Neural Networks 32 (2012) 333–338

Contents lists available at SciVerse ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

2012 Special Issue

Multi-column deep neural network for traffic sign classification

Dan Cireşan *, Ueli Meier, Jonathan Masci, Jürgen Schmidhuber

IDSIA - USI - SUPSI — Galleria 2, Manno - Lugano 6928, Switzerland

ARTICLE INFO

Keywords:
Deep neural networks
Image classification
Traffic signs
Image preprocessing

ABSTRACT

We describe the approach that won the final phase of the German traffic sign recognition benchmark. Our method is the only one that achieved a better-than-human recognition rate of 99.46%. We use a fast, fully parameterizable GPU implementation of a Deep Neural Network (DNN) that does not require careful design of pre-wired feature extractors, which are rather learned in a supervised way. Combining various DNNs trained on differently preprocessed data into a Multi-Column DNN (MCDNN) further boosts recognition performance, making the system insensitive also to variations in contrast and illumination.

1. Introduction

The human visual system efficiently recognizes and localizes objects within cluttered scenes. For artificial systems, however, this is still difficult, due to viewpoint-dependent object variability, and the high in-class variability of many object types. Deep hierarchical neural models roughly mimic the nature of mammalian visual cortex, and are among the most promising architectures for such tasks. The most successful hierarchical object recognition systems all extract localized features from input images, convolving image patches with filters. Filter responses are then repeatedly pooled and re-filtered, resulting in a deep feed-forward network architecture whose output feature vectors are eventually classified. One of the first hierarchical neural systems was the Neocognitron by Fukushima (1980), which inspired many of the more recent variants.

Unsupervised learning methods applied to patches of natural images tend to produce localized filters that resemble off-center-on-surround filters, orientation-sensitive bar detectors, Gabor filters (Hoyer & Hyvärinen, 2000; Olshausen & Field, 1997; Schmidhuber, Eldracher, & Foltin, 1996). These findings in conjunction with experimental studies of the visual cortex justify the use of such filters in the so-called *standard model* for object recognition (Mutch & Lowe, 2008; Riesenhuber & Poggio, 1999; Serre, Wolf, & Poggio, 2005), whose filters are fixed, in contrast to those of Convolutional Neural Networks (CNNs) (Behnke, 2003; LeCun, Bottou, Bengio, & Haffner, 1998; Simard, Steinkraus, & Platt, 2003), whose weights (filters) are randomly initialized and learned in a supervised way using back-propagation (BP). A DNN, the basic building block of our proposed MCDNN, is a hierarchical deep neural network, alternating convolutional with max-pooling layers (Riesenhuber & Poggio, 1999; Scherer, Müller, & Behnke, 2010; Serre et al., 2005). A single DNN of our team won the offline Chinese character recognition competition (Liu, Yin, Wang, & Wang, 2011), a classification problem with 3755 classes. Ciresan, Meier, Gambardella, and Schmidhuber (2011) report state-of-the-art results on isolated handwritten character recognition using a MCDNN with 7 columns. Meier, Ciresan, Gambardella, and Schmidhuber (2011) show that there is no need for optimizing the combination of different DNNs: simply averaging their outputs generalizes just as well or even better on the unseen test set.

Despite the hardware progress of the past decades, computational speed is still a limiting factor for deep architectures characterized by many building blocks. For our experiments we therefore rely on a fast implementation on Graphics Processing Units (GPUs) (Ciresan, Meier, Masci, Gambardella, & Schmidhuber, 2011a). Our implementation is flexible and fully online (i.e., weight updates after each image). It allows for training large DNN within days instead of months, thus making MCDNN feasible.

Recognizing traffic signs is essential for the automotive industry's efforts in the field of driver assistance, and for many other traffic-related applications. The German traffic sign recognition benchmark (GTSRB) (Stallkamp, Schlipsing, Salmen, & Igel, 2011), a 43 class classification challenge, consisted of two phases: an online preliminary evaluation followed by an on-site final competition at the International Joint Conference on Neural Networks in 2011. We won the preliminary phase (Ciresan, Meier, Masci, & Schmidhuber, 2011b) using a committee of (Multi-Layer Perceptrons) MLP trained on provided features, and a DNN trained on raw pixel intensities. Here we present the method that won the on-site competition using a MCDNN, instead of a committee of MLP and DNN. Our new approach does not use handcrafted features anymore, relying only on the raw pixel images.

