Multi-column Deep Neural Networks for Image Classification

Dan Cireşan, Ueli Meier and Jürgen Schmidhuber
IDSIA-USI-SUPSI
Galleria 2, 6928 Manno-Lugano, Switzerland
{dan,ueli,juergen}@idsia.ch

Abstract

Traditional methods of computer vision and machine learning cannot match human performance on tasks such as the recognition of handwritten digits or traffic signs. Our biologically plausible, wide and deep artificial neural network architectures can. Small (often minimal) receptive fields of convolutional winner-take-all neurons yield large network depth, resulting in roughly as many sparsely connected neural layers as found in mammals between retina and visual cortex. Only winner neurons are trained. Several deep neural columns become experts on inputs preprocessed in different ways; their predictions are averaged. Graphics cards allow for fast training. On the very competitive MNIST handwriting benchmark, our method is the first to achieve near-human performance. On a traffic sign recognition benchmark it outperforms humans by a factor of two. We also improve the state-of-the-art on a plethora of common image classification benchmarks.

1. Introduction

Recent publications suggest that unsupervised pre-training of deep, hierarchical neural networks improves supervised pattern classification [2, 10]. Here we train such nets by simple online back-propagation, setting new, greatly improved records on MNIST [19], Latin letters [13], Chinese characters [22], traffic signs [33], NORB (jittered, cluttered) [20] and CIFAR10 [17] benchmarks.

We focus on deep convolutional neural networks (DNN), introduced by [11], improved by [19], refined and simplified by [1, 32, 7]. Lately, DNN proved their mettle on data sets ranging from handwritten digits (MNIST) [5, 7], handwritten characters [6] to 3D toys (NORB) and faces [34]. DNNs fully unfold their potential when they are wide (many maps per layer) and deep (many layers) [7]. But training them requires weeks, months, even years on CPUs. High data transfer latency prevents multi-threading and multi-CPU code from saving the situation. In recent years, however, fast parallel neural net code for graphics cards (GPUs)

has overcome this problem. Carefully designed GPU code for image classification can be up to two orders of magnitude faster than its CPU counterpart [35, 34]. Hence, to train huge DNN in hours or days, we implement them on GPU, building upon the work of [5, 7]. The training algorithm is fully online, i.e. weight updates occur after each error back-propagation step. We will show that properly trained wide and deep DNNs can outperform all previous methods, and demonstrate that unsupervised initialization/pretraining is not necessary (although we don't deny that it might help sometimes, especially for datasets with few samples per class). We also show how combining several DNN columns into a Multi-column DNN (MCDNN) further decreases the error rate by 30-40%.

2. Architecture

The initially random weights of the DNN are iteratively trained to minimize the classification error on a set of labeled training images; generalization performance is then tested on a separate set of test images. Our architecture does this by combining several techniques in a novel way:

(1) Unlike the small NN used in many applications, which were either shallow [32] or had few maps per layer (LeNet7, [20]), ours are deep and have hundreds of maps per layer, inspired by the Neocognitron [11], with many (6-10) layers of non-linear neurons stacked on top of each other, comparable to the number of layers found between retina and visual cortex of macaque monkeys [3].

(2) It was shown [14] that such multi-layered DNN are hard to train by standard gradient descent [36, 18, 28], the method of choice from a mathematical/algorithmic point of view. Today's computers, however, are fast enough for this, more than 60000 times faster than those of the early 90s[1]. Carefully designed code for massively parallel graphics processing units (GPUs normally used for video games) allows for gaining an additional speedup factor of 50-100 over serial code for standard computers. Given enough labeled data, our networks do not need additional heuristics

---

[1] 1991 486DX-33 MHz, 2011 i7-990X 3.46 GHz

such as unsupervised pre-training [29, 24, 2, 10] or carefully prewired synapses [27, 31].

(3) The DNN of this paper (Fig. 1a) have 2-dimensional layers of winner-take-all neurons with overlapping receptive fields whose weights are shared [19, 1, 32, 7]. Given some input pattern, a simple max pooling technique [27] determines winning neurons by partitioning layers into quadratic regions of local inhibition, selecting the most active neuron of each region. The winners of some layer represent a smaller, down-sampled layer with lower resolution, feeding the next layer in the hierarchy. The approach is inspired by Hubel and Wiesel's seminal work on the cat's primary visual cortex [37], which identified orientation-selective *simple cells* with overlapping local receptive fields and *complex cells* performing down-sampling-like operations [15].

