Image Classification with Growing Neural Networks

Iveta Mrazova and Marek Kukacka

***Abstract*—Future multi-media technologies are expected to support efficient on-line processing of huge amounts of high-dimensional data without any special pre-processing. In this paper, we will introduce a new model of the so-called Growing Hierarchical Neural Networks (GHNN) applicable to image classification without requiring advanced domain-specific feature extraction techniques. It can be, moreover, supposed that the involved dynamic data-dependent adjustment of both the number and position of the neurons improves generalization. Experimental results obtained so far for two case studies on face and hand-written digit recognition show that local features detected automatically by GHNN-networks impact a transparent and compact representation of the extracted knowledge.**

***Index Terms*—Convolutional neural networks, image classification, face recognition, self-organization.**

I. Introduction

Traditional pattern recognition systems consist of two main modules – a feature detector and a classifier. The feature detector should extract from the data low-dimensional vectors invariant to transformations and distortions. The classifier is usually general-purpose and trainable, yet oriented towards low-dimensional data with easily separable classes. With regard to the vast variety of images to be handled automatically within the framework of future multi-media technologies, an urgent need for robust pre-processing techniques is obvious. Recently, e.g. Facebook provides a new automatic routine to detect faces in photos [1].

Promising alternatives to traditional pattern recognition systems are based on a hierarchical combination of simple locally detected features, their position and orientation to achieve invariant image categorization [2]-[4]. In particular, the massively parallel kind of low-level feature extraction impacts a superior performance of Convolutional Neural Networks (CNNs) [5]. Anyway, the training process of CNNs is often quite time-consuming due to a relatively high complexity of the entire model. With the aim to speed-up their training and enhance detection of relevant image features, we will introduce a new type of hierarchical self-organizing networks - the so-called Growing Hierarchical Neural Networks (GHNN).

Manuscript received September 20, 2012; revised December 7, 2012. This research was partially supported by the Czech Science Foundation under Grant No. P103/10/0783, P202/10/1333, Grant No. 136109 of GAUK, and by Grant No. 201/09/H057 of GA ČR.

The authors are with the Department of Theoretical Computer Science and Mathematical Logic, Faculty of Mathematics and Physics, Charles University, Malostranske nam. 25, 118 00 Praha, Czech Republic (e-mail: iveta.mrazova@mff.cuni.cz, mkukacka@gmail.com).

GHNNs are inspired by CNNs but benefit from their ability to adjust dynamically the number of local image features to be used later on by the network. The following section analyzes therefore neural network paradigms related to dynamic feature extraction. The next two sections introduce the new models of hybrid self-organizing networks and GHNN networks applicable to efficient image processing. Afterwards, the results of supporting experiments (involving two large image data sets) will be discussed. The concluding section summarizes the achieved results.

II. Related Works

CNN-networks [5] are known to outperform all other neural network paradigms when used for 2D-image recognition with minimum or no advanced pre-processing. The principles of weight sharing and local receptive fields keep down the number of trainable parameters in CNNs. With local receptive fields, perceptron-like neurons can extract elementary visual features such as oriented edges or corners. Local receptive fields may or may not overlap. The extracted features are then combined by the subsequent layers in order to detect more complex higher-order features.

In each layer, the neurons are organized in the so-called feature maps. All neurons in a feature map are constrained to perform the same operation, however, on different parts of the image and share thus the same set of weights. While training the network, the respective weight adjustments are summed up and applied to the entire shared weight vector. If the input image is shifted, the feature map output will be shifted by the same amount, but will be left unchanged otherwise. Different feature maps usually have different weight vectors so that multiple features can be extracted at each location.

Once a feature is detected by the feature-extracting layer, a layer of sub-sampling neurons performs a kind of local averaging to blur the exact position of the feature. This makes the network less sensitive to the exact position and form of the processed patterns. Successive convolutional and sub-sampling layers are typically alternated: at each layer, the number of feature maps is increased as the spatial resolution is decreased. Each neuron from higher layers may have input connections from several feature maps in the preceding sub-sampling layer. Attached to the entire feature detector, CNNs also contain two layers of perceptron-like neurons representing a classifier of lower layer outputs. Perceptron-like neurons compute the dot product of their inputs and their weights, add a bias and apply the transfer function to the evaluated potential.