We first give a brief description of our MCDNN architecture, then describe the creation of the training set and the data

* Corresponding author. Tel.: +41 58 666 6710; fax: +41 58 666 6661.
E-mail addresses: dan@idsia.ch (D. Cireşan), ueli@idsia.ch (U. Meier), jonathan@idsia.ch (J. Masci), juergen@idsia.ch (J. Schmidhuber).
URL: http://www.idsia.ch/~ciresan (D. Cireşan).

0893-6080/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2012.02.023

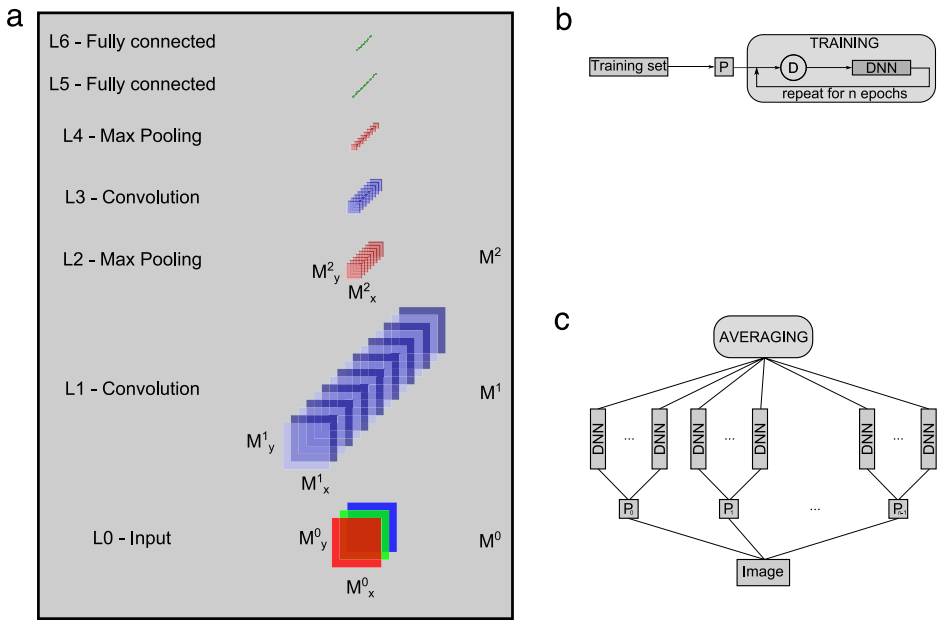

Fig. 1. (a) DNN architecture. (b) Training a DNN: the dataset is preprocessed (P) before training starts; during training all original or preprocessed images are randomly distorted before each epoch (D). (c) MCDNN architecture: the input image is preprocessed by n different preprocessors $P_0 - P_{n-1}$ and an arbitrary number of columns is trained on each preprocessed input. The final predictions are obtained by averaging individual predictions of each DNN.

preprocessing. We conclude by summarizing the results obtained during the on-site competition.

2. Multi-column deep neural networks

As a basic building block we use a deep hierarchical neural network that alternates convolutional with max-pooling layers, reminiscent of the classic work of Hubel and Wiesel (1962) and Wiesel and Hubel (1959) on the cat's primary visual cortex, which identified orientation-selective *simple cells* with overlapping local receptive fields and *complex cells* performing down-sampling-like operations. Such architectures vary in how *simple* and *complex* cells are realized and how they are initialized/trained. Here we give a brief description of our architecture; a detailed description of the GPU implementation can be found in Ciresan et al. (2011a).

2.1. DNN

A DNN (Fig. 1a) consists of a succession of convolutional and max-pooling layers, and each layer only receives connections from its previous layer. It is a general, hierarchical feature extractor that maps raw pixel intensities of the input image into a feature vector to be classified by several, usually 2 or 3, fully connected layers. All adjustable parameters are jointly optimized through minimization of the misclassification error over the training set.

