# Design and Optimization of the Model for Traffic Signs Classification Based on Convolutional Neural Networks

Jiarong Song, Zhong Yang(✉), Tianyi Zhang, and Jiaming Han

Nanjing University of Aeronautics and Astronautics,
Nanjing 210016, People's Republic of China
jrsnuaa@163.com, {YangZhong,hjm}@nuaa.edu.cn,
ufozty@126.com

**Abstract.** Recently, convolutional neural networks (CNNs) demonstrate state-of-the-art performance in computer vision such as classification, recognition and detection. In this paper, a traffic signs classification system based on CNNs is proposed. Generally, a convolutional network usually has a large number of parameters which need millions of data and a great deal of time to train. To solve this problem, the strategy of transfer learning is utilized in this paper. Besides, further improvement is implemented on the chosen model to improve the performance of the network by changing some fully connected layers into convolutional connection. This is because that the weight shared feature of convolutional layers is able to reduce the number of parameters contained in a network. In addition, these convolutional kernels are decomposed into multi-layer and smaller convolutional kernels to get a better performance. Finally, the performance of the final optimized network is compared with unoptimized networks. Experimental results demonstrate that the final optimized network presents the best performance.

**Keywords:** CNNs · Transfer learning · Optimization

## 1 Introduction

In recent years, with the improvement of calculation speed of computers, convolutional neural networks (CNNs) are applied more and more and present remarkable performance especially in computer vision. As a result, there is an increasing number of researchers begin to apply CNNs in their work to solve vision problems which are difficult for traditional vision methods.

A CNN with excellent performance generally contains a large number of parameters which need millions of data to train. However, data available to us are usually limited. In this case, training a CNN usually containing millions of parameters is possible to produce over fitting problem [1]. In order to solve this problem, paper [2] proposes feature extraction algorithms to reduce the dimensionality of data to avoid over fitting. But handmade feature extraction easily abandons some useful features, which indicates methods in [2] may weaken the performance of a network. A network that is restricted to work on a small labelled region is presented in [3]. Undeniably, this method may also limit the application of the network in practice. Paper [4] develops an approach which learns to get effective filter sets based on the fixed filter sets to overcome the problem of limited data. Nevertheless, it probably limits the learning ability of feature extraction of a network. A noisy CNN algorithm is described in [5] which assists a network to give better performance on small data. Paper [6] investigates data augmentation complementary approaches to deal with limited training data. However, these two methods are not able to essentially optimize the structure of a network. Yu et al. propose a method of 1 x 1 convolutional layers to reduce the parameters to get a CNN model with better performance on hyperspectral image classification in [7]. However, the model is still large when it contains a great number of features. All of the above articles put forward their respective methods which are instructive to get an excellent model based on small data, even though they have a few disadvantages.

In this paper, based on small data, the strategy of transfer learning is utilized by employing a well-trained feature extractor and further optimization is implemented on the chosen CNN model to design a new model. Compared with above approaches, the method in this paper enables to reduce the amount of parameters and alleviate the over fitting problem without data preprocessing. Besides, this method will not restrict the application of the network in practice. Based on AlexNet [8], the first step is changing some fully connected layers into convolutional layers. In addition, these convolutional kernels are furtherly decomposed into multi-layer and smaller convolutional kernels to decrease the amount of parameters. Finally, experimental results demonstrate that the new model achieves excellent classification ability on our data and presents strong robustness.

## 2 Classification Network

In this section, as one of the most famous classification networks, AlexNet [8] which achieves an excellent performance of the 1000 class classification is introduced. Its structure can be divided into two parts. The first part is a feature extractor and the second part is a classifier.

### 2.1 The Feature Extractor

The feature extractor in AlexNet mainly consists of convolutional layers, activation function units and pooling layers. Convolutional kernels are similar to filters used in traditional vision systems. However, convolutional kernels, which are more advantageous than filters used in traditional vision systems, are able to not only extract low-level features of images such as edge information, direction information but also extract high-level features such as class information, etc.

The Eq. (1) illustrates the output size of an image after the convolution operation. The Eq. (1) shows that the size of output is generally smaller than the size of input after convolution operation. In this case, increase of convolutional layers easily leads to the shrinkage of data dimension propagating forward in a CNN. It means features of data that can be used in a network are greatly reduced, which is a restriction on the depth of a CNN. In order to solve this problem, padding which means to resize an input into a larger size is introduced. It is realized by boundary filling according to the size of the convolutional kernels. After padding, the size of output after the convolution operation is illustrated in (2). The Eq. (3) presents the size of output after an operation consisting of two steps. The first step is padding and the second step is convolution operation. The convolutional kernel is supposed to be in $N \times N$ size and its stride is supposed to be 1. As a result, the size of output is able to keep as the same as the input, which efficiently solves the shrinkage of data problem.