DOI: 10.7763/IJCTE.2013.V5.722

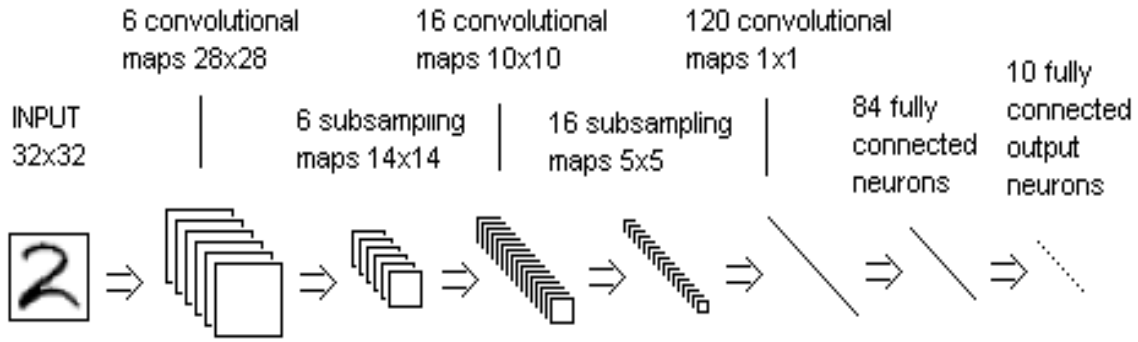

Fig. 1. The structure of the LeNet-5 convolutional neural network

A. The LeNet-5 Model

LeNet-5 is a CNN-network consisting of 7 layers (we do not count the input) that can be trained e.g. by means of back-propagation [6]. Its structure together with the number and size of feature maps used in the respective layers, is shown in Fig. 1. The first layer of the network consists in this example of six convolutional feature maps. Their perception windows are 5 pixels high and 5 pixels wide. Since neighboring perception windows overlap and the size of the input is set to 32 by 32, each feature map in the first layer consists of a grid of 28 by 28 neurons. All the neurons in the respective feature map share thus the same set of 25 weights and the same bias detecting the same feature at all possible locations in the input.

In the following layers, the size of the non-overlapping perception windows is 2×2. In the sub-sampling layers, the four input values of each neuron are summed together, multiplied by an adjustable coefficient, added to an adjustable bias and passed through a sigmoidal activation function. In the fifth layer, the perception windows are of the size 5×5. The sixth layer contains 84 fully connected perceptron-like neurons. The output layer consists of RBF-like neurons [7].

B. Self-Organization and Adaptive Topology

Hybrid Convolutional Neural Networks (HCNNs) [8] use weight sharing and alternating feature extracting and sub-sampling layers, too. For feature detection, they apply, however, RBF-neurons computing the output *y* as:

$$y = \exp\left(-\frac{\| \mathbf{x}-\mathbf{w} \|}{2\sigma^2}\right) \quad (1)$$

where *x* is the input vector, the weight vector **w** corresponds to the center of the Gaussian and σ controls its shape. RBF-neurons fire stronger if the presented input is closer to their weight vectors. Otherwise, HCNN-networks have the same architecture and process the input data in a way very similar to original CNNs. HCNN-networks can be trained by a slightly modified back-propagation algorithm but alternatively, the SOM-learning rules [9] can be used. In particular, the winner-takes-all (WTA) principle impacts fast training and robust networks [8]. On the other hand, the number of feature maps is fixed and has to be set in advance.

In general, it is, however, extremely difficult to estimate an adequate number of significant features in advance. Therefore, we will prefer to adjust it dynamically during training. From the literature, there are known two basic self-organizing architectures with an adaptive topology and Kohonen-like learning [9] – namely the Growing Neural Gas Networks (GNG) and Evolving Trees (ET). GNG-networks [10] are initialized with two neurons. During training, for each presented input pattern **x**, the closest neuron is found and the squared distance between its weight vector **w** and **x** is added to the error counter of this neuron. At regular intervals, a new neuron is added to the network at the location with the maximum accumulated error counter.