(4) Note that at some point down-sampling automatically leads to the first 1-dimensional layer. From then on, only trivial 1-dimensional winner-take-all regions are possible, that is, the top part of the hierarchy becomes a standard multi-layer perceptron (MLP) [36, 18, 28]. Receptive fields and winner-take-all regions of our DNN often are (near-)minimal, e.g., only 2x2 or 3x3 neurons. This results in (near-)maximal depth of layers with non-trivial (2-dimensional) winner-take-all regions. In fact, insisting on minimal 2x2 fields automatically defines the entire deep architecture, apart from the number of different convolutional kernels per layer [19, 1, 32, 7] and the depth of the plain MLP on top.

(5) Only winner neurons are trained, that is, other neurons cannot forget what they learnt so far, although they may be affected by weight changes in more peripheral layers. The resulting decrease of synaptic changes per time interval corresponds to biologically plausible reduction of energy consumption. Our training algorithm is fully online, i.e. weight updates occur after each gradient computation step.

(6) Inspired by microcolumns of neurons in the cerebral cortex, we combine several DNN columns to form a Multi-column DNN (MCDNN). Given some input pattern, the predictions of all columns are averaged:

$$y^i_{MCDNN} = \frac{1}{N} \sum_j^{\#columns} y^i_{DNN_j} \qquad (1)$$

where i corresponds to the ith class and j runs over all DNN. Before training, the weights (synapses) of all columns are randomly initialized. Various columns can be trained on the same inputs, or on inputs preprocessed in different ways. The latter helps to reduce both error rate and number of columns required to reach a given accuracy. The MCDNN architecture and its training and testing procedures are illustrated in Figure 1.
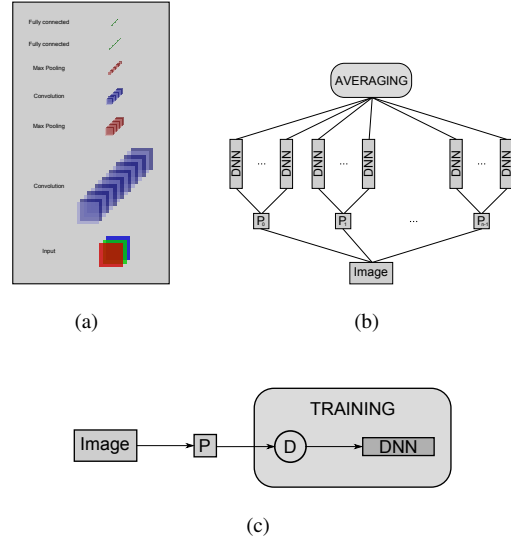

Figure 1. (a) DNN architecture. (b) MCDNN architecture. The input image can be preprocessed by $P_0 - P_{n-1}$ blocks. An arbitrary number of columns can be trained on inputs preprocessed in different ways. The final predictions are obtained by averaging individual predictions of each DNN. (c) Training a DNN. The dataset is preprocessed before training, then, at the beginning of every epoch, the images are distorted (D block). See text for more explanations.

3. Experiments

We evaluate our architecture on various commonly used object recognition benchmarks and improve the state-of-the-art on all of them. The description of the DNN architecture used for the various experiments is given in the following way: 2x48x48-100C5-MP2-100C5-MP2-100C4-MP2-300N-100N-6N represents a net with 2 input images of size 48x48, a convolutional layer with 100 maps and 5x5 filters, a max-pooling layer over non overlapping regions of size 2x2, a convolutional layer with 100 maps and 4x4 filters, a max-pooling layer over non overlapping regions of size 2x2, a fully connected layer with 300 hidden units, a fully connected layer with 100 hidden units and a fully connected output layer with 6 neurons (one per class). We use a scaled hyperbolic tangent activation function for convolutional and fully connected layers, a linear activation function for max-pooling layers and a softmax activation function for the output layer. All DNN are trained using on-line gradient descent with an annealed learning rate. During training, images are continually translated, scaled and rotated (even elastically distorted in case of characters), whereas only the original images are used for validation. Training ends once the validation error is zero or when the learning rate reaches its predetermined minimum. Initial weights are drawn from a uniform random distribution in the range $[-0.05, 0.05]$.

