New Architecture of Correlated Weights Neural Network for Global Image Transformations

Sławomir Golak, Anna Jama(✉), Marcin Blachnik, and Tadeusz Wieczorek

Department of Industrial Informatics, Silesian University of Technology, Krasinskiego 8, 40-019 Katowice, Poland
anna.jama@polsl.pl

Abstract. The paper describes a new extension of the convolutional neural network concept. The developed network, similarly to the CNN, instead of using independent weights for each neuron in the network uses related weights. This results in a small number of parameters optimized in the learning process, and high resistance to overtraining. However unlike the CNN, instead of sharing weights, the network takes advantage of weights correlated with coordinates of a neuron and its inputs, calculated by a dedicated subnet. This solution allows the neural layer of the network to perform global transformation of patterns what was unachievable for convolutional layers. The new network concept has been confirmed by verification of its ability to perform typical image affine transformations such as translation, scaling and rotation.

Keywords: Network architecture · Spatial transformation · CNN

1 Introduction

Recent approaches to object recognition make essential use of machine learning methods [1, 2]. To increase their performance, we can collect larger datasets, learn more powerful models, and use better techniques for preventing overfitting [3]. To create network capable to learn to recognize thousands of objects from millions of images, we need to build a model with a large learning capacity. Convolutional neural networks (CNNs) constitute one such class of models. They are powerful visual models which recently enjoyed a great success in large-scale image and video recognition [4–6], what has become possible thanks to the large public image repositories, such as ImageNet, and high performance computing systems, such as GPUs or large-scale distributed clusters [7]. CNN combine three architectural ideas to ensure some degree of shift, scale, and distortion invariance: local receptive fields, shared weights (or weight replication), and spatial or temporal subsampling [8, 9]. The main benefit of using CNNs is the reduced amount of parameters that have to be determined during a learning process. CNN can be regarded as a variant of the standard neural network which instead of using fully connected hidden layers, introduces a special network structure, which consists of alternating so-called convolution and pooling layers [10].

One of the limitations of CNN is the problem of global pattern transformations such as e.g. rotation or scaling of an image. Convolutional neurons which have spatially limited input field are unable to identify such transformations. They are only capable to local pattern transformations such as detection of local features on the image. The size increase of spatial transformation area would require the enlargement of the neurons input field, therefore the increase of weight vector size and result in the loss of the primary advantage of the CNN network, which is a small number of parameters.

In order to overcome the constraints of CNN in the paper we introduce a new network which can replace the convolutional layer or it can be used as an independent network able to learn any global and/or local transformations. The proposed network consists of two networks. The main network is a single fully connected layer of neurons, aimed at direct performing image transformation. However, weights of this network are not determined directly, instead to limit the number of trainable parameters, the weights of the neurons are obtained by the sub-network, which is relatively small network. The second network takes advantage of the observation that for global transformations particular weights of individual neurons are strongly correlated with one another, hence the entire network is called the Correlated Weights Neural Network (CWNN). Proposed network is a continuation of the earlier idea of using weights calculated by the subnet in a Radial Basis Network (RBF) for pattern classification [11] called the Induced Weights Artificial Neural Network (IWANN). However, in the IWANN the values of weights were determined only based on the input coordinates, consequently the coordinates of the neurons were ignored. This approach was related to different structure and application of the IWANN network.

In the paper we describe and explain the structure of the CWNN network (Sects. 2 and 3), then Sect. 4 describes the training algorithm, and in Sect. 5 we demonstrate its application to train global transformations such as scaling, translation and rotation. We end with a summary of the obtained results and draw further research perspectives.

2 Problem Definition

Neural network presented in this paper is dedicated to global pattern transformations. Figure 1 shows linearized representation of the main network's neurons which are responsible for the input-output transformation, and assume gray scale images. This single layered network consists of linear neurons, witch without the bias, are sufficient to implement the network. Each neuron determine the output value (the gray level of the pixel for output image) as a linear combination of all pixels in the input image. In case of a N size pattern, this operation can be perform by a network which has N output neurons associated to each of the N inputs. Total number of weights for this network is N2, which for relatively small (32×32 pixels) images presented in the further part of the article leads to the number of 1 million parameters describing the network. The number of parameters is directly reflected in the complexity of the network learning process, and in the size requirements for training dataset.