In order to find quickly the winning neurons, the ET-model [11] organizes the neurons into a hierarchical tree structure. ETs are initialized with a single neuron. During training, for each presented input pattern, the winning neuron is determined as a leaf of the current ET. The distance from the winning neuron to its neighbors is determined as the number of connections on the path between the considered neurons (in the tree structure) minus one. Whenever the error counter of the winning neuron reaches a splitting threshold, the neuron is labeled as an inner neuron and one or more new neurons are added to the network as its descendants. Inner neurons of ETs are static and therefore, ETs may occasionally determine a wrong winning neuron for the presented pattern.

III. The New Hybrid SOM-Model (HEG-Networks)

The above-discussed network structures provide several characteristics important for efficient yet robust image processing. Convolutional architectures support hierarchical coupling of locally detected pattern features all over the presented image. The actual number of (low-level) features to be searched for can be found automatically with the principles of GNGs. At the same time, fast processing within the layers can be achieved by applying the tree-like search principles of ETs to search for the winning neurons.

Moreover, GNG-like inter-connections between the leaf neurons prevent most of the inaccuracies caused by the tree-like search due to the simultaneous adjustment of both the winning neuron and its neighbors. A new hybrid

self-organizing neural network (HEG) combining the advantages of ETs and GNGs could thus enhance the power of feature detecting layers of GHNNs discussed later on in more detail. An example for this kind of architecture is shown in Fig. 2. Fig. 3. sketches the main principle of growing in HEG-networks and Algorithm 1 formalizes their training in mode detail.

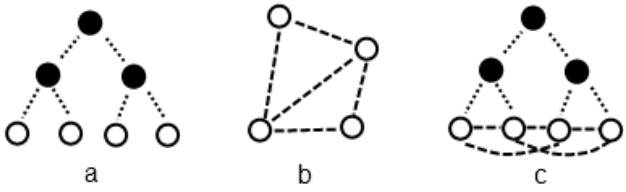

Fig. 2. Network structure formed by the HEG model (c) combines the benefits of an efficient tree search by ETs (a) and locally inter-connected GNGs (b).

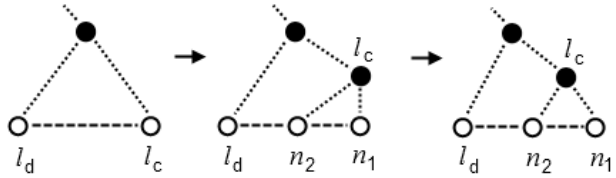

Fig. 3. The process of neuron splitting in HEG-networks (from the left to the right): after the neuron l_c has been selected for splitting, two new neurons n_1 and n_2 are added to the network as children of l_c. Finally, the entire tree is balanced by positioning the weight vector of l_c between the weight vectors of n_1 and n_2.

A good criterion to add a neuron (feature map) could be obviously based on the accumulated squared distance between the respective neurons and all the image samples they represent. New neurons should be added whenever any of the neuron error counters reaches a pre-defined threshold (e.g. 2 *dim*, where *dim* denotes the dimension of the input space). Unfortunately, problems might occur then for small densely populated areas with too many patterns leading to high values of the error counters, too. In such a case, a stopping criterion based on Gaussian-like error counters yields better results:

$$\text{error}(\mathbf{w},\mathbf{x}) = \max\left(0; \frac{1}{\sqrt{e}} - exp\left(-\frac{\|\mathbf{w}-\mathbf{x}\|^2}{2\sigma^2}\right)\right) \quad (3)$$

where $\boldsymbol{w}$ is the weight vector of the neuron closest to the presented sample $\mathbf{x}$ and σ is the variance of the Gaussian. Lower values of the variance parameter naturally imply a higher number of the detected pattern clusters.

IV. Growing Hierarchical Neural Networks

In comparison to original CNNs, GHNNs are enhanced by unsupervised training and the capability to adjust dynamically the number of feature maps in feature-extracting layers. The type of features detected in feature-extracting layers depends on the size and contents of the perception windows. In GHNNs, the perception windows are fully overlapping. The resulting network thus has a pyramidal architecture with a successively reduced size of feature maps in higher layers – see Fig. 4. Usually, the number of feature maps increases in higher layers. The output of a neuron in the respective feature map represents its reaction to all the feature maps from the preceding (sub-sampling) layer.