2.1.1. Convolutional layer

Each convolutional layer performs a 2D convolution of its M^{n-1} input maps with a filter of size $K_x^n \times K_y^n$. The resulting activations of the M^n output maps are given by the sum of the M^{n-1} convolutional responses which are passed through a nonlinear activation function:

$$\mathbf{Y}_j^n = f\left(\sum_{i=1}^{M^{n-1}} \mathbf{Y}_i^{n-1} * \mathbf{W}_{ij}^n + b_j^n\right), \tag{1}$$

where n indicates the layer, $\mathbf{Y}$ is a map of size $M_x \times M_y$, and $\mathbf{W}_{ij}$ is a filter of size $K_x \times K_y$ connecting input map i with output map j, b_j^n is the bias of output map j, and $*$ is the valid 2D convolution. That is, for an input map $\mathbf{Y}^{n-1}$ of size $M_x^{n-1} \times M_y^{n-1}$ and a filter $\mathbf{W}$ of size $K_x^n \times K_y^n$ the output map $\mathbf{Y}^n$ is of size $M_x^n = M_x^{n-1} - K_x^n + 1$, $M_y^n = M_y^{n-1} - K_y^n + 1$. Note that the summation in Eq. (1) runs over all M^{n-1} input maps.

2.1.2. Max-pooling layer

The biggest architectural difference between our DNN and the CNN of LeCun et al. (1998) is the use of max-pooling layers (Riesenhuber & Poggio, 1999; Scherer et al., 2010; Serre et al., 2005) instead of sub-sampling layers. The output of a max-pooling layer is given by the maximum activation over non-overlapping rectangular regions of size $K_x \times K_y$. Max-pooling creates slight position invariance over larger local regions and down-samples the input image by a factor of K_x and K_y along each direction.

2.1.3. Classification layer

Kernel sizes of convolutional filters and max-pooling rectangles are chosen such that either the output maps of the last convolutional layer are down-sampled to 1 pixel per map, or a fully connected layer combines the outputs of the last convolutional layer into a 1D feature vector. The last layer is always a fully connected layer with one output unit per class in the recognition task. We use a softmax activation function for the last layer such that each neuron's output activation can be interpreted as the probability of a particular input image belonging to that class.

2.1.4. Training a single DNN

The training procedure of a single DNN is illustrated in Fig. 1b. A given dataset is preprocessed (P) before training starts, and then continually distorted (D) during training. Note that the preprocessing (details in Section 3.1) is not stochastic and is done for the whole dataset prior to training. Distortions on the other hand are stochastic and applied to each preprocessed image