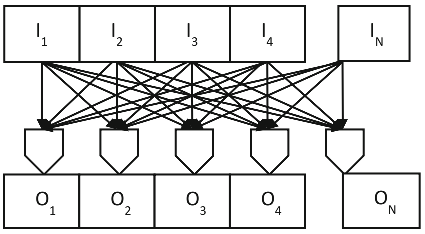

Fig. 1. Neural network for a global transformation of a pattern.

However, in vast majority of global transformation, network weights are correlated with positions of neurons and their inputs. Table 1 presents the values of weights for the network from Fig. 1, which performs translation of one-dimensional pattern, by 2 elements to the right (for simplicity we consider N = 5). The columns in the table correspond to the inputs, and rows to the outputs, so a single row represents weights of a single linear neuron. The final value of each output is determined as the weighted sum of inputs described by weights in particular row.

Table 1. Weights of the neural network translating a linear pattern by two elements.

	I1	I2	I3	I4	5
O1	0	0	0	0	0
O2	0	0	0	0	0
O3	1	0	0	0	0
O4	0	1	0	0	0
O5	0	0	1	0	0

The table content shows a clear regularity in the set of weights. The value of the weight which connects the i-th output's neuron with j-th input, in this network, can be described by simple equation:

$$w_{i,j} = \begin{cases} 1 & j - t = i \\ 0 & j - t \neq i \end{cases} \tag{1}$$

Where: t - size of translation, i-output position, j-input position

This formula replaces 25 network weights, and it's degree of complexity is independent of the size of transformed pattern. The essential idea of the CWNN network can be explained by using this example. It involve the fact that the weights of the network are not stored as static values, but calculated based on the mutual position of neurons and their inputs. The practical use of the relation between neuron weights, requires a subsystem which is able to learn the dependences between neurons and its weights. To address this issue, we utilize additional subnetwork presented in the following section.

3 Network Model

Figure 2 shows the structure of a single-layer network with the Correlated Weights Neural Layer (CWNL). The network has one input layer, which topology results from the size and dimensionality of processed input pattern. Network inputs are described by the coordinates vector, which size is compatible with dimensionality of the input data. The topology of the main, active CWNL layer, which have signals transmitted directly on the network output, is compatible with dimensionality and the size of the output pattern. It should be taken into consideration that both dimensionality and the size of the input and the output can be completely different.

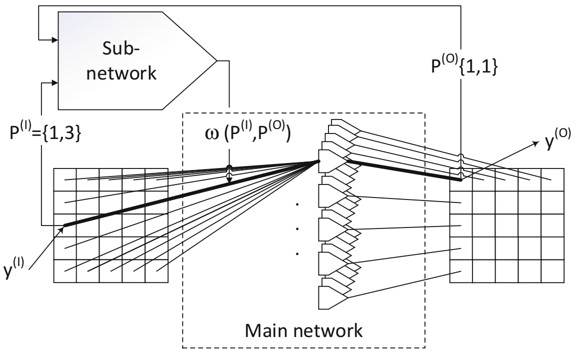

Fig. 2. Structure of the neural network with correlated weights.

It has been assumed that for the neural network performing affine transformations on an image, it is sufficient to use neurons with a linear transition function, that takes as an argument a weighted sum of inputs, without the bias value (2).

$$y_i^{(O)} = \sum_{j=1}^{N^{(I)}} y_j^{(I)} \omega_{i,j,1}^{(M)} \tag{2}$$

Where: $y_i^{(O)}$ – i-*th* output of the layer with correlated weights, $y_j^{(I)}$ – j-*th* input of the layer with correlated weights, $N^{(I)}$ – number of inputs for the CWNL layer, $\omega_{i,j,1}^{(M)}$ – output of subnet calculating weights – weight of connection between the i-*th* output with the j-*th* input of the CWNL layer, M - the number of the last subnet layer.

The values of connected weights are calculated by the subnetwork based on the coordinates of neuron in the CWNL layer and the coordinates of neurons inputs. These values are determined many times by the subnet for every combination of input image pixels and output image pixels. The subnetwork inputs are represented by a vector, which consists of coordinates of the CWNL neurons and their inputs:

$$P_{i,j} = P_i^{(I)} \cup P_j^{(O)} \tag{3}$$

where: $P_i^{(I)}$ – coordinates of the i-*th* input of the main network, $P_j^{(O)}$ – coordinates of the j-*th* output (neuron) of the main network

$$\omega_{i,j,k}^{(O)} = P_{i,j}[k] \tag{4}$$

where: the k-*th* input of the subnetwork calculating the weight of the connection between of the i-*th* input and the j-*th* neuron of the CWNL.