Algorithm 1: The training algorithm for the HEG-model / used to adjust feature extracting layers in GHNN-networks /

1) Initialize the network with a root neuron and two child neurons connected by an edge.
2) For each presented input pattern x:
   a) Find the first and second closest neurons i_1 and i_2, respectively, among the leaves of the current tree structure. Add the squared distance between the weight vector $\mathbf{w_{i1}}$ and $\mathbf{x}$ to the error counter of the neuron i_1 (alternatively, (3) can be used to adjust the error counter).
   b) Set the age parameter of the edge between the neurons i_1 and i_2 to zero (create this edge if it does not yet exist).
   c) Adjust the weights of the winning neuron i_1 according to (2), with the learning rates α set e.g. to 0.05. No other neurons are adapted.

$$\mathbf{w}_{i_1}^{new} = \mathbf{w}_{i_1}^{old} + \alpha(\mathbf{x} - \mathbf{w}_{i_1}^{new}) \quad (2)$$

3) Decrease the error counters of all neurons (by a pre-set factor, e.g. 0.999) and increment the *age* parameter of all edges of the neuron i_1.
4) At regular intervals:
   a) Add two new leaf neurons to the network, if the conditions for splitting a leaf neuron are met:
   b) The leaf neuron l_c with the largest error counter value is found. If this value is greater than a chosen threshold (e.g. *2 dim*, where *dim* denotes the dimension of the input space), l_c is labeled as an inner neuron and stripped of all edges to other leaves.
   c) Two new leaf neurons n_1 and n_2 are added to the tree as child neurons of l_c - neuron *n* put at the position of l_c will be connected to all neighbors of l_c, and n_2 placed halfway between n1 and its neighbor ld with the next maximum error counter value, splitting thus the original edge connecting l_d and n_1 into two new edges (see Fig. 3).
      i. All neuron error counters are set to zero.
   d) Remove edges with the age parameter exceeding a pre-set threshold value *maxAge* from the network along with any isolated neurons.
   e) Balance the entire tree, such that the weights of every inner neuron correspond to average weights of its child neurons.
5) Stop training if the pre-set number (e.g. 10) of consequent checks for neuron splitting did not result into the addition of new neurons.

Smaller values used for σ during training enforce a tighter representation of the extracted knowledge while its larger values used during recall allow capturing minor differences of the presented test samples, too. Together with the dynamic selection of relevant image features, a smoother network

function supports higher robustness and improved generalization.

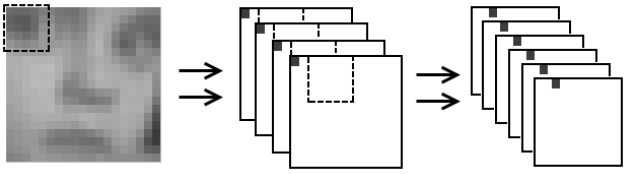

Fig. 4. In every feature-extracting layer of a multi-layered feature detector, neuron output values are determined according to a perception window of a pre-set size sliding over the lower (subsampling) layer. Perception windows both in the original image and in the following layers are depicted by dashed squares, the corresponding values of the feature maps are labeled by black squares.

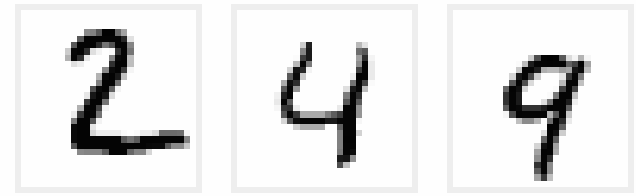

Fig. 5. Size-normalized samples from the MNIST database.

V. Supporting Experiments

The performance of GHNN-networks has been tested on two image classification tasks. For face recognition, we considered a subset of the CBCL Face Database provided by the MIT Center for Biological and Computational Learning [12]. The 19×19 images were histogram equalized and normalized. The training data consisted of 2000 face images and 2000 non-face images. The test set comprised 472 face images and 5000 non-face images. In handwritten digit classification, we used a subset of the MNIST database [13] containing 1000 training and 500 test samples from ten roughly balanced classes – for a few examples see Fig. 5. The samples are 28 by 28 pixels in size with the digits aligned in the center of the sample. The performed experiments involved the following 3 architectures:

1) GHNNs with a dynamically HEG-adjusted number of feature maps. Their actual number can be found in the third column of Table II.
2) HCNNs – both with the RBF-like neurons and with the WTA-functionality – having the same number of feature maps like the corresponding GHNNs.
3) HCNNs with the same number of feature maps like the commercially used LeNet-5 model [5]. Results obtained for 5-layer networks are stated in the brackets – see Table I and Table II.