3.1. MNIST

The original MNIST digits [19] are normalized such that the width or height of the bounding box equals 20 pixels. Aspect ratios for various digits vary strongly and we therefore create six additional datasets by normalizing digit width to 10, 12, 14, 16, 18, 20 pixels. This is like seeing the data from different angles. We train five DNN columns per normalization, resulting in a total of 35 columns for the entire MCDNN. All 1x29x29-20C4-MP2-40C5-MP3-150N-10N DNN are trained for around 800 epochs with an annealed learning rate (i.e. initialized with 0.001 multiplied by a factor of 0.993/epoch until it reaches 0.00003). Training a DNN takes almost 14 hours and after 500 training epochs little additional improvement is observed. During training the digits are randomly distorted before each epoch (see Fig. 2a for representative characters and their distorted versions [7]). The internal state of a single DNN is depicted in Figure 2b, where a particular digit is forward propagated through a trained network and all activations together with the network weights are plotted.

Results of all individual nets and various MCDNN are summarized in Table 1. MCDNN of 5 nets trained with the same preprocessor achieve better results than their constituent DNNs, except for original images (Tab. 1). The MCDNN has a very low 0.23% error rate, improving state of the art by at least 34% [5, 7, 25] (Tab. 2). This is the first time an artificial method comes close to the ≈0.2% error rate of humans on this task [21]. Many of the wrongly classified digits either contain broken or strange strokes, or have wrong labels. The 23 errors (Fig. 2c) are associated with 20 correct second guesses.

We also trained a single DNN on all 7 datasets simultaneously which yielded worse result (0.52%) than both MCDNN and their individual DNN. This shows that the improvements come from the MCDNN and not from using more preprocessed data.

Table 2. Results on MNIST dataset.

Method	Paper	Error rate[%]
CNN	[32]	0.40
CNN	[26]	0.39
MLP	[5]	0.35
CNN committee	[6]	0.27
MCDNN	this	**0.23**

How are the MCDNN errors affected by the number of preprocessors? We train 5 DNNs on all 7 datasets. A MCDNN 'y out-of-7' (y from 1 to 7) averages 5y nets trained on y datasets. Table 3 shows that more preprocessing results in lower MCDNN error.

We also train 5 DNN for each odd normalization, i.e. W11, W13, W15, W17 and W19. The 60-net MCDNN performs (0.24%) similarly to the 35-net MCDNN, indicat-
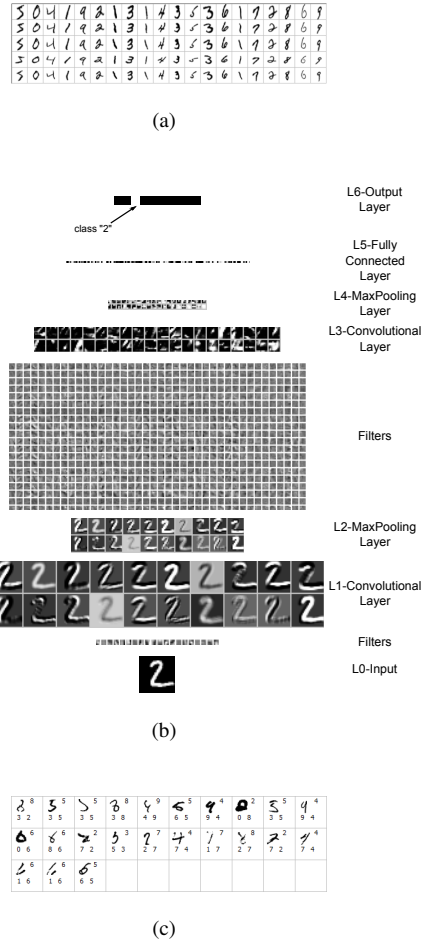

(a)

(b)

(c)

Figure 2. (a) Handwritten digits from the training set (top row) and their distorted versions after each epoch (second to fifth row). (b) DNN architecture for MNIST. Output layer not drawn to scale; weights of fully connected layers not displayed. (c) The 23 errors of the MCDNN, with correct label (up right) and first and second best predictions (down left and right).

ing that additional preprocessing does not further improve recognition.

Table 3. Average test error rate [%] of MCDNN trained on y preprocessed datasets.

y	# MCDNN	Average Error[%]
1	7	0.33±0.07
2	21	0.27±0.02
3	35	0.27±0.02
4	35	0.26±0.02
5	21	0.25±0.01
6	7	0.24±0.01
7	1	0.23

We conclude that MCDNN outperform DNN trained on the same data, and that different preprocessors further decrease the error rate.

