A Committee of Neural Networks for Traffic Sign Classification

Dan Cireşan, Ueli Meier, Jonathan Masci and Jürgen Schmidhuber

***Abstract*— We describe the approach that won the preliminary phase of the German traffic sign recognition benchmark with a better-than-human recognition rate of 98.98%. We obtain an even better recognition rate of 99.15% by further training the nets. Our fast, fully parameterizable GPU implementation of a Convolutional Neural Network does not require careful design of pre-wired feature extractors, which are rather learned in a supervised way. A CNN/MLP committee further boosts recognition performance.**

I. Introduction

THE most successful hierarchical visual object recognition systems extract localized features from input images, convolving image patches with filters whose responses are then repeatedly sub-sampled and re-filtered, resulting in a deep feed-forward network architecture whose output feature vectors are eventually classified. The Neocognitron [1] 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 [2], [3], [4]. These findings as well as experimental studies of the visual cortex justify the use of such filters in the so-called *standard model* for object recognition [5], [6], [7], whose filters are fixed, in contrast to those of Convolutional Neural Networks (CNNs) [8], [9], [10], whose weights (filters) are learned in a supervised way through back-propagation (BP).

To systematically test classification performance of various architectures, we developed a fast CNN implementation on Graphics Processing Units (GPUs). Most previous GPU-based CNN implementations [11], [12] were hard-coded to satisfy GPU hardware constraints, whereas ours is flexible and fully online (i.e., weight updates after each image). Other flexible implementations [13] are not fully exploiting the latest GPUs. It allows for training large CNNs within days instead of months, such that we can investigate the influence of various structural parameters by exploring large parameter spaces [14] and performing error analysis on repeated experiments.

Here we present results on the German traffic sign recognition benchmark [15], a 43 class, single-image classification challenge. We first give a brief description of our CNN, then describe the creation of the training set, and the data preprocessing. Finally we present experimental results and show how a committee of a CNN trained on raw pixels and an MLP trained on standard feature descriptors can boost recognition performance.

Dan Cireşan, Ueli Meier, Jonathan Masci and Jürgen Schmidhuber are with IDSIA, University of Lugano, SUPSI (email: {dan, ueli, jonathan, juergen}@idsia.ch).

This work was 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.

II. Convolutional neural networks

CNNs are hierarchical neural networks whose convolutional layers alternate with subsampling layers, reminiscent of simple and complex cells in the primary visual cortex [16]. CNNs vary in how convolutional and subsampling layers are realized and how they are trained. Here we give a brief description of the main building blocks (Fig. 1). A detailed description of the GPU implementation can be found in [17].

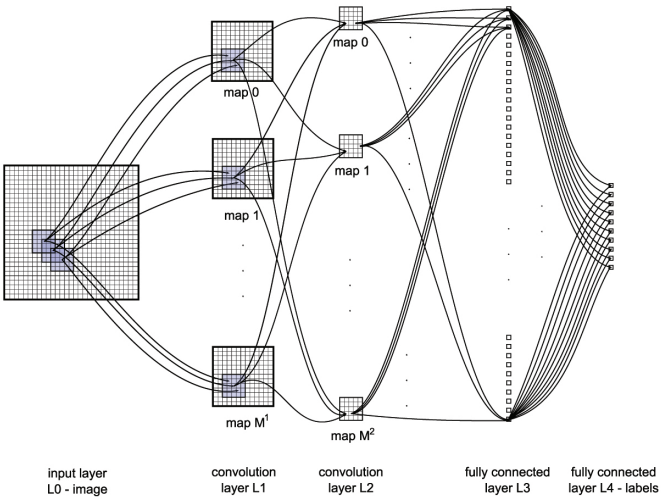

Fig. 1. Architecture of a convolutional neural network. Here the convolutional layers are fully connected. Both convolutional layers are using a 5 x 5 kernel and skipping factors of 1.

