Traffic sign recognition based on deep convolutional neural network*

YIN Shi-hao (尹世豪), **DENG Ji-cai** (邓计才)**, **ZHANG Da-wei** (张大伟), **and DU Jing-yuan** (杜靖远)
School of Information Engineering, Zhengzhou University, Zhengzhou 450001, China

(Received 12 September 2017)

Traffic sign recognition (TSR) is an important component of automated driving systems. It is a rather challenging task to design a high-performance classifier for the TSR system. In this paper, we propose a new method for TSR system based on deep convolutional neural network. In order to enhance the expression of the network, a novel structure (dubbed block-layer below) which combines network-in-network and residual connection is designed. Our network has 10 layers with parameters (block-layer seen as a single layer): the first seven are alternate convolutional layers and block-layers, and the remaining three are fully-connected layers. We train our TSR network on the German traffic sign recognition benchmark (GTSRB) dataset. To reduce overfitting, we perform data augmentation on the training images and employ a regularization method named "dropout". The activation function we employ in our network adopts scaled exponential linear units (SELUs), which can induce self-normalizing properties. To speed up the training, we use an efficient GPU to accelerate the convolutional operation. On the test dataset of GTSRB, we achieve the accuracy rate of 99.67%, exceeding the state-of-the-art results.
Document code: A **Article ID:** 1673-1905(2017)06-0476-5
DOI https://doi.org/10.1007/s11801-017-7209-0

With the rapid development of driver assistance systems and autonomous vehicles in recent years, traffic sign recognition (TSR) attracts attention of many researchers. Traffic signs provide lots of useful information of the traffic environment, such as speed limits, directions, danger warning, and so on. Recognizing traffic signs timely makes driving safe and convenient. However, all the traffic sign images, which are input to the TSR system, are photographed from real environment, and they have various lighting conditions, viewpoints, partial occlusions, and resolutions. Therefore, it is important to overcome these difficulties to ensure the robustness of model.

Many models of TSR system have been proposed in the past decade[1-3]. Generally, the TSR process can be divided into two stages, i.e., detection and recognition, and we focus on the recognition stage in this paper. Lim et al[4] used color/shape and pictogram information to extract features and then fed them to radial basis function neural network (RBFNN) to classify the traffic signs. Madani and Yusof[5] employed a pre-trained multiclass support vector machine (MCSVM) to classify the traffic signs instead of the aforementioned RBFNN. Recently, convolutional neural networks (CNNs) are popular in the computer vison area, because of their excellent classification performance. Lau et al[6] proposed a TSR method based on CNN and achieved a good accuracy on the Malaysia traffic sign database. In order to improve the accuracy of recognizing traffic signs, we propose a deep CNN architecture and utilize it on the German traffic sign recognition benchmark (GTSRB) dataset.

Rectified linear unit (ReLU) is the most popular activation function used by deep CNNs in recent years. Deep CNNs with ReLUs train faster than their equivalents with hyperbolic tangent or sigmoid units[7]. Unfortunately, ReLU units can be fragile during training and "die". Leaky ReLU was proposed to fix the "dying ReLU" problem[8]. Xu et al[9] proved that leaky ReLUs perform better than ReLUs in CNN. In order to train deep CNNs, batch normalization or layer normalization is typically used to normalize the output of every layer in network. The activation function we employ in our network is scaled exponential linear units (SELUs) proposed by Klambauer et al[10]. The SELUs converge to zero mean and unit variance when propagated through the network. In our network, we find that SELUs achieve higher accuracy than ReLUs (about 0.95%) and leaky ReLUs (about 0.4%) on the GTSRB dataset. The SELU activation function is given by

$$\mathrm{selu}(x)=\begin{cases}\lambda x & ,x>0\\ \lambda\alpha(\mathrm{e}^{x}-1), & x\le 0\end{cases}, \tag{1}$$

* This paper was presented in part at the CCF Chinese Conference on Computer Vision, Tianjin, 2017. This paper was recommended by the program committee.
** E-mail: iejcdeng@zzu.edu.cn

where λ=1.050 7 and α=1.673 26.

In traditional CNNs, layers were typically structured by stacked convolutional layers (optionally followed by normalization layer and pooling layer) and several fully-connected layers, such as LeNet-5[11], AlexNet[12] and VGG[13]. Lin et al[14] proposed a novel deep neural network called network in network (NIN) to enhance the representation power of convolutional neural network. Inspired by the NIN, Szegedy et al[15] designed a deep convolutional neural network architecture codenamed "Inception" and took the crown of the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2014.