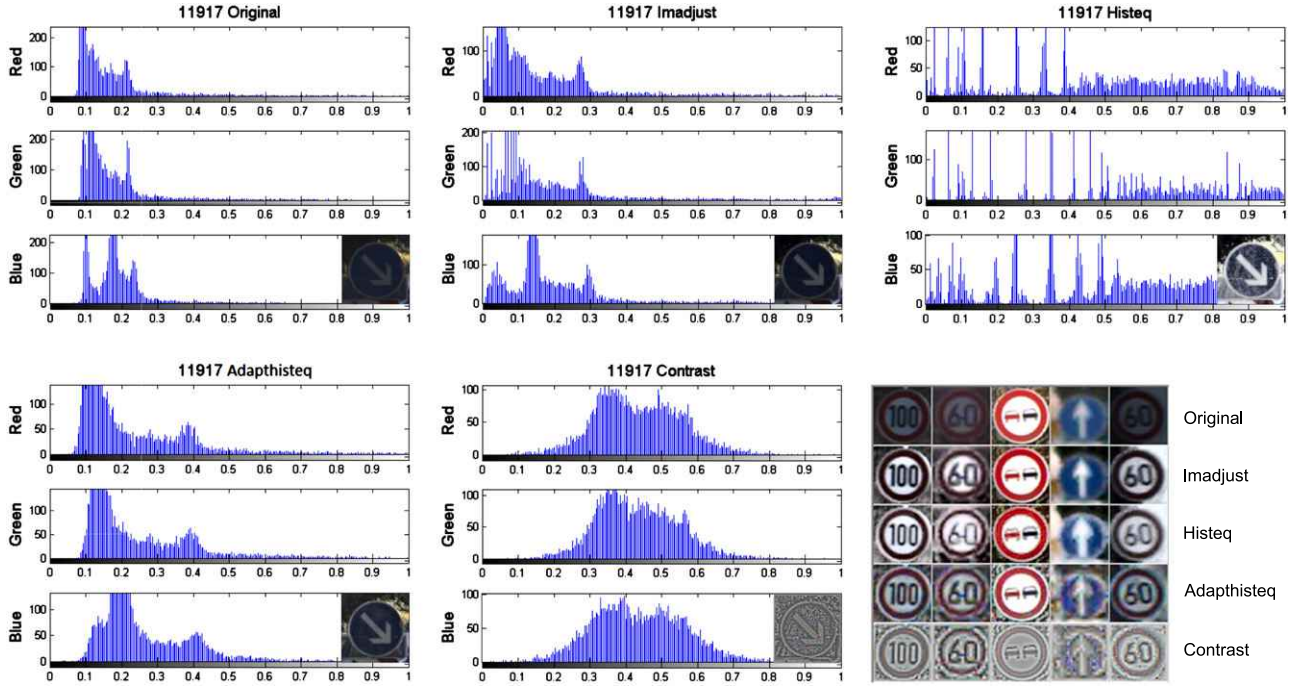

Fig. 2. Histogram of pixel intensities for image 11,917 from the test set of the preliminary phase of the competition, before and after normalization, as well as an additional selection of 5 traffic signs before and after normalization.

during training, using random but bounded values for translation, rotation and scaling. These values are drawn from a uniform distribution in a specified range, i.e. $\pm 10\%$ of the image size for translation, 0.9–1.1 for scaling and $\pm 5^\circ$ for rotation. The final, fixed sized image is obtained using bilinear interpolation of the distorted input image. These distortions allow us to train DNN with many free parameters without overfitting and greatly improve generalization performance (i.e. the error rate on the first phase of GTSRB decreases from 2.83% to 1.66%, Ciresan et al., 2011b). All DNN are trained using on-line gradient descent with an annealed learning rate.

2.1.5. Forming the MCDNN

Finally, we form an MCDNN by averaging the output activations of several DNN columns (Fig. 1c). For a given input pattern, the predictions of all columns are averaged. Before training, the weights of all columns are randomly initialized. Various columns can be trained on the same inputs, or on inputs preprocessed in different ways. If the errors of P different models have zero mean and are uncorrelated, the average error might be reduced by a factor of P simply by averaging the P models (Bishop, 2006). In practice, errors of models trained on similar data tend to be highly correlated. To avoid this problem, our MCDNN combines various DNN trained on differently normalized data. A key question is whether to optimize the combination of outputs of various models or not (Duin, 2002). Common problems during training include: (a) additional training data is required, and (b) there is no guarantee that the trained MCDNN generalize well to the unseen test data. For handwritten digits it was shown (Meier et al., 2011), that simply averaging the outputs of many DNN generalizes better on the test set than a linear combination of all the DNN with weights optimized over a validation set (Hashem & Schmeiser, 1995; Ueda, 2000). We therefore form the MCDNN by democratically averaging the outputs of each DNN.

3. Experiments

We use a system with a Core i7–950 (3.33 GHz), 24 GB DDR3, and four graphics cards of type GTX 580. Images from the training set might be translated, scaled and rotated, whereas only the undeformed, original or preprocessed images are used for validation. Training ends once the validation error is zero (usually after 15–30 epochs). Initial weights are drawn from a uniform random distribution in the range $[-0.05, 0.05]$. Each neuron's activation function is a scaled hyperbolic tangent (e.g. LeCun et al., 1998).

3.1. Data preprocessing

The original color images contain one traffic sign each, with a border of 10% around the sign. They vary in size from 15×15 to 250×250 pixels and are not necessarily square. The actual traffic sign is not always centered within the image; its bounding box is part of the annotations. The training set consists of 39,209 images; the test set of 12,630 images. We crop all images and process only the image within the bounding box. Our MCDNN implementation requires all training images to be of equal size. After visual inspection of the training image size distribution we resize all images to 48×48 pixels. As a consequence, the scaling factors along both axes are different for traffic signs with rectangular bounding boxes. Resizing forces them to have square bounding boxes.