All the networks had feature maps of the same size that is given by both the dimension of the input data and the size of the applied perception windows (all the perception windows were of size 2×2):

- CBCL data: 18×18, 9×9, 8×8, 4×4 and 3×3,
- MNIST data: 27×27, 13×13, 12×12, 6×6 and 5×5.

The variance of the Gaussians used in feature-extracting layers during training has been set experimentally to 0.4, 0.5 and 0.6, respectively. During recall, it has been chosen as 1.6, 2.0 and 2.4 for the CBCL images and 2.8, 3.5 and 4.2 for the MNIST data. The attached classifier always contained 30 neurons in one hidden layer. The number of its output neurons reflected the number of data classes. All the tests were performed 10-times. The results shown in Table I and Table II correspond to the measured values averaged over the 10 runs of the respective network model. In the experiments, we were in particular interested in answering the following questions:

1) How fast is training of GHNN-networks when compared to HCNN-like networks?
2) What is the accuracy of GHNN-networks?

What is the character of the internal knowledge representation extracted by the networks and how many features does the network actually use?

A. The First Set of Experiments

The dynamically adjusted number of feature maps and their mutual coupling are expected to speed-up training. On the other hand, the HEG-like adjustment procedure might slightly slow it down in comparison with almost optimal network architectures. Experimental results from Table I confirm these expectations. GHNNs perform comparably for both types of HCNN-like networks initialized with the same number of feature maps as found by the respective GHNN. But in comparison with HCNN-like networks of the same architecture like the original LeNet-5, GHNNs were roughly two times faster.

B. The Second Set of Experiments

Due to the dynamically found number and coupling of feature maps, we would expect a higher testing accuracy for GHNNs when compared to other HCNN-like networks. As the CBCL test data classes are extremely unbalanced, we decided to consider also the information about precision and recall. Precision corresponds to the number of correctly classified positive examples divided by the total number of samples classified as positive. Recall stands for the number of correctly classified positive samples divided by the total number of actually positive samples.

Fig. 6. Size-normalized samples from the CBCL Face database (from top to bottom): faces classified as faces, faces classified as non-faces, non-faces classified as non-faces and non-faces classified as faces.

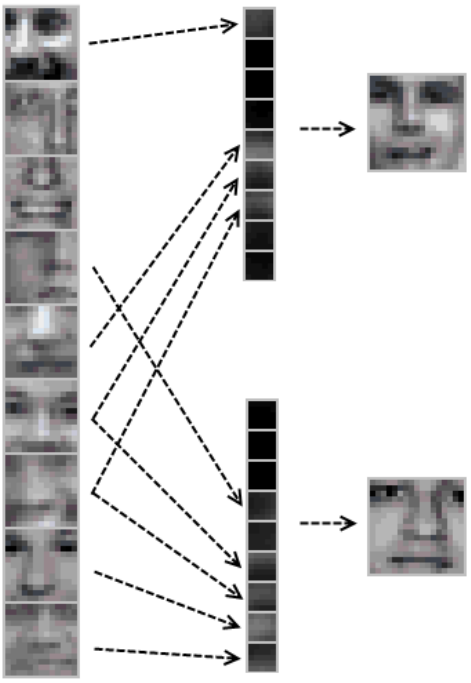

Fig. 7. Advanced features detected at higher levels of GHNN-networks are composed of simple local features detected already at its lower layer: left, this graphics shows the representative images for nine neurons (maps) of the third feature extracting layer (middle). On the right, two face images are shown that belong to two different face clusters. The upper face can be characterized e.g. by a dark eye area and a significant cheek (feature 1). The lower one depicts on the other hand a wider nose and mouth region (feature 8 and 9).