Residual network was firstly proposed by He et al in Ref.[16], in which they gave empirical evidence for the advantages of adding residual connection to the network. Szegedy et al[17] combined the Inception architecture with residual connections and got higher accuracy than that without residual connections on the ILSVRC 2012 classification task.

In order to extract different level features by using several sizes of convolutional filters, and benefit from adding residual connection, we design a novel architecture named block-layer as shown in Fig.1. To simplify the representation, we omit the activation function in the figure. All the convolutional layers and max-pooling layers in the block-layer have the same stride of 1 and the same padding scheme of "same-padding" to ensure their output grids match the sizes of their input. In other words, the output of block-layer has the same size as its input. This mechanism makes the block-layer be added to anywhere in traditional plain CNNs. In the block-layer, 1×1 convolutions before the expensive 5×5 and 3×3 convolutions are used as dimension reduction models to limit the size of our network.

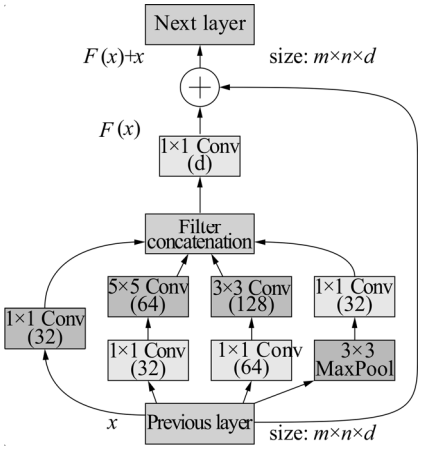


Fig.1 The block-layer used in our TSR network

The output $H(x)$ of block-layer is a function of its input x. The stacked underlying mapping of the block-layer is denoted as $F(x)$. Because of adding the residual connection, the output of block-layer can be described as follows:

$$H(x)=F(x)+x. \quad (2)$$

Now we are ready to describe the overall architecture of our traffic sign recognition model. A concise representation is summarized in Fig.2. For simplicity, the block-layer described above is represented to a single layer. The network contains 10 layers with weights: the first seven are alternate convolutional layers and block-layers, and the remaining three are fully-connected layers. The output of last fully-connected layer is fed to a 43-way softmax which produces a distribution over the 43 class traffic sign labels. Max-pooling layers follow the layers C3, C5 and C7. The grid of pooling is 3 and the stride is set to 2. In other words, we employ the overlapping pooling in our network.

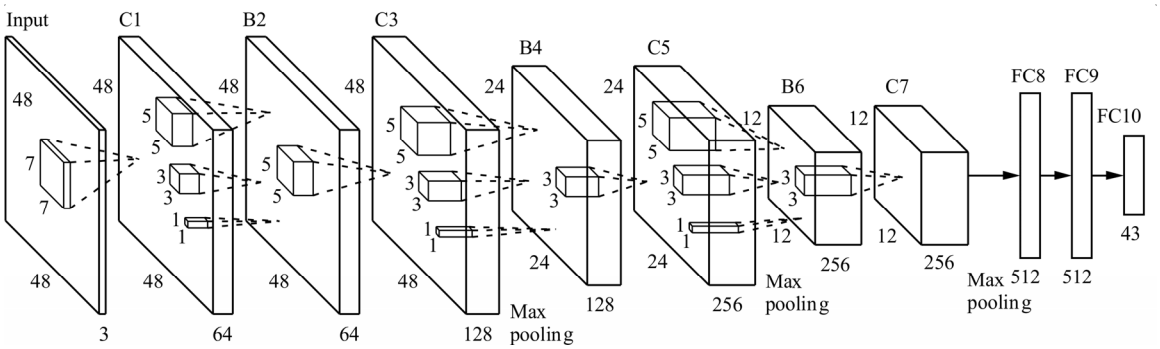


Fig.2 A simple illustration of the architecture of our deep convolutional neural network for TSR

The input of our network is an RGB image with size of 48×48×3. The preprocessing of the input image is to linearly scale it to achieve zero mean and unit norm. As depicted in Fig.2, the layers B2, B4 and B6 are block-layers connected to previous corresponding convolutional layers (maybe pooled). The layer C1 filters the input image with 64 kernels of size 7×7×3 with a stride of 1 pixel. The layer C3 has 128 kernels of size 5×5×64 connected to previous block-layer. The layer C5 has 256 kernels of size 3×3×128, and the layer C7 has 256 kernels of size 3×3×256. The first two fully-connected layers (FC8 and FC9) have 512 neurons each. The last fully-connected layer (FC10) has 43 neurons. A schematic view of our network in detail is also depicted in Fig.3.