High contrast variation among the images calls for contrast normalization. We use the following standard normalizations:

- *Image adjustment (Imadjust)* increases image contrast by mapping pixel intensities to new values such that 1% of the data is saturated at low and high intensities (MATLAB, 2010).
- *Histogram equalization (Histeq)* enhances contrast by transforming pixel intensities such that the output image histogram is roughly uniform (MATLAB, 2010).
- *Adaptive histogram equalization (Adapthisteq)* operates (unlike Histeq) on tiles rather than the entire image: the image is tiled in 8 nonoverlapping regions of 6×6 pixels each. Every tile's contrast is enhanced such that its histogram becomes roughly uniform (MATLAB, 2010).

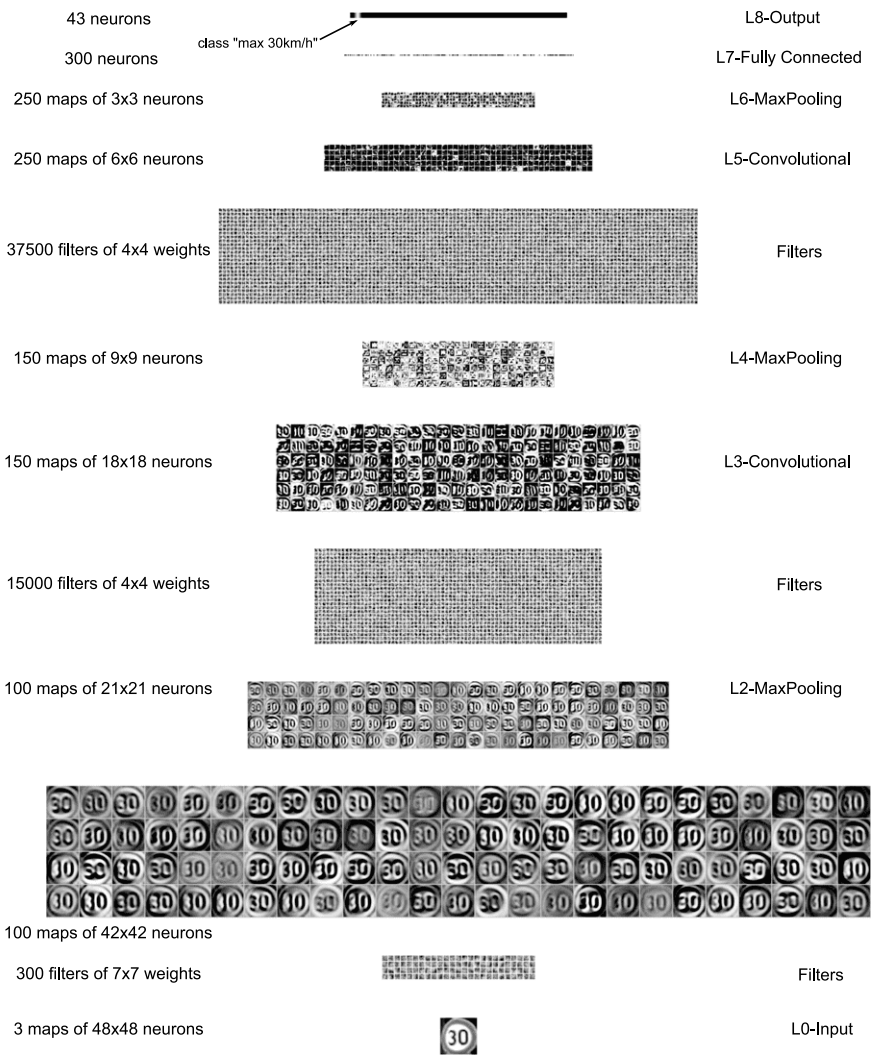

Fig. 3. DNN architecture from Table 1 together with all the activations and the learned filters. Only a subset of all the maps and filters are shown, the output layer is not drawn to scale and weights of fully connected layers are not displayed. For better contrast, the filters are individually normalized.