A. Convolutional layer

A convolutional layer is parametrized by: the number of maps, the size of the maps, kernel sizes and skipping factors. Each layer has M maps of equal size (M_x, M_y). A kernel (blue rectangle Fig 1) of size (K_x, K_y) is shifted over the valid region of the input image (that is, the kernel has to be completely inside the image). The skipping factors S_x and S_y define how many pixels the filter/kernel skips in x- and y-direction between subsequent convolutions. The output map size is then defined as:

$$M_x^n = \frac{M_x^{n-1} - K_x^n}{S_x^n + 1} + 1; \quad M_y^n = \frac{M_y^{n-1} - K_y^n}{S_y^n + 1} + 1 \quad (1)$$

where index n indicates the layer. Each map in layer L^n is connected to at most M^{n-1} maps in layer L^{n-1}. Neurons of a map share their weights, but have different input fields.

978-1-4244-9637-2/11/$26.00 ©2011 IEEE

B. Max-pooling layer

The biggest architectural difference of our implementation compared to the CNN of [8] is the use of a max-pooling layer [18] instead of a sub-sampling layer. In the implementation of [10] such layers are missing, and instead of a pooling or averaging operation, nearby pixels are simply skipped prior to the convolution. The output of the max-pooling layer is given by the maximum activation over non-overlapping rectangular regions of size (K_x, K_y). Max-pooling creates position invariance over larger local regions and down-samples the input image by a factor of K_x and K_y along each direction.

C. Classification layer

Kernel sizes of convolutional filters and max-pooling rectangles as well as skipping factors can be chosen such that the output maps of the last convolutional layer are down-sampled to 1 pixel per map. Alternatively, 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 soft max as the last layer's activation function, thus each neuron's output represents the class probability.

III. Experiments

We use a system with a Core i7-920 (2.66GHz), 12 GB DDR3, and four graphics cards: 2 x GTX 480 and 2 x GTX 580. Correctness of the implementation is checked by comparing the analytical gradient with the finite difference approximation of the gradient. Our plain feed-forward CNN architecture is trained using on-line gradient descent. Images from the training set might be translated, scaled and rotated, whereas only the original images are used for validation. Training ends once the validation error is zero (usually after 10 to 50 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 [8].

A. 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 26640 images; the test set of 12569 images. We crop all images and process only the bounding box. Our CNN 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 test three different types of normalization: 1) Pixels of all three color channels are linearly scaled to plus-minus one standard deviation around the average pixel intensity; 2) Pixels of all three color channels are linearly scaled to plus-minus two standard deviations around the average pixel intensity; 3) Contrast-limited Adaptive Histogram Equalization (CLAHE) [19]. We also create a gray-scale representation of the original color images. In total we perform experiments on 8 different datasets: the original, as well as sets resulting from three different normalizations, in color and gray-scale (Fig. 2).

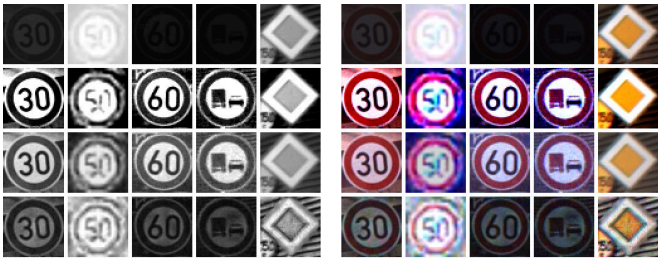

Fig. 2. Five gray-scale (left) and color (right) traffic signs normalized differently. Original images (first row), $\pm 1\sigma$ normalization (second row), $\pm 2\sigma$ normalization (third row) and CLAHE (fourth row).

B. CNNs

Initial experiments with different normalizations and varying network depths (4 to 7 hidden layers) showed that deep nets work better than shallow ones, consistent with our previous work on image classification [20], [17]. We report results for a single CNN with seven hidden layers (Table I). The input layer has either three maps of 48x48 pixels for each color channel, or a single map of 48x48 pixels for gray-scale images. The output layer has 43 neurons, one for each class.

TABLE I

The architecture of the convolutional neural network.