At the same time, in order to reduce the data dimension propagating forward in a network, pooling layers acting as the role of sampling are introduced. The main function of pooling layers is to simplify the output of their upper convolutional layers [9, 10]. Activation function used in AlexNet, which introduces nonlinear factors into a network, is ReLU function rather than tanh function [8]. Since data saturation which will induce gradient vanishing will never occur in ReLU function. Additionally, computing the derivative of ReLU function is faster than tanh [11, 12]. For ReLU function, if the input value is larger than 0, the gradient is set to be one. Otherwise, the gradient is zero.

$$size_{output} = \left(\frac{H_{input} - N}{str} + 1\right)\left(\frac{W_{input} - N}{str} + 1\right) \tag{1}$$

$$size_{output\_p} = \left(\frac{H_{input} + 2P - N}{str} + 1\right)\left(\frac{W_{input} + 2P - N}{str} + 1\right) \tag{2}$$

$$\begin{aligned} size_{output\_} &= \left[\left(H_{input} + 2 \times \frac{N-1}{2} - N\right) + 1\right]\left[\left(W_{input} + 2 \times \frac{N-1}{2} - N\right) + 1\right] \\ &= H_{input} \times W_{input} \end{aligned} \tag{3}$$

where $H_{input} \times W_{input}$ presents the size of the original input, $P$ presents the boundary filling, $size_{output}$ presents the size of output without padding, $size_{output\_P}$ and $size_{output\_}$ present the size of output with padding, the convolutional kernel is in $N \times N$ size and its stride is $str$. Besides, all of the above formulas represent operations on a single channel.

## 2.2 The Classifier

The classifier in AlexNet mainly consists of fully connected layers and output layers. Outputs of the classifier are inputs of the feature extractor. Fully connected layers learn to combine features from the feature extractor in a reasonable way to get the most likely class of inputs of the whole network, which is similar to a function mapping [13, 14]. In order to alleviate over fitting [1, 15] and improve robustness and accuracy of a convolutional neural network, dropout layers are introduced [8, 15].

The principle of dropout is randomly selecting activated neurons according to a certain probability and setting their outputs to be zero. In this case, a whole network is divided into many subnetworks since only a part of neurons work every time [8]. During the training phase, our purpose is to improve the performance of each subnetwork as far as possible. Hence, each subnetwork finally inclines to give correct classification. Additionally, because that only one subnetwork works every time. As a result, all these subnetworks work independently rather than in a certain combination [8, 15], which is able to efficiently avoid over fitting. Meanwhile, with dropout layers, fewer features are used at every turn during the training phase. This is also capable of avoiding over fitting. During the test phase, local noises may lead some subnetworks to output bad scores. However, most subnetworks still have good performances, which helps the whole network to show a remarkable performance. This character leads the whole network to demonstrate excellent robustness.

Besides, in order to use more information of data, dropout layers will be cancelled sometimes during the test phase, which needs additional data processing. In this case, the value of inputs should be reduced according to the drop ratio. Otherwise, inputs for neurons are larger than expected, which is showed in Eqs. (4) and (5). This may cause some mistakes because of the change in data distribution.

$$input_{withdropout} = p \sum w_i x_i + b \tag{4}$$

$$input_{withoutdropout} = \sum w_i x_i + b \tag{5}$$

where $input_{withdropout}$ and $input_{withoutdropout}$ present the total input of a neuron, $p$ presents the value of the drop ratio and $p$ is in region [0, 1], $w_i$ presents the value of a weight, $x_i$ presents an input of this neuron, $b$ presents the value of the bias.

## 3 Transfer Learning and Optimization

### 3.1 Transfer Learning and Its Reason and Reasonability

Training a convolutional network is greedy for data. However, our samples are limited. In order to alleviate the over fitting problem and gain an excellent performance, transfer learning [16–19] is introduced here. This is because that data used in our network and used in AlexNet are both image data, which implies that the feature extractor trained on AlexNet is suitable for our network. Transfer learning is achieved by fine tuning the feature extractor of AlexNet and designing a new classifier in this paper.