Table 1. Test error rate [%] of the 35 NNs trained on MNIST. Wxx - width of the character is normalized to xx pixels

Trial	W10	W12	W14	W16	W18	W20	ORIGINAL
1	0.49	0.39	0.40	0.40	0.39	0.36	0.52
2	0.48	0.45	0.45	0.39	0.50	0.41	0.44
3	0.59	0.51	0.41	0.41	0.38	0.43	0.40
4	0.55	0.44	0.42	0.43	0.39	0.50	0.53
5	0.51	0.39	0.48	0.40	0.36	0.29	0.46
avg.	0.52±0.05	0.44±0.05	0.43±0.03	0.40±0.02	0.40±0.06	0.39±0.08	0.47±0.05
			35-net average error: 0.44±0.06				
5 columns MCDNN	0.37	0.26	0.32	0.33	0.31	0.26	0.46
			35-net MCDNN: **0.23%**				

3.2. NIST SD 19

The 35-columns MCDNN architecture and preprocessing used for MNIST are also applied to Latin characters from NIST SD 19 [13]. For all tasks our MCDNN achieves recognition rates 1.5-5 times better than any published result (Tab. 4). In total there are 82000 characters in the test set, but there are many more easy to classify digits (58000) than hard to classify letters (24000). This explains the lower overall error rate of the 62-class problem compared to the 52-class letters problem. From all errors of the 62-class problem 3% of the 58000 digits are misclassified and 33% of the 24000 letters are misclassified. Letters are in general more difficult to classify, but there is also a higher amount of confusion between similar lower- and upper-case letters such as i,I and o,O for example. Indeed, error rates for the case insensitive task drop from 21% to 7.37%. If the confused upper- and lower-case classes are merged, resulting in 37 different classes, the error rate is only slightly bigger (7.99%). Upper-case letters are far easier to classify (1.83% error rate) than lowercase letters (7.47%) due to the smaller writer dependent in-class variability. For a detailed analysis of all the errors and confusions between different classes, the confusion matrix is most informative (Supplementary material Fig. S1).

Table 4. Average error rates of MCDNN for all experiments, plus results from the literature. * case insensitive

Data (task)	MCDNN error [%]	Published results Error[%] and paper	
all (62)	**11.63**		
digits (10)	**0.77**	3.71 [12]	1.88 [23]
letters (52)	**21.01**	30.91[16]	
letters* (26)	**7.37**	13.00 [4]	13.66[16]
merged (37)	**7.99**		
uppercase (26)	**1.83**	10.00 [4]	6.44 [9]
lowercase (26)	**7.47**	16.00 [4]	13.27 [16]

3.3. Chinese characters

Compared to Latin character recognition, isolated Chinese character recognition is a much harder problem, mainly because of the much larger category set, but also because of wide variability of writing styles, and the confusion between similar characters. We use a dataset from the Institute of Automation of Chinese Academy of Sciences (CASIA [22]), which contains 300 samples for each of 3755 characters (in GB1 set). This resulted in a data set with more than 1 Million characters (3GB of data) which posed a major computational challenge even to our system. Without our fast GPU implementation the nets on this task would train for more than one year. Only the forward propagation of the training set takes 27h on a normal CPU, and training a single epoch would consequently have lasted several days. On our fast GPU implementation on the other hand, training a single epoch takes 3.4h, which makes it feasible to train a net within a few days instead of many months.

We train following DNN, 1x48x48-100C3-MP2-200C2-MP2-300C2-MP2-400C2-MP2-500N-3755N, on offline as well as on online characters. For the offline character recognition task, we resize all characters to 40x40 pixels and place them in the center of a 48x48 image. The contrast of each image is normalized independently.

As suggested by the organizers, the first 240 writers from the database CASIA-HWDB1.1 are used for training and the remaining 60 writers are used for testing. The total numbers of training and test characters are 938679 and 234228, respectively.

For the online dataset, we draw each character from its list of coordinates, resize the resulting images to 40x40 pixels and place them in the center of a 48x48 image. Additionally, we smooth-out the resulting images with a Gaussian blur filter over a 3x3 pixel neighborhood and uniform standard deviation of 0.75. As suggested by the organizers, the characters of 240 writers from database CASIA-

OLHWDB1.1 are used for training the classifier and the characters of the remaining 60 writers are used for testing. The resulting numbers of training and test characters are 939564 and 234800, respectively.