Layer	Type	# maps & neurons	kernel
0	input	1 or 3 maps of 48x48 neurons	
1	convolutional	100 maps of 46x46 neurons	3x3
2	max pooling	100 maps of 23x23 neurons	2x2
3	convolutional	150 maps of 20x20 neurons	4x4
4	max pooling	150 maps of 10x10 neurons	2x2
5	convolutional	250 maps of 8x8 neurons	3x3
6	max pooling	250 maps of 4x4 neurons	2x2
7	fully connected	200 neurons	
8	fully connected	43 neurons	

We summarize the result of various CNNs trained on gray-scale and color images in Tables II and III. The latter perform better, which might seem obvious, but the former also achieve highly competitive performance.

Additional translations, scalings and rotations of the training set considerably improve generalization. At the beginning of each epoch, each image in the training set is deformed using random but bounded values for translation, rotation and scaling (see Table III). Values for translation, rotation and scaling are drawn from a uniform distribution in a specified range, i.e. $\pm T\%$ of the image size for translation, $1 \pm S/100$ for scaling and $\pm R^\circ$ for rotation. The final image is obtained

TABLE II

ERROR RATES OF A CNN TRAINED ON PREPROCESSED GRAY-SCALE IMAGES. THE TRAINING DATA IS RANDOMLY TRANSLATED (T), SCALED (S) AND ROTATED (R).

Deformation			Test error rate[%]			
T[%]	S[%]	R[°]	no	$\pm 1\sigma$	$\pm 2\sigma$	CLAHE
0	0	0	3.43	3.65	3.18	2.73
5	0	0	2.28	2.32	1.79	1.77
5	10	10	2.10	2.42	1.82	1.53
10	5	5	2.13	1.97	1.74	1.55
10	10	10	1.79	2.02	1.42	**1.36**

using bilinear interpolation of the distorted input image. From all tried normalization methods, CLAHE yields the best result.

TABLE III

ERROR RATES OF A CNN TRAINED ON PREPROCESSED COLOR IMAGES. THE TRAINING DATA IS RANDOMLY TRANSLATED (T), SCALED (S) AND ROTATED (R).

Deformation			Test error rate[%]			
T[%]	S[%]	R[°]	no	$\pm 1\sigma$	$\pm 2\sigma$	CLAHE
0	0	0	2.83	2.98	2.78	2.32
5	0	0	1.76	2.11	1.91	1.42
5	10	10	1.41	1.99	1.61	1.86
10	5	5	1.88	1.80	1.85	1.42
10	10	10	1.66	1.88	1.58	**1.27**

In Fig. 3 we show the weights from the first layer of a CNN. We train a CNN with bigger filters (9x9) only for displaying purposes. The learned filters respond to blobs, edges and to other shapes in the input image.

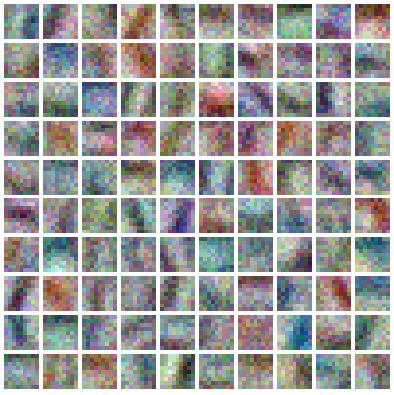

Fig. 3. The learned filters (kernels) of the first convolutional layer of a CNN. The layer has 100 maps each connected to all three R-G-B maps from the input layer through 300 9x9 kernels. Every displayed filter is the superposition of 3 filters corresponding to the R, G and B color channels.

C. Committee of a CNN and an MLP

We train various MLPs on the provided features, since the feature descriptors might offer complementary information with respect to the CNNs fed with raw pixel intensities. We only use HOG and HAAR features because recognition rates of MLPs trained on HUE features are inadequate. Table IV summarizes results of various MLPs with one and two hidden layers. The MLPs are trained in batch mode using a scaled conjugate gradient algorithm. The MLP architecture is not crucial; results for HOG features are very similar for the various architectures. Unsurprisingly, high-dimensional HAAR features (11584 dimensions) call for bigger MLPs. Hence HAAR-trained MLPs are also omitted from the committee.