The results reported in Table I support our hypothesis for the CBCL data. Moreover, several of the misclassified images are really difficult to classify automatically, e.g. blurred faces or faces with glasses, non-face images with face-like components (mostly reminding us of eyes and noses) – for examples, see Fig. 6. Anyway, for the MNIST data and an optimized architecture, HCNN-networks with the WTA-functionality outperform our model, probably due to the aggressively reduced number of features considered by sub-sampling layers.

C. The Third Set of Experiments

We compared the number of (detected/pre-set) features with the number of features active in average for each presented image. The features were considered active if the output of the corresponding neuron exceeded the value of 0.7. Moreover, by calibrating the feature (maps) with image parts closest to them, we can easily interpret the function of the network – see Fig. 7. From Table II, we can conclude that GHNNs are capable of finding an adequate structure for the given task. Especially when considering higher-level features, GHNNs provide a more compact representation of the processed data than HCNNs with the RBF-like neurons. On the other hand, GHNNs do not eliminate too much information by the sub-sampling layers like HCNNs with the WTA-functionality which seems to be advantageous for very complex problems like face recognition.

TABLE I: THE PERFORMANCE OF THE TESTED MODELS ON THE CBCL FACE DATA AND MNIST HANDWRITTEN DIGIT DATA. IN THE BRACKETS ARE STATED THE RESULTS OBTAINED FOR 5-LAYER MODELS USING THE SAME NUMBER OF FEATURE MAPS LIKE THE COMMERCIALLY USED LENET-5 MODEL [6].

		GHNN	HCNN-WTA	HCNN-RBF
CBCL (4 layers)	detector training (sec)	0.7	**0.4**	0.7
	classif. training (sec)	9.1	**9.0**	9.1
	training accuracy (%)	**96.8**	96.5	93.2
	testing accuracy (%)	**92.7**	91.9	91.5
	precision (%)	**58.4**	52.4	50.8
	recall (%)	53.8	**61.7**	57.8
CBCL (5 layers)	detector training (sec)	2.1	**1.5** *(6.0)*	2.1 *(6.9)*
	classif. training (sec)	**20.4**	20.5 *(45.0)*	20.6 *(45.3)*
	training accuracy (%)	**97.7**	94.6 *(97.3)*	86.8 *(82.6)*
	testing accuracy (%)	**92.1**	82.4 *(82.7)*	73.9 *(65.4)*
	precision (%)	**55.3**	28.7 *(26.8)*	22.0 *(17.6)*
	recall (%)	45.3	62.5 *(57.6)*	**79.2** *(82.0)*
MNIST (4 layers)	detector training (sec)	5.5	**3.0**	4.9
	classif. training (sec)	29.6	**29.5**	29.6
	training accuracy (%)	93.6	**96.8**	82.7
	testing accuracy (%)	90.4	**92.8**	81.2
MNIST (5 layers)	detector training (sec)	**14.4**	17.6 *(32.2)*	**14.4** *(16.5)*
	classif. training (sec)	64.2	64.2 *(122.7)*	**64.1** *(123.7)*
	training accuracy (%)	96.8	**96.8** *(97.2)*	70.4 *(44.2)*
	testing accuracy (%)	**91.8**	86.3 *(85.5)*	68.7 *(43.4)*

TABLE II: THE NUMBER OF FEATURES DETECTED AND USED IN AVERAGE FOR THE PROCESSED PATTERNS FROM THE CBCL AND MNIST DATA. IN THE BRACKETS ARE STATED THE RESULTS OBTAINED FOR 5-LAYER MODELS USING THE SAME NUMBER OF FEATURE MAPS LIKE THE COMMERCIALLY USED LENET-5 MODEL [6].

			GHNN	HCNN-WTA	HCNN-RBF
		detected features	used features	used features	used features
CBCL (4 layers)	layer 1	3.0	3.0	3.0	3.0
	layer 2	3.0	3.0	3.0	3.0
	layer 3	9.1	9.1	8.9	9.1
	layer 4	9.1	7.4	2.3	9.0