All methods previously applied to this dataset perform some feature extraction followed by a dimensionality reduction, whereas our method directly works on raw pixel intensities and learns the feature extraction and dimensionality reduction in a supervised way. On the offline task we obtain an error rate of **6.5%** compared to 10.01% of the best method [22]. Even though much information is lost when drawing a character from it's coordinate sequence, we obtain a recognition rate of **5.61%** on the online task compared to 7.61% of the best method [22].

We conclude that on this very hard classification problem, with many classes (3755) and relatively few samples per class (240), our fully supervised DNN beats the current state-of-the-art methods by a large margin.

3.4. NORB

We test a MCDNN with four columns on NORB (jittered-cluttered) [20], a collection of stereo images of 3D models (Figure 3). The objects are centrally placed on randomly chosen backgrounds, and there is also cluttering from a peripherally placed second object. This database is designed for experimenting with 3D object recognition from shape. It contains images of 50 toys belonging to 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. The objects were imaged by two cameras under 6 lighting conditions, 9 elevations (30 to 70 degrees every 5 degrees), and 18 azimuths (0 to 340 every 20 degrees). The training set has 10 folds of 29160 images each for a total of 291600 images; the testing set consists of two folds totaling 58320 images.
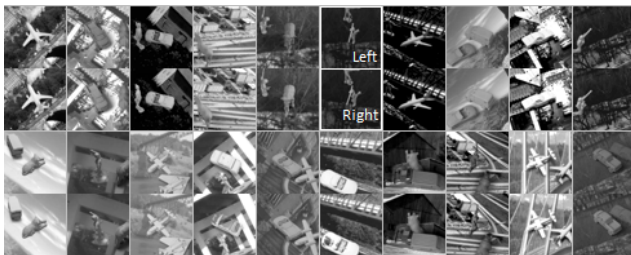

Figure 3. Twenty NORB stereo images (left image - up, right image - down).

No preprocessing is used for this dataset. We scale down images from the original 108x108 to 48x48 pixels. This size is big enough to preserve the details present in images and small enough to allow fast training. We perform two rounds of experiments, using only the first two folds (to compare with previous results that do not use the entire training data)

Table 5. Error rates, averages and standard deviations over 4 runs of a 9 layer DNN on the NORB test set.

training set size	errors for 4 runs [%]				mean[%]
first 2 folds	4.49	4.71	4.82	4.85	4.72 ± 0.16
	4-net MCDNN error: 3.57%				
all 10 folds	3.32	3.18	3.73	3.36	3.40 ± 0.23
	4-net MCDNN error: **2.70%**				

previous state of the art: 5.00% - [8]; 5.60% - [30]

and using all training data.

We tested several distortion parameters with small nets and found that maximum rotation of $15°$, maximum translation of 15% and maximum scaling of 15% are good choices, hence we use them for all NORB experiments.

To compare to previous results, we first train only on the first 2-folds of the data. The net architecture is deep, but has few maps per layer: 2x48x48-50C5-MP2-50C5-MP2-50C4-MP2-300N-100N-6N. The learning rate setup is: eta start 0.001; eta factor 0.95; eta stop 0.000003. Due to small net size, training is fast at 156s/epoch for 114 epochs. Testing one sample requires 0.5ms. Even when we use less data to train, the MCDNN greatly improves the state of the art from 5% to 3.57% (Table 5).

Our method is fast enough to process the entire training set though. We use the same architecture but double the number of maps when training with all 10 folds: 2x48x48-100C5-MP2-100C5-MP2-100C4-MP2-300N-100N-6N. The learning rate setup remains the same. Training time increases to 34min/epoch because the net is bigger, and we use five times more data. Testing one sample takes 1.3ms. All of this pays off, resulting in a very low 2.70% error rate, further improving the state of the art.

Although NORB has only six classes, training and test instances sometimes differ greatly, making classification hard. More than 50% of the errors are due to confusions between cars and trucks. Considering second predictions, too, the error rate drops from 2.70% to 0.42%, showing that 84% of the errors are associated with a correct second prediction.

3.5. Traffic signs

Recognizing traffic signs is essential for the automotive industry's efforts in the field of driver's assistance, and for many other traffic-related applications. We use the GTSRB traffic sign dataset [33].

