44 (68.44)		88.24 (89.01)		-		-	
[2]	-		86.9		91.4		77.61	

^aCaffe implementation of VGG using only the central crop and one scale [17].

^bUsing multiple scales, dense evaluation and 7 networks. ILSVRC submission [17].

^cILSVRC submission. 1 crop (144 crops)[18].

^dILSVRC submission. 7 networks, 1 crop (144 crops)[18].

^eCaffe implementation of GoogLeNet, 1 crop (10 crops)[23].

3.4 Scene Recognition Performance

We also evaluated the performance for scene recognition using the MIT67 [12] and SUN397 [24] datasets (see Table 3). Similarly to the results for object recognition, we observe that averaging is a better solution for combining homogeneous architectures, while a combination network is more suitable for heterogeneous ones. The gain is particularly significant for SUN397, with a very competitive performance of 57.30%, 7.53% better than the best performing base network. As networks trained on scene-oriented datasets are more suitable for scene classification [26], we also report the performance of the networks trained on Places, donated as D_p and VD_p . The results verify the effectiveness of the proposed HCNNS.

Table 3. Accuracy (%) for scene recognition.

Method	MIT67		SUN397	
	Acc	Gain	Acc	Gain
D	59.34	-	46.41	-
D'	59.52	-	45.85	-
VD	69.70	-	49.41	-
[D,D']+avg	62.68	3.16	47.77	1.36
[D,VD]+avg	70.75	2.05	52.28	2.87
[D,D']+2CL	58.27	-1.25	45.54	-0.87
[D,VD]+2CL	72.31	2.61	57.30	7.53
SFV_ImageNet [4]	72.86	-	54.4	-
Meta_Objects [22]	78.90	-	58.11	-
SFV [4]	79.01	-	61.72	-
[Dp,VDp]+2CL	80.52	-	64.91	-

Table 4. Accuracy (%) for different fine tuning strategies for [D,VD]+2CL.

Architecture	Accuracy (%)			
	Cal101	Cal256	MIT67	SUN397
Only softmax	91.89	78.07	79.93	64.08
Softmax, FC α	92.82	79.07	80.52	64.91
Softmax, FC α	93.18	79.50	81.06	65.32
FC6, FC17				

3.5 Impact of Fine Tuning

Fine tuning can improve the recognition accuracy by refining the parameters of the network. We analyze its impact on the recognition accuracy for our standard architecture [D,VD]+2CL, with one, two and three layers refined (see Table 4). First we consider retraining only the softmax layer (see Fig. 5f). We see that the performance improves compared to individual networks. However retraining the two layers of the combination network (see Fig. 5a) improves the performance significantly around 1–3%. Further propagating fine tuning to the next layer, which actually corresponds to layers in the base networks (see Fig. 5e), can still improve the performance. The lower the layer, the less related to the task (less specific), so the gain is marginal. However, the training cost increases significantly.

4 Conclusion

In this paper we studied the unusual case of combining deep networks with heterogeneous architectures, and proposed HCNNs which combine them by

concatenating high-layer features and stacking another (combination) network. The proposed method achieves significant performance on many challenging benchmarks. To understand the combination mechanism, we backpropagate the optimal output and evaluate how HCNNs select features from each model. The results show that it can automatically leverage the different descriptive abilities of the original models. Compared with the conventional model averaging method (i.e. ensemble), HCNNs can generate new features from different base networks not just combining probabilities. This allows them to work better when the features are heterogeneous, while averaging often works better for homogeneous (similar model, so the ensemble reduces the variance). The combination network strategy is flexible and it can benefit from large datasets and fine tuning to specific tasks. In this work, we only implement two existing models to show the effectiveness of HCNNs. In the future, more heterogeneous models can be included in the proposed method, where the models can either be publicly available models or trained from scratch.