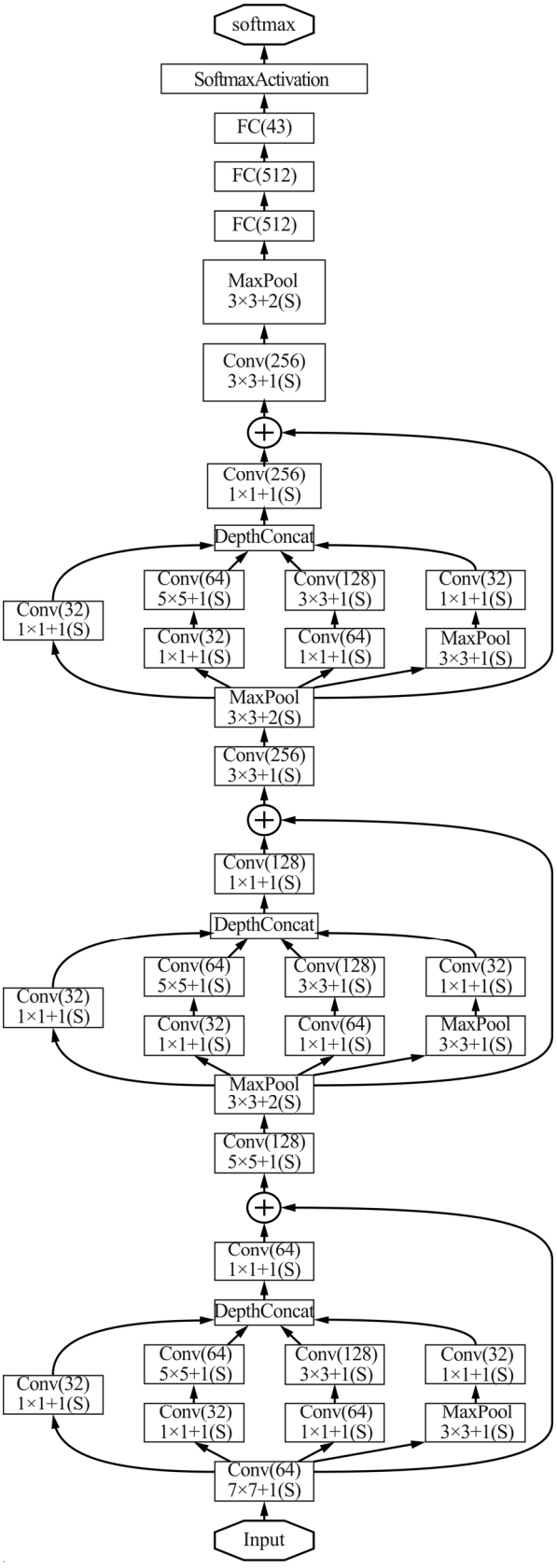


Fig.3 A detailed representation of our TSR network

The GTSRB dataset[18] has 51 893 images (39 209 training images and 12 630 testing images) of the 43 classes shown in Fig.4. The size of the traffic sign images varies between 15×15 and 222×193. The images contain 10% margin (at least 5 pixels) around the traffic sign. All the images were collected from real environment, which have different illumination and weather conditions. Some of the traffic sign images are not easy to recognize because of low resolution, bad illumination, partial occlusion, motion blur, and so on. Fig.5 shows some example images, which are selected from the GTSRB dataset under contaminated conditions.

Fig.4 The 43 traffic sign classes in the GTSRB dataset

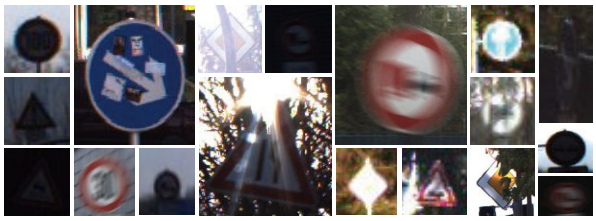


Fig.5 Examples under contaminated conditions picked from the GTSRB dataset, such as low resolution, bad illumination, partial occlusion and motion blur

In the training set of GTSRB, the number of images in each class varies between 210 and 2 250. In order to avoid the influence of unbalanced data on the experimental result, we extend the number of images in each class to 2 250. For the classes containing less than 2 250 images, we randomly select some images in the classes, and then adjust brightness, contrast, rotation and so on. In this way, the number of training data can be increased to 96 750.