### 3.2 Network Structure and Optimization

The simplest transfer learning method is just changing the output number of AlexNet into our class number, which is marked as network 1. The classifier of network1 contains nearly $5 \times 10^7$ parameters which are large for our data sets. Thus, changing fully connected layers into convolutional layers is considered. This is because that the weight shared feature of convolutional layers [20, 21] is able to reduce the amount of parameters. In this paper, the first fully connected layer of AlexNet is changed into a convolutional layer which consists of $5 \times 5 \times 256$ sized convolutional kernels. After this step, network 2 is constructed and the classifier of network 2 contains nearly $4 \times 10^7$ parameters that are still large. Thus, further optimization is needed.

Before doing further optimization, two combinations of convolutional kernels are compared with each other in Table 1. In Table 1, the input is supposed to be in size of $H \times W \times C$ and all kernels are in $N \times N \times C$ size. Table 1 shows that the $3 \times 3 \times C$ and $3 \times 3 \times C$ combination structure includes fewer parameters. It implies that a network which consists of multi-layer and smaller convolutional kernels contains fewer parameters. According to this conclusion, based on the preliminary improvement, further optimization is implemented on network 2. Further optimization is decomposing $5 \times 5 \times C$ convolutional kernels into multi-layer and smaller $3 \times 3 \times C$ and $3 \times 3 \times C$ sized convolutional kernels. After this step, network 3 which is our final model is obtained. The classifier of network 3 contains around $1 \times 10^7$ parameters and N presents class number. The structure of network 3 is showed in Fig. 1.

**Table 1.** Comparison of two different combinations of convolutional kernels.

| The combinational of convolutional kernels | Perception size | Number of weights | Number of multiply-adds |
|---|---|---|---|
| $5 \times 5 \times C$ | 25 | $25C$ | $25\ HWC^2$ |
| $3 \times 3 \times C$ and $3 \times 3 \times C$ | 25 | $18C$ | $18\ HWC^2$ |

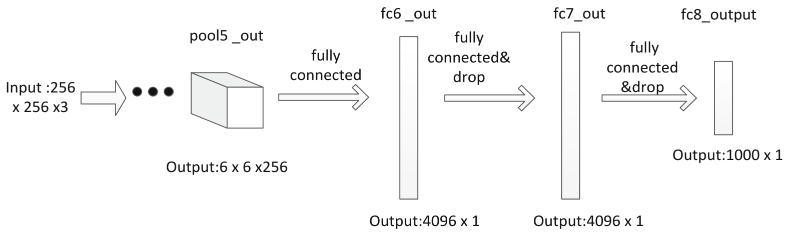


**Fig. 1.** The structure of network 3

## 4 Experiment and Analysis

### 4.1 Experimental Data

The purpose of this paper is to classify traffic signs. Training sets consist of samples from other places and testing sets consist of samples from campus. Both of them are divided into three categories. Because of data limitation, data amount is augmented by rotating, cropping and mirroring. Details are given in Table 2.

### 4.2 The Influence of Drop Ratio

According to Sect. 2, dropout layers are introduced into CNNs to prevent over fitting [1]. Generally, the higher the drop ratio is, more independent subnetworks will be [8], more excellent performance a network presents. Since our data sets are small, in order

**Table 2.** Data statistics.

| Samples | Training set | Testing set |
|---|---|---|
| No parking sign | 398 | 55 |
| Speed limit sign | 386 | 50 |
| Crosswalk | 358 | 47 |
| total | 1142 | 152 |

to obtain a remarkable performance, a little higher drop ratio is chosen. However, too much high drop ratio will have bad influence on a network. Figure 2 shows loss values of different drop ratios based on network 3. Figure 3 is the local enlarged image of Fig. 2. Drop ratio1, drop ratio2, drop ratio3, drop ratio4 are respectively equal to 0.4, 0.9, 0.8, 0.6. Interval [130, 138] is in the region in which all loss values are stable.