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 within the

bounding box. Our DNN implementation requires all training images to be of equal size. After visual inspection of the image size distribution we resize all images to 48×48 pixels. As a consequence, scaling factors along both axes are different for traffic signs with rectangular bounding boxes. Resizing forces them to have square bounding boxes.

Our MCDNN is the only artificial method to outperform humans, who produced twice as many errors. Since traffic signs greatly vary in illumination and contrast, standard image preprocessing methods are used to enhance/normalize them (Fig. 4a and supplementary material). For each dataset five DNN are trained (architecture: 3x48x48-100C7-MP2-150C4-150MP2-250C4-250MP2-300N-43N), resulting in a MCDNN with 25 columns, achieving an error rate of **0.54%** on the test set. Figure 4b 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% percent 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). Our method outperforms the second best algorithm by a factor of 3. It takes 37 hours to train the MCDNN with 25 columns on four GPUs. The trained MCDNN can check 87 images per second on one GPU (and 2175 images/s/DNN).

3.6. CIFAR 10

CIFAR10 is a set of natural color images of 32x32 pixels [17]. It contains 10 classes, each with 5000 training samples and 1000 test samples. Images vary greatly within each class. They are not necessarily centered, may contain only parts of the object, and show different backgrounds. Subjects may vary in size by an order of magnitude (i.e., some images show only the head of a bird, others an entire bird from a distance). Colors and textures of objects/animals also vary greatly.

Our DNN input layers have three maps, one for each color channel (RGB). We use a 10-layer architecture with very small kernels: 3x32x32-300C3-MP2-300C2-MP2-300C3-MP2-300C2-MP2-300N-100N-10N. Just like for MNIST, the initial learning rate 0.001 decays by a factor of 0.993 after every epoch. Transforming CIFAR color images to gray scale reduces input layer complexity but increases error rates. Hence we stick to the original color images. As for MNIST, augmenting the training set with randomly (by at most 5%) translated images greatly decreases the error from 28% to 20% (the NN-inherent local translation invariance by itself is not sufficient). By additional
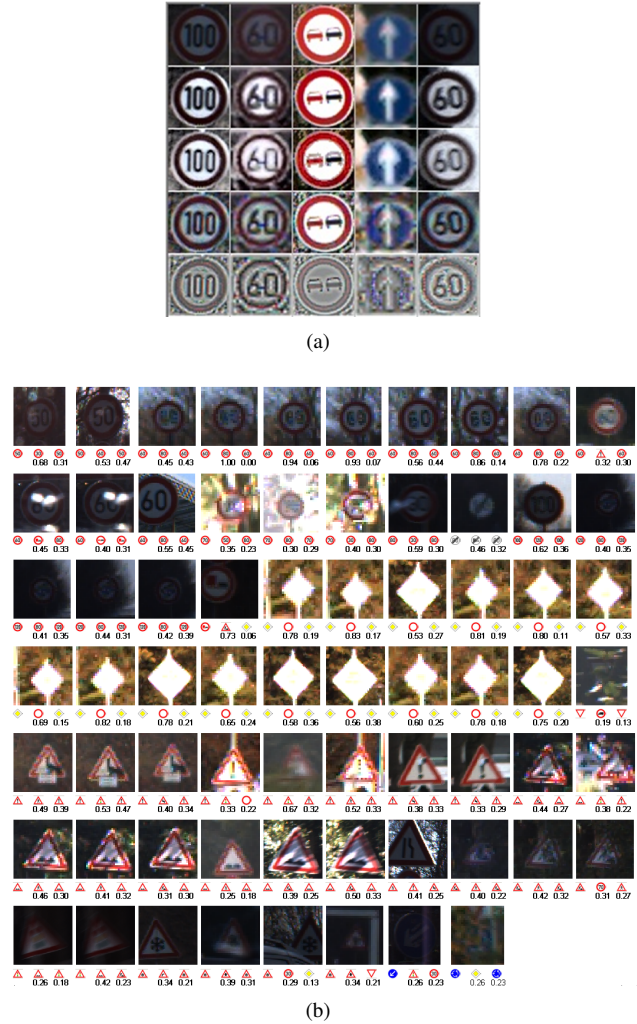

(a)

(b)

Figure 4. (a) Preprocessed images, from top to bottom: original, Imadjust, Histeq, Adapthisteq, Conorm. (b) The 68 errors of the MCDNN, with correct label (left) and first and second best predictions (middle and right).