TABLE IV

ERROR RATES [%] OF THREE DIFFERENT MLPS TRAINED ON THE PROVIDED HOG (HOG01, HOG02 AND HOG03), AND HAAR FEATURE DESCRIPTORS. MLP1: 1 HIDDEN LAYER WITH 200 HIDDEN UNITS; MLP2: 1 HIDDEN LAYER WITH 500 HIDDEN UNITS; MLP3 2 HIDDEN LAYERS WITH 500 AND 250 HIDDEN UNITS.

	HOG01	HOG02	HOG03	HAAR
MLP1	6.86	**4.55**	5.96	12.92
MLP2	6.77	4.58	5.78	12.34
MLP3	7.18	4.84	5.88	10.94

Since both CNNs and MLPs approximate posterior class probabilities, we can easily form a committee by averaging their outputs. In Table V we list committee results for MLPs trained on all three HOG feature descriptors and a CNN trained on randomly translated, scaled and rotated CLAHE color images (Tab. III). For all committees, a slight performance boost is observed, with the best committee reaching a recognition rate of 99.15%.

TABLE V

ERROR RATES [%] OF VARIOUS COMMITTEES OF CNNS LISTED IN TABLES II, III AND MLPS LISTED IN TABLE (IV).

	HOG01	HOG02	HOG03
MLP1 / CNN	0.95	0.92	1.01
MLP2 / CNN	0.95	1.00	0.97
MLP3 / CNN	0.95	0.96	**0.85**

In Figure 4 we plot the errors of the committee's CNN together with those of the best committee from Table V. Disappointingly, both the CNN and the committee wrongly classify many "no vehicles" traffic signs (class 15), which seem easy to recognize. Closer inspection of the CNNs output activations, however, reveals that almost all those signs are confused with the "no overtaking" sign (class 9). And indeed, CLAHE introduces artifacts that makes these two classes hard to distinguish. Most of the remaining errors are either due to bad illumination conditions, blurry images or destroyed traffic signs.

IV. CONCLUSIONS

The best of our CNNs for the German traffic sign recognition benchmark achieves a 98.73% recognition rate, avoiding the cumbersome computation of handcrafted features. Further improvements are obtained using a committee of a CNN and an MLP (up to 99.15% recognition rate). Whereas the CNNs are trained on raw pixel intensities, the MLPs are trained on feature descriptors offering complementary