In order to reduce overfitting, we conduct data augmentation on training dataset. Because the images in the GTSRB dataset have different sizes, we extract the interest regions of training images and scale them to the size of 52×52 first. For a scaled image, we randomly extract 48×48 patches from it. This increases the size of our training set by a factor of 16. Another way to enlarge the training set we employ is to adjust the brightness and contrast of the image randomly. Also considering that the images of the GTSRB dataset were collected from real world, and some of them have a bad lighting condition,

we only adjust the brightness and contrast by a factor randomly picked in a short interval.

We train our network using TensorFlow[19] machine learning system developed by Google Inc. Our training uses Adam[20] optimizer with 0.000 1 learning rate (decreasing the learning rate by 10% every 30 epochs). The batch size is set to 128. We use weight decay (the L2 penalty multiplier is set to 0.000 5) and dropout[21] regularization for the first two fully-connected layers (dropout ratio to 0.5) to reduce overfitting. We initialize the weights in our network by using the random initialization procedure of Ref.[22]. The biases are initialized with zero in our network. We train our network for roughly 100 cycles through the balanced training set, which takes about 17 h on an NVIDIA Tesla M60 8G GPU. The details of the experiment are shown in Fig.6.

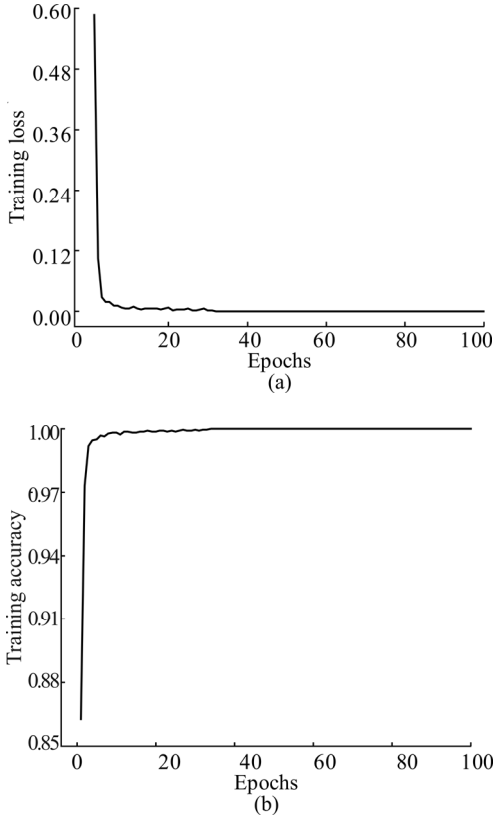


Fig.6 Training on the GTSRB dataset: (a) Training loss vs. training epochs; (b) Training accuracy vs. training epochs

In test time, we extract the region of traffic sign and scale it to 48×48 pixels first, and then input them to the trained network. The detail of testing accuracy is shown in Fig.7. The final accuracy on the GTSRB of our network is 99.67%. Tab.1 presents a comparison with some proposed algorithms submitted to the GTSRB and the human performance. The speeds of our network on both CPU and GPU setting in test stage are summarized in Tab.2. The experimental results show that the GPU is dozens of times faster than the CPU. Our network can process more than 300 images per second on an NVIDIA GTX 1060 6G GPU. In the case of using GPU, our network obtains high computational efficiency in the recognition process.

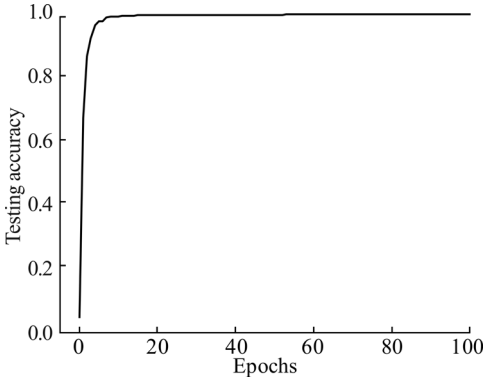


Fig.7 The testing accuracy on the GTSRB dataset vs. training epochs

Tab.1 Comparison of results on the GTSRB testing set

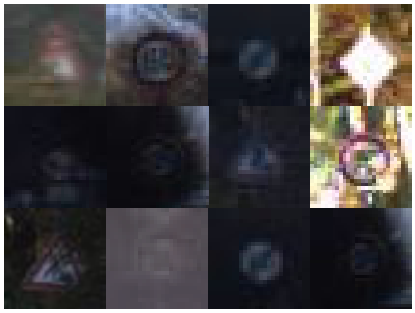
Team	Method	Accuracy (%)
Ours	Deep CNNs	99.67
IDSIA	Committee of CNNs	99.46
INI-RTCV	Human (best individual)	99.22
INI-RTCV	Human (average)	98.84
Sermanet	Multi-scale CNN	98.31
CAOR	Random Forests	96.14

Tab.2 The speeds of our network on different devices

Device	Speed
Nvidia GTX 1060 6G	3 ms/frame
Nvidia Tesla M60 8G (Single core)	4 ms/frame
Nvidia GTX 960 2G	8 ms/frame
Intel i7-6700K	45 ms/frame

Some true positive and false negative examples in test stage are shown in Fig.8. For the images under lightly contaminated conditions (Fig.8(a)), our trained network could make true prediction. But for some images under extremely contaminated conditions (Fig.8(b)), our network does not work well.