Table 6. Error rates, averages and standard deviations for 10 runs of a 10 layer DNN on the CIFAR10 test set. The nets in the first row are trained on preprocessed images (see traffic sign preprocessing), whereas those in the second row are trained on original images.

preprocessing	errors for 8 runs [%]				mean[%]
yes	16.47	19.20	19.72	20.31	18.93 ± 1.69
no	15.63	15.85	16.13	16.05	15.91 ± 0.22
	8-net average error: $17.42 \pm 1.96\%$				
	8-net MCDNN error: **11.21%**				
	previous state of the art: 18.50% - [8]; 19.51% - [7]				

scaling (up to $\pm 15\%$), rotation (up to $\pm 5°$), and up to $\pm 15\%$ translation, the individual net errors decrease by another 3% (Tab. 6). The above small maximal bounds prevent loss of too much information leaked beyond the 32×32 pixels rectangle.
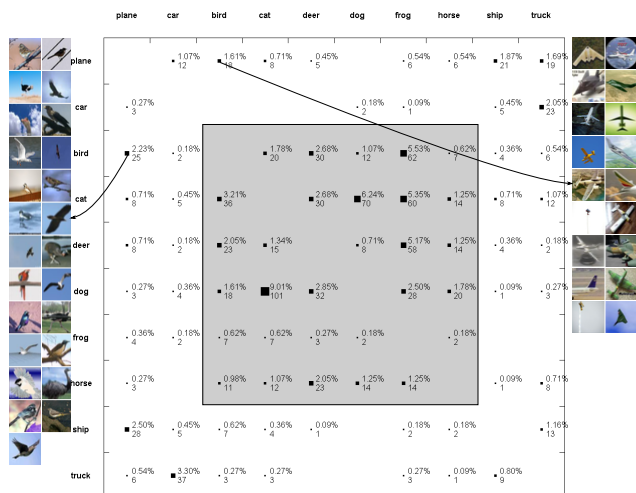
Figure 5. Confusion matrix for the CIFAR10 MCDNN: correct labels on vertical axis; detected labels on horizontal axis. Square areas are proportional to error numbers, shown both as relative percentages of the total error number, and in absolute value. Left - images of all birds classified as planes. Right - images of all planes classified as birds. Confusion sub-matrix for animal classes has a gray background.

We repeat the experiment with different random initializations and compute mean and standard deviation of the error, which is rather small for original images, showing that our DNN are robust. Our MCDNN obtains a very low error rate of 11.21%, greatly rising the bar for this benchmark.

The confusion matrix (Figure 5) shows that the MCDNN almost perfectly separates animals from artifacts, except for planes and birds, which seems natural, although humans easily distinguish almost all the incorrectly classified images, even if many are cluttered or contain only parts of the objects/animals (see false positive and false negative images in Figure 5). There are many confusions between different animals; the frog class collects most false positives from other animal classes, with very few false negatives. As expected, cats are hard to tell from dogs, collectively causing 15.25% of the errors. The MCDNN with 8 columns (four trained on original data and one trained for each preprocessing used also for traffic signs) reaches a low 11.21% error rate, far better than any other algorithm.

4. Conclusion

This is the first time human-competitive results are reported on widely used computer vision benchmarks. On many other image classification datasets our MCDNN improves the state-of-the-art by 30-80% (Tab. 7). We drastically improve recognition rates on MNIST, NIST SD 19, Chinese characters, traffic signs, CIFAR10 and NORB. Our method is fully supervised and does not use any additional unlabeled data source. Single DNN already are sufficient to obtain new state-of-the-art results; combining them into MCDNNs yields further dramatic performance boosts.

Table 7. Results and relative improvements on different datasets.

Dataset	Best result of others [%]	MCDNN [%]	Relative improv. [%]
MNIST	0.39	0.23	41
NIST SD 19	see Table 4	see Table 4	30-80
HWDB1.0 on.	7.61	5.61	26
HWDB1.0 off.	10.01	6.5	35
CIFAR10	18.50	11.21	39
traffic signs	1.69	0.54	72
NORB	5.00	2.70	46

Acknowledgment

References

[1] S. Behnke. *Hierarchical Neural Networks for Image Interpretation*, volume 2766 of *Lecture Notes in Computer Science*. Springer, 2003. 1, 2

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Neural Information Processing Systems*, 2007. 1, 2

[3] N. P. Bichot, A. F. Rossi, and R. Desimone. Parallel and serial neural mechanisms for visual search in macaque area V4. *Science*, 308:529–534, 2005. 1