The signal is processed by subsequent layers of the subnetwork:

$$\omega_{i,j,k}^{(m)} = f^{(m)}\left(\sum_{l=1}^{O^{(m-1)}} \omega_{i,j,l}^{(m-1)} w_{k,l}^{(m)} + b_k^{(m)}\right) \tag{5}$$

where: $\omega_{i,j,k}^{(m)}$ – output of the k-*th* neuron of m-th layer of the subnetwork calculating the weight for the connection between j-*th* neuron and i-th input of the CWNL layer, $f^{(m)}$ – the transition function of neurons in the m-*th* layer, $O^{(m)}$ – the number of neurons in the m-*th* subnetwork layer, $w_{k,l}^{(m)}$, $b_k^{(m)}$- standard weight and bias of the subnet neuron.

In the presented network structure, it was proposed to use the multilayer perceptron as a subnetwork. However, this task can be performed by any other approximator.

4 Learning Method

The learning algorithm of the CWNN network is based on the classical minimization of the square error function, defined as:

$$SSE = \frac{1}{2}\sum_{j=1}^{N^{(O)}}\left(y_j^{(O)} - d_j\right)^2 \tag{6}$$

Where: $N^{(O)}$– number of neurons in the layer with correlated weights, $y_j^{(O)}$– j-*th* output of the layer with correlated weights, d_j – desired value of the j-*th* output of the CWNN network.

Effective learning of CWNN requires the use of one of the gradient methods to minimalize the error. In case of presented network, the parameters optimized in the learning process include only: weights and biases of the subnetwork, that calculates the main layer weights. This required modification of the classical backpropagation method, to allow for an error transfer to the subnet. The error value for the output neuron of the subnetwork, which calculates the weight of the connection between the j-th output and the i-th input of the main network, is determined by the equation:

$$\sigma_{i,j,k}^{(M)} = \left(y_i^{(O)} - d_i\right) y_j^{(I)} \tag{7}$$

where: $y_j^{(I)}-$ i-*th* output of the layer with correlated weights, $y_j^{(O)}-$ j-*th* output of the layer with correlated weights, d_i – value of the i-*th* input of main network.

In the next stage the error is back propagated from the output through subsequent network layers. The aim of the backpropagation is to update each of the weights in the subnetwork so that they cause the actual output to be closer the target output, thereby minimizing the error for each output neuron and the network as a whole.

$$\sigma_{i,j,k}^{(m)} = \sum_{i=1}^{O^{(m+1)}} \left(\left(\sigma_{i,j,l}^{(m+1)} w_{k,l}^{(m+1)} \right) f' \left(\sum\nolimits_{l=1}^{O^{(m-1)}} \omega_{i,j,l}^{(m-1)} w_{k,l}^{(m)} + b_k^{(m)} \right) \right) \tag{8}$$

Where:$\omega_{i,j,k}^{(m)}$ – output of the k-*th* neuron of m-th layer of the subnetwork calculating the weight for the connection between j-*th* neuron and i-th input of the CWNL layer, $O(m)-$ the number of neurons in the m-*th* subnetwork layer, $w_{k,l}^{(m)}$, $b_k^{(m)}$- standard weight and bias of the subnet neuron.

Based on the error value it is possible to determine partial derivative for all parameters (weights and biases) of the subnet. Determination of partial derivative requires summation of derivatives calculated for all weights provided by the subnet:

$$\frac{\partial E}{\partial w_{k,l}^{(m)}} = \sum_{i=1}^{N^{(I)}} \sum_{j=1}^{N^{(O)}} \sum_{l=1}^{O^{(m-1)}} \sigma_{i,j,k}^{(m)} \omega_{k,l}^{(m-1)}; \frac{\partial E}{\partial b_k^{(m)}} = \sum_{i=1}^{N^{(I)}} \sum_{j=1}^{N^{(O)}} \sum_{l=1}^{O^{(m-1)}} \sigma_{i,j,k}^{(m)} \tag{9}$$

where: $\omega_{i,j,k}^{(m)}$ – output of the k-*th* neuron of m-*th* layer of the subnetwork calculating the weight for the connection between j-*th* neuron and i-*th* input of the CWNL layer, $O^{(m)}$- the number of neurons in the m-*th* subnetwork layer, $w_{k,l}^{(m)}$, $N^{(O)}-$ number of neurons in the layer with correlated weights,$N^{(I)}-$ number of inputs for the CWNL layer

For the presented network partial derivatives were computed with the use of the above equations.