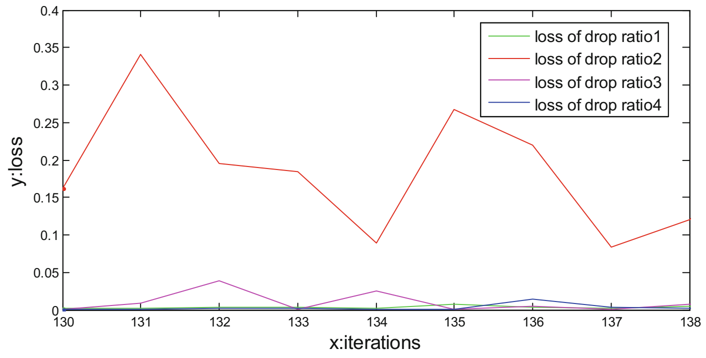


**Fig. 2.** Loss values on training sets

Evidently, according to Figs. 2 and 3, drop ratio4 and drop ratio1 have the best loss values. Drop ratio2 has the worst loss value when compared with others. This is because that too many features are dropped during the training phase with drop ratio2, which easily generates under fitting [1]. Under the condition of drop ratio3, many useful features are still dropped when compared with drop ratio1 and drop ratio4. Obviously, if the drop ratio is set to be 1, the network will stop updating.

In addition, accuracy values of drop ratio1 and drop ratio4 are compared with each other in Fig. 4. Interval [76, 87] is in the region in which both accuracy values are stable. Figure 4 illustrates that drop ratio4 has a higher accuracy value on the testing set than drop ratio1 even though they have nearly same loss values on the training set. It implies that network 3 with drop ratio4 has better robustness than this network with drop ratio1. On the basis of this section, drop ratio4 is chosen as the value of the drop ratio in the next section.

## 4.3 Training and Testing of Networks

According to session 4.2, all drop ratios in network 1, in network 2 and in network 3 are set to be 0.6. Classifiers of these three networks have the same learning rate 0.01 for

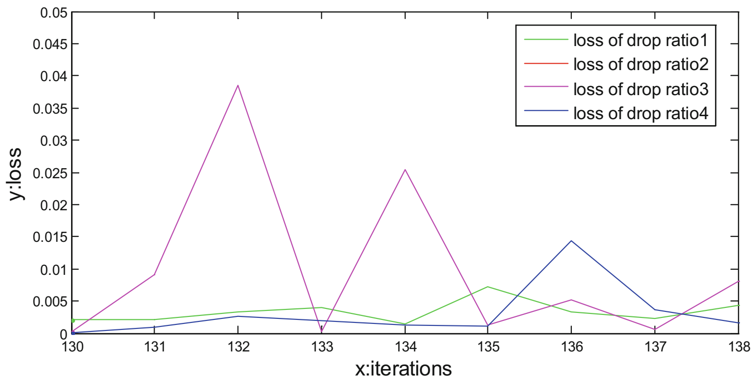


**Fig. 3.** Local enlarged image of Fig. 2

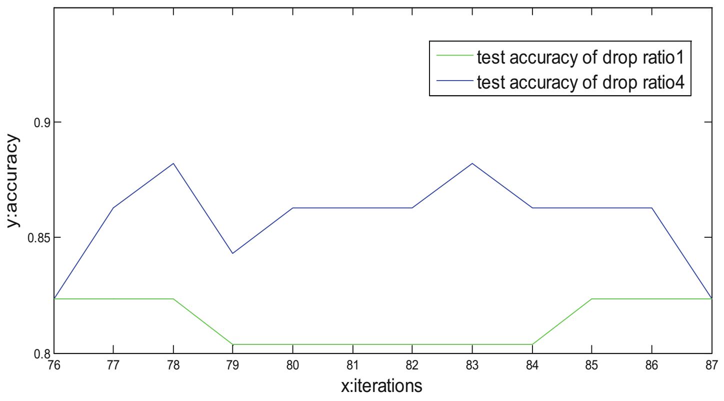


**Fig. 4.** Accuracy values on testing sets

weights and 0.02 for biases. The learning rate of feature extractors of these three networks is almost equal to 0. Experimental results are shown in Figs. 5 and 6, which respectively present loss values on training sets and accuracy values on testing sets. Interval [112, 128] is in the region in which all loss values are stable and interval [126, 135] is in the region in which all accuracy values are stable. Figures 5 and 6 present that these three networks all have good loss values which are not more than 0.04 and accuracy values which are not less than 88%. Since that transfer learning method is used and data sets are not large. In this case, these three networks are likely to show excellent performance in training loss and testing accuracy.