- *Contrast normalization (Conorm)* enhances edges through filtering the input image by a difference of Gaussians. We use a filter size of 5 × 5 pixels (Sermanet & LeCun, 2011).

Note that the above normalizations, except Conorm, are not performed in RGB-color space but rather in a color space that has image intensity as one of its components. For this purpose we transform the image from RGB- to Lab-space, perform the normalization and then transform the normalized image back to RGB-space. The effect of the four different normalizations is illustrated in Fig. 2, where histograms of pixel intensities together with original and normalized images are shown.

3.2. Results

Initial experiments with varying network depths showed that deep nets work better than shallow ones, consistent with our previous work on image classification (Ciresan, Meier, Gambardella, & Schmidhuber, 2010; Ciresan et al., 2011a). We report results for a single DNN with 9 layers (Table 1); the same architecture is shown in Fig. 3 where the activations of all layers together with the filters of a trained DNN are illustrated. Filters of the first layer are shown in color but consist in principle of three independent filters, each connected to the red, green and blue channel of the input image, respectively. The input layer has three maps of 48 × 48 pixels for each color channel; the output layer consists of 43 neurons, one per class. The used architecture has approximately 1.5 million free parameters, half of which are from the last two fully connected layers. It takes 37 h to train the MCDNN with 25 columns on four GPUs. After training, 87 images per second can be processed on a single GPU.

Fig. 4. The learned filters of the first convolutional layer of a DNN. The layer has 100 maps each connected to the three color channels of the input image for a total of 3 × 100 filters of size 15 × 15. Every displayed filter is the superposition of the 3 filters that are connected to the red, green and blue channel of the input image respectively. For better contrast, the filters are individually normalized. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

We also train a DNN with bigger filters, 15 × 15 instead of 7 × 7, in the first convolutional layer and plot them in Fig. 4. They are randomly initialized, and learn to respond to blobs, edges and other shapes in the input images. This illustrates that even the first layer

Table 1
9 layer DNN architecture.

Layer	Type	# Maps & neurons	Kernel
0	Input	3 maps of 48 × 48 neurons	
1	Convolutional	100 maps of 42 × 42 neurons	7 × 7
2	Max pooling	100 maps of 21 × 21 neurons	2 × 2
3	Convolutional	150 maps of 18 × 18 neurons	4 × 4
4	Max pooling	150 maps of 9 × 9 neurons	2 × 2
5	Convolutional	250 maps of 6 × 6 neurons	4 × 4
6	Max pooling	250 maps of 3 × 3 neurons	2 × 2
7	Fully connected	300 neurons	1 × 1
8	Fully connected	43 neurons	1 × 1

of a very deep (9 layers) DNN can be successfully trained by simple gradient descent, although it is usually the most problematic one (Hochreiter, Bengio, Frasconi, & Schmidhuber, 2001).

In total we trained 25 nets, 5 randomly initialized nets for each of the five datasets (i.e. original plus 4 different normalizations). The results are summarized in Table 2. Each column shows the recognition rates of 5 randomly initialized DNN. Mean and standard deviations are listed for each of the five distinct datasets as well as for all 25 DNN. The MCDNN results (but not the recognition rates) after averaging the outputs of all 25 DNN are shown as well. All individual DNN are better than any other method that entered the competition. Moreover, the resulting MCDNN with 25 DNN columns achieves a recognition rate of 99.46% and a drastic improvement with respect to any of the individual DNN.

Fig. 5 depicts all errors, plus ground truth and first and second predictions. Over 80% of the 68 errors are associated with correct second predictions. Erroneously predicted class probabilities tend to be very low – here the MCDNN is quite unsure about its classifications. In general, however, it is very confident – most of its predicted class probabilities are close to one or zero. Rejecting only 1% of all images (confidence below 0.51) results in an even lower error rate of 0.24%. To reach an error rate of 0.01% (a single misclassification), only 6.67% of the images have to be rejected (confidence below 0.94).