CBCL (5 layers)	layer 1	2.8 *(6)*	2.8	2.8 *(6.0)*	2.8 *(6.0)*
	layer 2	2.8 *(6)*	2.8	2.8 *(5.6)*	2.8 *(6.0)*
	layer 3	8.7 *(16)*	8.7	8.4 *(14.7)*	8.7 *(16.0)*
	layer 4	8.7 *(16)*	7.3	2.3 *(1.44)*	8.7 *(16.0)*
	layer 5	48.0 *(120)*	6.9	7.6 *(8.4)*	40.2 *(101.0)*
MNIST (4 layers)	layer 1	6.7	6.9	6.9	6.9
	layer 2	6.7	6.9	3.3	6.9
	layer 3	17.5	16.9	15.8	17.5
	layer 4	17.5	7.0	1.3	16.7
MNIST (5 layers)	layer 1	7.1 *(6)*	7.1	7.1 *(6.0)*	7.1 *(6.0)*
	layer 2	7.1 *(6)*	7.1	3.1 *(3.6)*	7.1 *(6.0)*
	layer 3	17.6 *(16)*	16.8	16.0 *(14.9)*	17.6 *(16.0)*
	layer 4	17.6 *(16)*	6.8	1.3 *(1.3)*	16.8 *(15.3)*
	layer 5	58.6 *(120)*	15.6	18.7 *(20.1)*	57.01 *(116.7)*

VI. Summary

The introduced GHNN-networks represent a powerful multi-layered self-organizing architecture capable of constructing automatically an appropriate feature detector from the processed data. In accordance with other approaches (e.g. [14], [15]), an adequate network structure avoiding redundant processing has shown to yield improved generalization. The main benefits of the proposed GHNN-model with the HEG-networks used for feature extraction consist in:

1) an automatic adjustment of the network topology to the dimensionality and inner structure of the processed data. In comparison with fixed-size networks like e.g. Kohonen maps, this results in rapid training and improved generalization (due to a reduced number of used neurons),
2) an efficient processing of the image data, in particular, when dealing with large high-dimensional datasets (due to the tree-like structure of the HEG-networks and substantially reduced pre-processing),
3) a transparent hierarchical representation of the extracted knowledge. More complex features extracted in higher layers of the detector are composed of simpler features found in its lower layers.

References

[1] Social Media. (July 2010). [Online]. Available: http://www.edition.cnn.com/2010/TECH/ social. media/07/02/facebook.recognition.index.html

[2] S. Fidler, G. Berginc, and A. Leonardis, "Hierarchical statistical learning of generic parts of object structure," in *Proc. 2006 CVPR Conf.*, Washington, 2006, pp. 182-189.

[3] K. Horio, A. Aikawa, and T. Yamakawa, "Pattern recognition based on relative position of local features using self-organizing map," in *Proc. 2006 ICICIC Conf.*, 2006, pp. 293-296.

[4] D. Zhong and I. Defee, "Face retrieval based on robust local features and statistical-structural learning approach," *EURASIP J. on Advances in Signal Processing*, vol. 2008, pp. 12, April 2008.

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

[6] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533-536, October 1986.

[7] J. Moody and C. Darken, "Learning with localized receptive fields," in *Proc. of the 1988 Connectionist Models Summer School*, 1988, pp. 133-143.

[8] I. Mrazova and M. Kukacka, "Hybrid convolutional neural networks," in *Proc. 2008 INDIN Conf.*, 2008, pp. 469-474.

[9] T. Kohonen, *Self-Organizing Maps*, Berlin: Springer-Verlag, 2001.

[10] B. Fritzke, "Growing neural gas learns topologies," *Advances in Neural Information Processing Systems*, vol. 7, pp. 625-632, August 1995.

[11] E. Oja, J. Pakkanen, and J. Iivarinen, "The evolving tree – a novel self-organizing network for data analysis," *Neural Processing Letters*, vol. 20, no. 3, pp. 199-211, November 2004.

[12] Center for Biological and Computational learning at MIT. (November 2012). Face data. [Online]. Available: http://www.cbcl.mit.edu/cbcl/software-datasets/FaceData2.html

[13] Y. LeCun and C. Cortes. (November 2012). The MNIST database of handwritten digits. [Online]. Available: http://www.yann.lecun.com/exdb/mnist/

[14] I. Mrazova and D. Wang, "Improved generalization of neural classifiers with enforced internal representation," *Neurocomputing*, vol. 16-18, no. 70, pp. 2940-2952, October 2007.

[15] V. N. Vapnik, *Statistical Learning Theory*, New York: Wiley, 1998.