[4] P. R. Cavalin, A. de Souza Britto Jr., F. Bortolozzi, R. Sabourin, and L. E. S. de Oliveira. An implicit segmentation-based method for recognition of handwritten strings of characters. In *SAC*, pages 836–840, 2006. 4

[5] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220, 2010. 1, 3

[6] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Convolutional neural network committees for handwritten character classification. In *International Conference on Document Analysis and Recognition*, pages 1250–1254, 2011. 1, 3

[7] 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*, pages 1237–1242, 2011. 1, 2, 3, 6

[8] A. Coates and A. Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning*, 2011. 5, 6

[9] E. M. Dos Santos, L. S. Oliveira, R. Sabourin, and P. Maupin. Overfitting in the selection of classifier ensembles: a comparative study between pso and ga. In *Conference on Genetic and Evolutionary Computation*, pages 1423–1424. ACM, 2008. 4

[10] A. C. P.-A. M. P. V. Dumitru Erhan, Yoshua Bengio and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660, 2010. 1, 2

[11] K. Fukushima. Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980. 1

[12] E. Granger, P. Henniges, and R. Sabourin. Supervised Learning of Fuzzy ARTMAP Neural Networks Through Particle Swarm Optimization. *Pattern Recognition*, 1:27–60, 2007. 4

[13] P. J. Grother. NIST special database 19 - Handprinted forms and characters database. Technical report, National Institute of Standards and Thechnology (NIST), 1995. 1, 4

[14] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001. 1

[15] D. H. Hubel and T. Wiesel. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, 160:106–154, 1962. 2

[16] A. L. Koerich and P. R. Kalva. Unconstrained handwritten character recognition using metaclasses of characters. In *Intl. Conf. on Image Processing*, pages 542–545, 2005. 4

[17] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Computer Science Department, University of Toronto, 2009. 1, 6

[18] Y. LeCun. Une procédure d'apprentissage pour réseau a seuil asymmetrique (a learning scheme for asymmetric threshold networks). In *Proceedings of Cognitiva 85*, pages 599–604, Paris, France, 1985. 1, 2

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. 1, 2, 3

[20] Y. LeCun, F.-J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition*, 2004. 1, 5

[21] Y. LeCun, L. D. Jackel, L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, U. A. Muller, E. Sackinger, P. Simard, and V. Vapnik. Learning algorithms for classification: A comparison on handwritten digit recognition. In J. H. Oh, C. Kwon, and S. Cho, editors, *Neural Networks: The Statistical Mechanics Perspective*, pages 261–276. World Scientific, 1995. 3

[22] C.-L. Liu, F. Yin, D.-H. Wang, and Q.-F. Wang. Chinese Handwriting Recognition Contest. In *Chinese Conference on Pattern Recognition*, 2010. 1, 4, 5

[23] J. Milgram, M. Cheriet, and R. Sabourin. Estimating accurate multi-class probabilities with support vector machines. In *Int. Joint Conf. on Neural Networks*, pages 1906–1911, 2005. 4

[24] M. Ranzato, Y.-L. B. Fu Jie Huang, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. of Computer Vision and Pattern Recognition Conference*, 2007. 2

[25] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'07)*. IEEE Press, 2007. 3

[26] M. A. Ranzato, C. Poultney, S. Chopra, and Y. Lecun. Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems (NIPS 2006)*, 2006. 3

[27] M. Riesenhuber and T. Poggio. Hierarchical models of object recognition in cortex. *Nat. Neurosci.*, 2(11):1019–1025, 1999. 2

[28] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. 1, 2

[29] R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighborhood structure. In *Proc. of the International Conference on Artificial Intelligence and Statistics*, volume 11, 2007. 2

[30] 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. 5

[31] T. Serre, L. Wolf, S. M. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(3):411–426, 2007. 2

[32] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition*, pages 958–963, 2003. 1, 2, 3

[33] 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. 1, 5

[34] D. Strigl, K. Kofler, and S. Podlipnig. Performance and scalability of gpu-based convolutional neural networks. *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, 0:317–324, 2010. 1

[35] 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. 1

[36] P. J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* PhD thesis, Harvard University, 1974. 1, 2

[37] D. H. Wiesel and T. N. Hubel. Receptive fields of single neurones in the cat's striate cortex. *J. Physiol.*, 148:574–591, 1959. 2