4. Conclusion

Our MCDNN won the German traffic sign recognition benchmark with a recognition rate of 99.46%, better than the one of humans on this task (98.84%), with three times fewer mistakes than the second best competing algorithm (98.31%). Forming a MCDNN from 25 nets, 5 per preprocessing method, increases the recognition rate from an average of 98.52%–99.46%. None of the preprocessing methods are superior in terms of single DNN recognition rates, but combining them into a MCDNN increases robustness to various types of noise and leads to more recognized traffic signs.

We plan to embed our method in a more general system that first localizes traffic signs in realistic scenes and then classifies them.

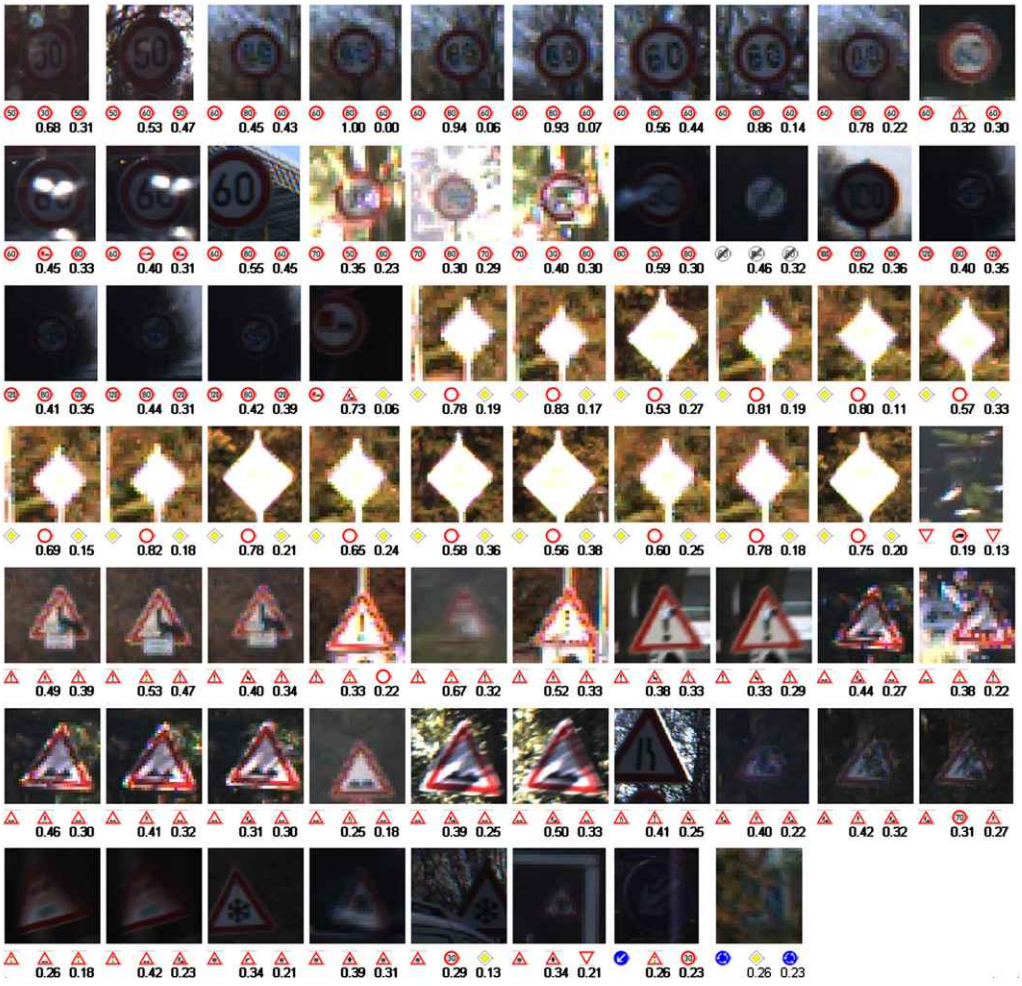

Fig. 5. The 68 errors of the MCDNN, with correct label (left) and first (middle) and second best (right) predictions. Best seen in color. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Recognition rates (%) of the MCDNN and its 25 DNN.