In order to furtherly analyze these results, we calculate average values of accuracy in region [126, 135] of these three networks according to Fig. 5. Results are presented in Table 3. Evidently, network 3 has the best result in test accuracy. This is because that network 3 has the fewest parameters when compared with other networks. Thus, under the condition of the same amount of data, network 3 relatively has the most

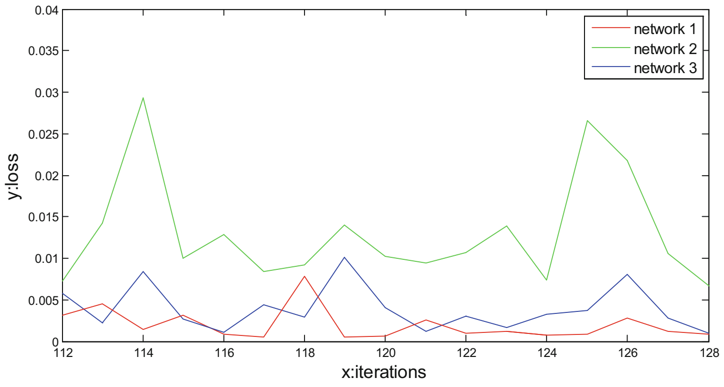


**Fig. 5.** Loss values on training sets

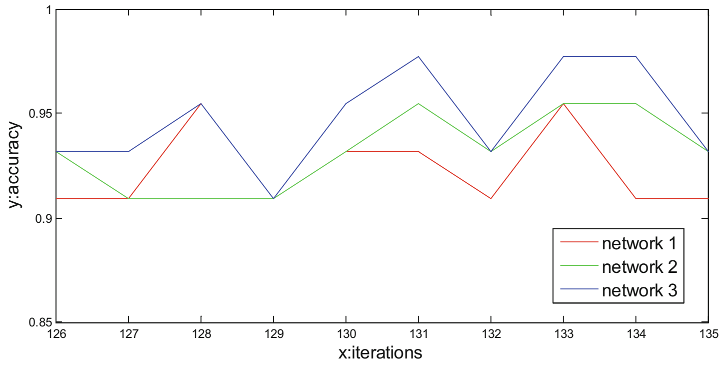


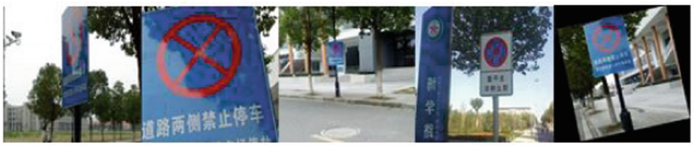
**Fig. 6.** Accuracy values on testing sets

**Table 3.** Test accuracy of three networks

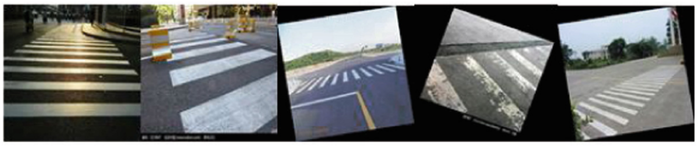
| Model | Test accuracy |
|---|---|
| Network 1 | 92.3% |
| Network 2 | 92.6% |
| Network 3 | 94.9% |

sufficient samples to get a more superior performance. From the experiment, it comes to a conclusion that network 3 has the best performance on our data.

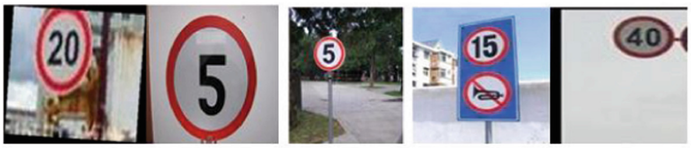
Finally, some prediction experiments of single inputs are conducted based on network 3 and results are presented in Fig. 7. In Fig. 7, results represent the probability that every image belongs to each class. Figure 7 shows that some pictures are rotated and fuzzy. However, network 3 still shows excellent performance on the accuracy of classification, which proves network 3 has a high accuracy of classification and strong robustness.



| | | | | | |
|---|---|---|---|---|---|
| Crosswalk | 0.0036 | 0.0000 | 0.0097 | 0.0000 | 0.0379 |
| No Parking Sign | 0.9907 | 1.0000 | 0.9902 | 0.9998 | 0.9620 |
| Speed Limit Sign | 0.0057 | 0.0000 | 0.0001 | 0.0002 | 0.0001 |



| | | | | | |
|---|---|---|---|---|---|
| Crosswalk | 1.0000 | 0.9998 | 0.9970 | 0.9982 | 0.9960 |
| No Parking Sign | 0.0000 | 0.0002 | 0.0030 | 0.0013 | 0.0000 |
| Speed Limit Sign | 0.0000 | 0.0000 | 0.0000 | 0.0005 | 0.0040 |



| | | | | | |
|---|---|---|---|---|---|
| Crosswalk | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| No Parking Sign | 0.0289 | 0.0000 | 0.0025 | 0.0991 | 0.0000 |
| Speed Limit Sign | 0.9711 | 1.0000 | 0.9975 | 0.9009 | 1.0000 |