5 Results

Effectiveness of the developed neural network has been exanimated by the implementation and analysis of the global image transformations, like scaling, rotation and translation. The popular CIFAR-10 collection was used in the experiments [12]. This dataset was primarily intended for testing classification models, but it can be considered as a useful source of images also for other applications. The collection contains various type of scenes, and consists of 60000 32 × 32 color images.

Due to the large computational complexity of learning process calculations were performed on the PLGRID cluster, which was created as part of the PL-Grid - Polish Science Infrastructure for Scientific Research in the European Research Area project. The PLGRID infrastructure contains computing power of over 588 teraflops and disk storage above 5.8 petabytes.

For each variant of the experiment a collection of images, for both training and testing set, contained only 50 examples. Global image transformations were used to verify the performance of the designed system. The input of the network was a primary image (1024 pixels). The expected response (the output of the network) was an image after the selected transformation. The network was trained in batch mode using the RPROP method (resilient back propagation) [13] due to the resistance of this method to the vanishing gradient problem. This phenomenon occurs in networks with a complex, multilayer structure as in the developed network. For small training sets this method is more effective than the popular group of stochastic gradient descent methods due to a more stable learning process. Weights of the subnetwork were initialized randomly with the use of Nguyen-Widrow method [14]. Due to very low susceptibility to overtraining by the new network, the stop procedure, based on the validation set, was omitted in the learning procedure. The correctness of this decision was confirmed by the obtained results. The quality of the image transformations, for: 50, 100, 1000 and 5000 epochs, was monitored during the learning process, as well as the course of changes in the MSE for both training and testing set. The same network parameters was applied to each variant of the experiment. In the first stage of research the network was trained to perform the image vertical scale transformation. The goal was to resize the image by 50% of height. The main layer with correlated weights, was compatible with the dimensions of the images in the training and test set, and had 1024 inputs and 1024 outputs, so that each neuron correspond to one pixel in 32 $\times$ 32 grid. The subnet, calculating weights of the main layer, consisted of 3 layers containing 8, 4, 1 neurons respectively, with a sigmoidal transition function. So a single sigmoid neuron providing the value in the range of [0..1] is present on the output of the sub-network. This is in line with the nature of the mapped transformations in which there are no negative weight values and values greater than 1. The convergence of the learning algorithm was measured using the mean squared error (per pixel) for each training epoch:

$$MSE = \frac{1}{S \cdot N^{(O)}} \sum_{k=1}^{S} \sum_{j=1}^{N^{(O)}} \left(y_{kj}^{(O)} - d_{kj} \right)^2 \tag{10}$$

where: S – number of examples in the set, $N^{(O)}$- number of network outputs, $y_{kj}^{(O)}$ – j-th output of the network for k-th example, d_{kj} – desired value of the j-*th* output (pixel brightness) for k-*th* example.

Figure 3 shows the decrease of the MSE during the learning process and a sample of images from the training set with the stages of scaling results during the learning process. It was observed that, after 1000 learning epochs, the outline of correct transformation appeared. After 5000 epochs, the quality of transformed image was satisfactory, although there were still disturbances in case of some images (see the fourth image – the plane). The black stains are areas where pixel values have exceeded the limit value 1. This is the result of a large proportion of bright pixels in the plane image in comparison to other images. Based on the MSE graph it can be concluded that the learning process can still be continued, which should result in further improvement of the quality of the transformation.

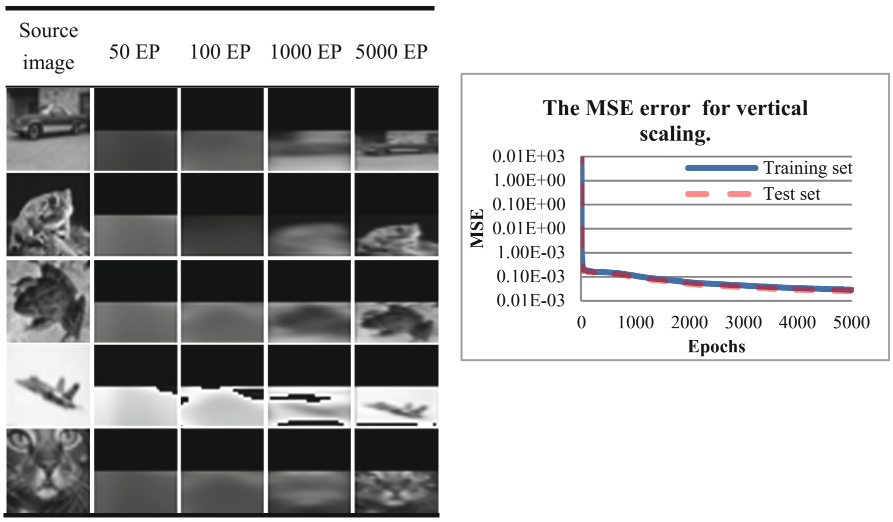

Fig. 3. The MSE error and quality of transformed image for vertical scaling.