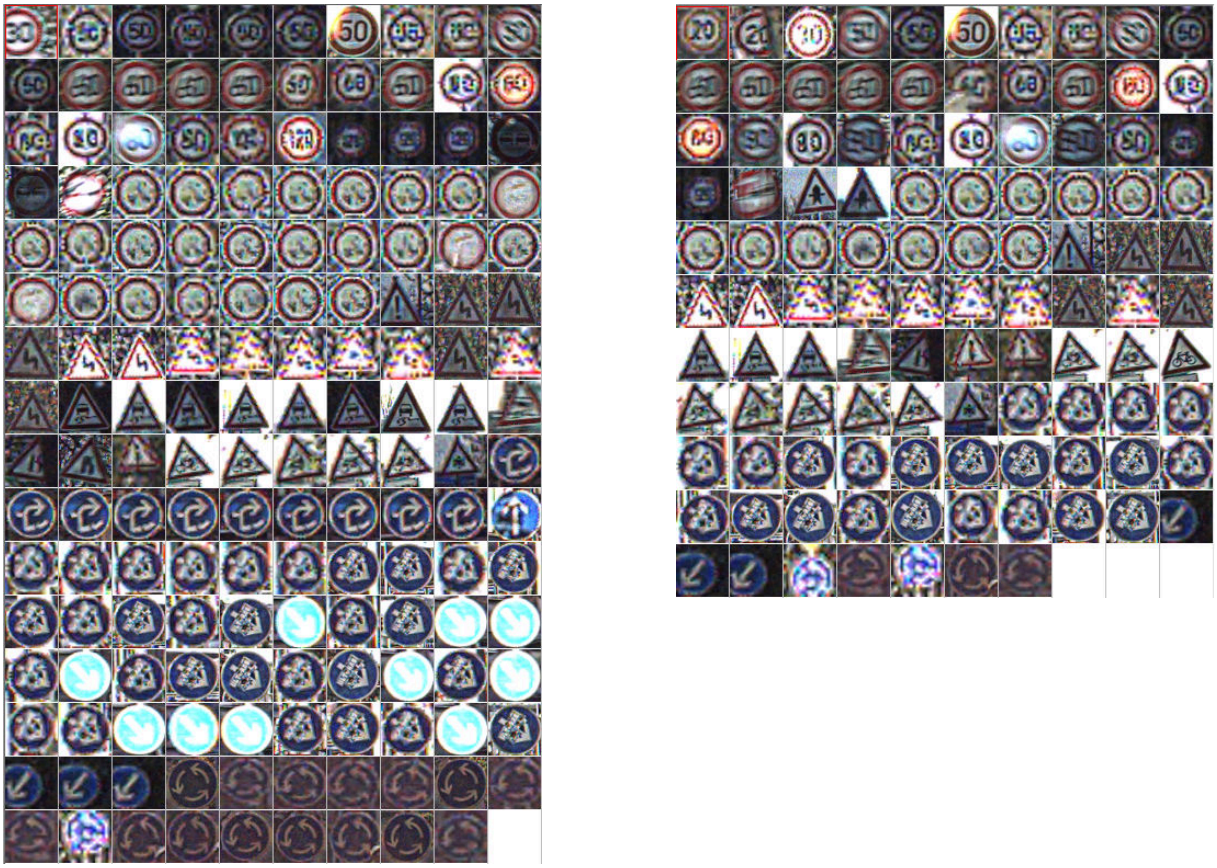

Fig. 4. CLAHE errors of the CNN used to build the best committee (left) together with the errors of the corresponding committee (right).

information. Although error rates of the best MLPs are 3-4% above those of the best CNNs, a committee consistently outperforms the individual classifiers.

REFERENCES

[1] K. Fukushima, "Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[2] J. Schmidhuber, M. Eldracher, and B. Foltin, "Semilinear predictability minimization produces well-known feature detectors," *Neural Computation*, vol. 8, no. 4, pp. 773–786, 1996.

[3] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, December 1997.

[4] P. O. Hoyer and A. Hyvärinen, "Independent component analysis applied to feature extraction from colour and stero images," *Network: Computation in Neural Systems*, vol. 11, no. 3, pp. 191–210, 2000.

[5] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nat. Neurosci.*, vol. 2, no. 11, pp. 1019–1025, 1999.

[6] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proc. of Computer Vision and Pattern Recognition Conference*, 2007.

[7] J. Mutch and D. G. Lowe, "Object class recognition and localization using sparse features with limited receptive fields," *Int. J. Comput. Vision*, vol. 56, no. 6, pp. 503–511, 2008.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[9] S. Behnke, *Hierarchical Neural Networks for Image Interpretation*, ser. Lecture Notes in Computer Science. Springer, 2003, vol. 2766.

[10] P. Simard, D. Steinkraus, and J. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Seventh International Conference on Document Analysis and Recognition*, 2003.

[11] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *International Workshop on Frontiers in Handwriting Recognition*, 2006.

[12] R. Uetz and S. Behnke, "Large-scale object recognition with CUDA-accelerated hierarchical neural networks," in *IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS)*, 2009.

[13] D. Strigl, K. Kofler, and S. Podlipnig, "Performance and scalability of GPU-based convolutional neural networks," in *18th Euromicro Conference on Parallel, Distributed, and Network-Based Processing*, 2010.

[14] N. Pinto, D. Doukhan, J. J. DiCarlo, and D. D. Cox, "A high-throughput screening approach to discovering good forms of biologically inspired visual representation." *PLoS computational biology*, vol. 5, no. 11, p. e1000579, Nov. 2009.

[15] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *International Joint Conference on Neural Networks*, 2011, accepted.

[16] D. H. Wiesel and T. N. Hubel, "Receptive fields of single neurones in the cat's striate cortex," *J. Physiol.*, vol. 148, pp. 574–591, 1959.

[17] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *International Joint Conference on Artificial Intelligence*, 2011, accepted.

[18] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International Conference on Artificial Neural Networks*, 2010.

[19] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[20] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep big simple neural nets for handwritten digit recognition," *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, 2010.