Acknowledgements. This work was supported in part by the National Basic Research 973 Program of China under Grant No. 2012CB316400, the National Natural Science Foundation of China under Grant Nos. 61532018, 61322212 and 61550110505, the National High Technology Research and Development 863 Program of China under Grant No. 2014AA015202, Beijing Science And Technology Project under Grant No. D161100001816001. This work is also funded by Lenovo Outstanding Young Scientists Program (LOYS).

References

1. Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 329–344. Springer, Heidelberg (2014). doi:[10.1007/978-3-319-10584-0_22](https://doi.org/10.1007/978-3-319-10584-0_22)
2. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: delving deep into convolutional nets. In: Proceedings of the British Machine Vision Conference, BMVC 2014 (2014)
3. Ciregan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012, pp. 3642–3649 (2012)
4. Dixit, M., Chen, S., Gao, D., Rasiwasia, N., Vasconcelos, N.: Scene classification with semantic fisher vectors. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, pp. 2974–2983 (2015)
5. Erhan, D., Bengio, Y., Courville, A., Vincent, P.: Visualizing higher-layer features of a deep network. Technical report, Department of IRO, Université de Montréal (2009)
6. Hansen, L.K., Salamon, P.: Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.* **12**(10), 993–1001 (1990)
7. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **37**(9), 1904–1916 (2015)

8. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 2014 ACM Conference on Multimedia, MM 2014. pp. 675–678 (2014)
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proceedings of the 26th Annual Conference on Neural Information Processing Systems, NIPS 2012, vol. 2, pp. 1097–1105 (2012)
10. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. *Neural Comput.* **1**(4), 541–551 (1989)
11. Lin, M., Chen, Q., Yan, S.: Network in network. In: Proceedings of the International Conference on Learning Representations, ICLR 2014 (2014)
12. Quattoni, A., Torralba, A.: Recognizing indoor scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009, pp. 413–420 (2009)
13. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Kholsa, A., Bernstein, M., Berg, A., Fei-Fei, L.: Imagenet large scale visual recognition challenge. *Int. J. Comput. Vis.* **115**(3), 211–252 (2015)
14. Sermante, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: integrated recognition, localization and detection using convolutional networks. In: Proceedings of the International Conference on Learning Representations, ICLR 2014 (2014)
15. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: visualising image classification models and saliency maps. In: Proceedings of the International Conference on Learning Representations Workshops, ICLR Workshops 2014 (2014)
16. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Proceedings of the 28th Annual Conference on Neural Information Processing Systems, NIPS 2014, vol. 1, pp. 568–576 (2014)
17. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proceedings of the International Conference on Learning Representations, ICLR 2015 (2015)
18. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, pp. 1–9 (2015)
19. Van Der Maaten, L., Hinton, G.: Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**, 2579–2605 (2008)
20. Wang, S., Jiang, S.: INSTRE: a new benchmark for instance-level object retrieval and recognition. *ACM Trans. Multimedia Comput. Commun. Appl. (TOMM)* **11**(3), 1–21 (2015)
21. Wu, C., Fan, W., He, Y., Sun, J., Naoi, S.: Cascaded heterogeneous convolutional neural networks for handwritten digit recognition. In: Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, pp. 657–660 (2012)
22. Wu, R., Wang, B., Wang, W., Yu, Y.: Harvesting discriminative meta object with deep CNN features for scene classification. In: IEEE International Conference on Computer Vision, ICCV 2015 (2015)
23. Wu, Z., Zhang, Y., Yu, F., Xiao, J.: A GPU implementation of GoogLeNet. Technical report, Princeton University (2014)

24. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: large-scale scene recognition from abbey to zoo. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognitions, CVPR 2010, pp. 3485–3492 (2010)
25. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Proceedings of the 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014, vol. 4, pp. 3320–3328 (2014)
26. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Proceedings of the 28th Annual Conference on Neural Information Processing Systems 2014, NIPS 2014, vol. 1, pp. 487–495 (2014)