**Fig. 7.** Prediction for different inputs

## 5 Conclusions

This paper utilizes the strategy of transfer learning to construct a CNN model to alleviate over fitting problem on small data. An excellently well-trained feature extractor that is difficult to achieve just on small data sets is obtained by using the strategy of transfer learning. In this case, our data are mainly used to train the classifier, which is more likely to generate a satisfying model. Meanwhile, in order to reduce the amount of parameters and get a network with more superior performance. In this paper, the first step is changing some fully connected layers into convolutional connection according to the weight shared feature of convolutional layers. Next, these convolutional kernels are decomposed into multi-layer and smaller convolutional kernels, which furtherly cuts down the number of parameters contained in the network. Finally, the experimental result proves the final optimized network has the best performance.

**Acknowledgement.** This work was supported by the National Natural Science Foundation of China (No. 61473144), Aviation Science Foundation (Key Laboratory, No. 20162852031) and Major Scientific Instruments and Equipment Development of the Ministry of Science and Technology (No. 2016YFF0103702).

## References

1. Gu, Y., et al.: An optimal sample data usage strategy to minimize over fitting and under fitting effects in regression tree models based on remotely-sensed data. Remote sens. **8**(11) pp. 1–13, Article 943 (2016)
2. Kumar, S., Ghosh, J., Crawford, M.M.: Best-bases feature extraction algorithms for classification of hyperspectral data. IEEE Trans. Geosci. Remote Sens. **39**(7), 1368–1379 (2001)
3. Windrim, L., et al.: Hyperspectral CNN classification with limited training samples (2016)
4. Jacobsen, J.H., et al.: Structured receptive fields in CNNs, pp. 2610–2619 (2016)
5. Audhkhasi, K., Osoba, O., Kosko, B.: Noise-enhanced convolutional neural networks. Neural Netw. Off. J. Int. Neural Netw. Soc. **78**, 15–23 (2015)
6. Cui, X., Goel, V., Kingsbury, B.: Data augmentation for deep neural network acoustic modeling. In: 2014 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2014, pp. 5582–5586. IEEE (2014)
7. Yu, S., Jia, S., Xu, C.: Convolutional neural networks for hyperspectral image classification. Neurocomputing **219**, 88–98 (2016)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: International Conference on Neural Information Processing Systems, pp. 1097–1105. Curran Associates Inc. (2012)
9. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
10. Sun, M., et al.: Learning pooling for convolutional neural network. Neurocomputing (2016)
11. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier networks. Learn. Stat. Optim. (2010)
12. Jarrett, K., et al.: What is the best multi-stage architecture for object recognition? In: Proceedings of International Conference on Computer Vision, pp. 2146–2153 (2009)
13. Li, H.X., Chen, C.P.: The equivalence between fuzzy logic systems and feedforward neural networks. IEEE Trans. Neural Netw. **11**(2), 356–365 (2000)
14. Wilamowski, B.M.: Neural network architectures and learning algorithms. Ind. Electron. Mag. IEEE **3**(4), 56–63 (2010)
15. Srivastava, N., et al.: Dropout: a simple way to prevent neural networks from over fitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
16. Tajbakhsh, N., et al.: Convolutional neural networks for medical image analysis: full training or fine tuning? IEEE Trans. Med. Imaging **35**(5), 1299–1312 (2016)
17. Krizhevsky, A.: Convolutional deep belief networks on CIFAR-10 (2012)
18. Shin, H.C., et al.: Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE Trans. Med. Imaging **35**(5), 1285 (2016)
19. Marmanis, D., et al.: Deep learning earth observation classification using ImageNet pretrained networks. IEEE Geosci. Remote Sens. Lett. **13**(1), 105–109 (2016)
20. Kim, J.K., Lee, M.Y., Kim, J.Y.: An efficient pruning and weight sharing method for neural network. In: IEEE International Conference on Consumer Electronics, p. 2 (2016)
21. Bouvrie, J.: Notes on convolutional neural networks. Neural Nets (2006)