Trial	Original	Imadjust	Histeq	Adapthisteq	Conorm
1	98.56	98.39	98.80	98.47	98.63
2	98.16	98.58	98.27	98.47	98.33
3	98.64	98.77	98.51	98.51	98.46
4	98.46	98.61	98.31	98.53	98.62
5	98.54	98.77	98.58	98.58	98.66
Avg.	98.47 ± 0.18	98.62 ± 0.15	98.48 ± 0.22	98.50 ± 0.04	98.54 ± 0.14
	Average DNN recognition rate: 98.52 ± 0.15				
	MCDNN: 99.46				

Acknowledgment

This work was partially supported by a FP7-ICT-2009-6 EU Grant under Project Code 270247: A Neuro-dynamic Framework for Cognitive Robotics: Scene Representations, Behavioral Sequences, and Learning.

References

Behnke, S. (2003). *Lecture notes in computer science*: *Vol. 2766*. *Hierarchical neural networks for image interpretation*. Springer.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, *22*, 3207–3220.

Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2011). Convolutional neural network committees for handwritten character classification. In *International conference on document analysis and recognition* (pp. 1250–1254). IEEE.

Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011a). Flexible, high performance convolutional neural networks for image classification. In *International joint conference on artificial intelligence* (pp. 1237–1242). AAAI Press.

Ciresan, D. C., Meier, U., Masci, J., & Schmidhuber, J. (2011b). A committee of neural networks for traffic sign classification. In *International joint conference on neural networks* (pp. 1918–1921). IEEE.

Duin, R. P. W. (2002). The combining classifier: to train or not to train? In *International conference on pattern recognition* (pp. 765–770). IEEE.

Fukushima, K. (1980). Neocognitron: a self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, *36*, 193–202.

Hashem, S., & Schmeiser, B. (1995). Improving model accuracy using optimal linear combinations of trained neural networks. *Transactions on Neural Networks*, *6*, 792–794.

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer, & J. F. Kolen (Eds.), *A field guide to dynamical recurrent neural networks*. IEEE Press.

Hoyer, P. O., & Hyvärinen, A. (2000). Independent component analysis applied to feature extraction from colour and stereo images. *Network: Computation in Neural Systems*, *11*, 191–210.

Hubel, D. H., & Wiesel, T. (1962). Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, *160*, 106–154.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

Liu, C.-L., Yin, F., Wang, Q.-F., & Wang, D.-H. (2011). ICDAR 2011 Chinese handwriting recognition competition. In *International conference on document analysis and recognition* (pp. 1464–1469). IEEE.

MATLAB, (2010) Version 7.10.0 (R2010a). Natick, Massachusetts: The MathWorks Inc.

Meier, U., Ciresan, D. C., Gambardella, L. M., & Schmidhuber, J. (2011). Better digit recognition with a committee of simple neural nets. In *International conference on document analysis and recognition* (pp. 1135–1139). IEEE.

Mutch, J., & Lowe, D. G. (2008). Object class recognition and localization using sparse features with limited receptive fields. *International Journal of Computer Vision*, *56*, 503–511.

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an overcomplete basis set: a strategy employed by V1? *Vision Research*, *37*, 3311–3325.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*, 1019–1025.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks* (pp. 82–91). Springer.

Schmidhuber, J., Eldracher, M., & Foltin, B. (1996). Semilinear predictability minimization produces well-known feature detectors. *Neural Computation*, *8*, 773–786.

Sermanet, P., & LeCun, Y. (2011). Traffic sign recognition with multi-scale convolutional networks. In *International joint conference on neural networks* (pp. 2809–2813). IEEE.

Serre, T., Wolf, L., & Poggio, T. (2005). Object recognition with features inspired by visual cortex. In *Computer vision and pattern recognition conference* (pp. 994–1000). IEEE.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *International conference on document analysis and recognition* (pp. 958–963). IEEE.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011). The German traffic sign recognition benchmark: a multi-class classification competition. In *International joint conference on neural networks* (pp. 1453–1460). IEEE.

Ueda, N. (2000). Optimal linear combination of neural networks for improving classification performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*, 207–215.

Wiesel, D. H., & Hubel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *Journal of Physiology*, *148*, 574–591.