(a)



(b)

Fig.8 Some examples in the test stage: (a) True positive examples; (b) False negative examples

This paper proposes a deep CNN used for traffic sign recognition. The network contains three block-layers except for single convolutional and fully-connected layers. The block-layer in our network can extract different level features, and can flexibly be added to traditional CNNs. On the test dataset of GTSRB, we achieve the accuracy of 99.67%. This result exceeds the human level.

In future work, we intend to improve the accuracy of our network and accelerate the calculation in prediction stage. In this regard, we are planning to consider the following issues: to adjust the number of layers in our network, to employ more methods of data augmentation, and to reduce the number of parameters.

References

[1] Nguwi Y.Y. and Kouzani A.Z., Neural Computing and Applications **17**, 265 (2008).

[2] Mogelmose A., Trivedi M.M. and Moeslund T.B., IEEE Transactions on Intelligent Transportation Systems **13**, 1484 (2012).

[3] Lu X., Wang Y., Zhou X., Zhang Z. and Ling Z., IEEE Transactions on Intelligent Transportation Systems **18**, 960 (2017).

[4] Lim K.H., Seng K.P. and Ang L.M., Intra Color-shape Classification for Traffic Sign Recognition, 2010 International Computer Symposium (ICS2010), 642 (2010).

[5] Madani A. and Yusof R., Neural Computing and Applications, 1 (2017).

[6] Lau M.M., Lim K.H. and Gopalai A.A., Malaysia Traffic Sign Recognition with Convolutional Neural Network, 2015 IEEE International Conference on Digital Signal Processing (DSP), 1006 (2015).

[7] Glorot X., Bordes A. and Bengio Y., Deep Sparse Rectifier Neural Networks, Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, 315 (2011).

[8] Maas A.L., Hannun A.Y. and Ng A.Y., Rectifier Nonlinearities Improve Neural Network Acoustic Models, Proceedings of the 30th International Conference on Machine Learning (ICML13) **28**, 6 (2013).

[9] Xu B., Wang N., Chen T. and Li M., Empirical Evaluation of Rectified Activations in Convolutional Network, arXiv preprint arXiv:1505.00853 (2015).

[10] Klambauer G., Unterthiner T., Mayr A. and Hochreiter S., Self-Normalizing Neural Networks, arXiv preprint arXiv:1706.02515 (2017).

[11] LeCun Y., Bottou L., Bengio Y. and Haffner P., Proceedings of the IEEE **86**, 2278 (1998).

[12] Krizhevsky A., Sutskever I. and Hinton G.E., Advances in Neural Information Processing Systems **25**, 1097 (2012).

[13] Simonyan K. and Zisserman A., Very Deep Convolutional Networks for Large-scale Image Recognition, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[14] Lin M., Chen Q. and Yan S., Network in Network, Proc. ICLR, 2014.

[15] Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V. and Rabinovich A., Going Deeper with Convolutions, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1 (2015).

[16] He K., Zhang X., Ren S. and Sun J., Deep Residual Learning for Image Recognition, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770 (2016).

[17] Szegedy C., Ioffe S., Vanhoucke V. and Alemi A.A., Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, AAAI, 4278 (2017).

[18] Stallkamp J., Schlipsing M., Salmen J. and Igel C., Neural Networks **32**, 323 (2012).

[19] Abadi M., Agarwal A., Barham P. and Brevdo E., Tensorflow: Large-scale Machine Learning on Heterogeneous Distributed Systems, arXiv preprint arXiv:1603.04467 (2016).

[20] Kingma D. and Ba J., Adam: A Method for Stochastic Optimization, arXiv preprint arXiv:1412.6980 (2014).

[21] Hinton G.E., Srivastava N., Krizhevsky A., Sutskever I. and Salakhutdinov R.R., Improving Neural Networks by Preventing Co-adaptation of Feature Detectors, arXiv preprint arXiv:1207.0580 (2012).

[22] Glorot X. and Bengio Y., Understanding the Difficulty of Training Deep Feedforward Neural Networks, Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 249 (2010).