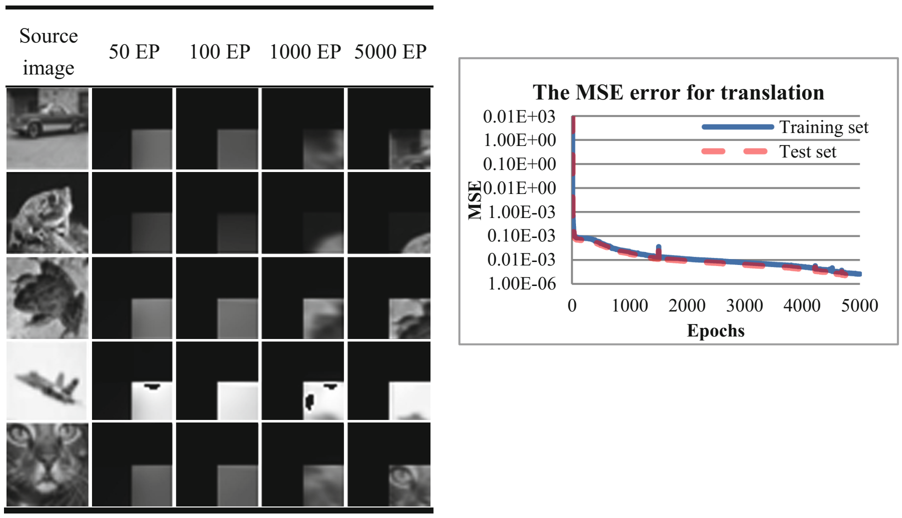

Fig. 4. The MSE error and quality of transformed image for translation.

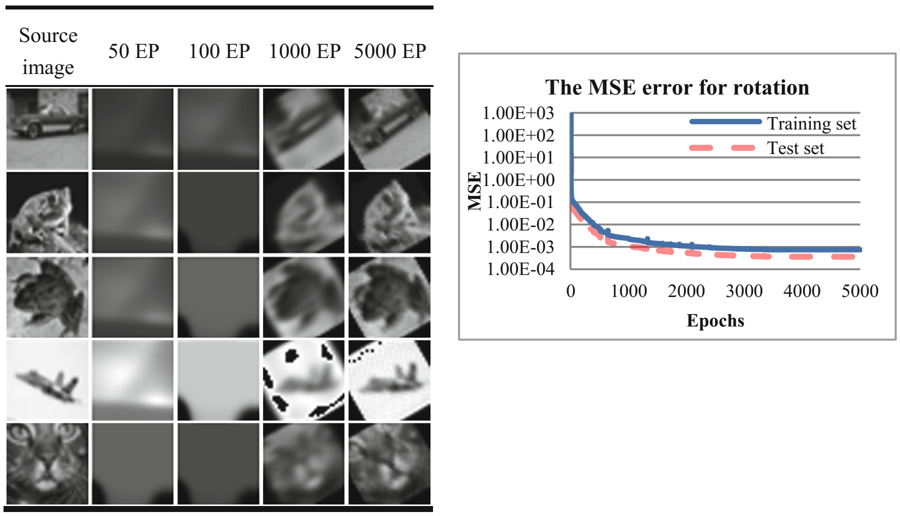

Fig. 5. The MSE error and quality of transformed image for rotation.

The same network was trained to perform an image translation. Quality of obtained results was acceptable after 5000 epochs, so the learning process was terminated. During the learning process the continuous decrease of the MSE was observed (Fig. 4) in both training and testing set. After 1000 learning epochs, the outline of the transformation appeared. The figure also shows stages of the image transformation during the learning process.

Image rotation by 30° was the most difficult challenge for the network due to the necessity of mapping trigonometric relations. During the learning process the decrease of MSE error is similar to the previous cases (Fig. 5). In analyzed range there was no overtraining of the network. After 100 epochs we can observe an outline of the rotated image, but the picture itself is blurry. After 1000 epochs the image become clearer and it is possible to recognize the shape and details of the source image in transformed picture. Like in the previous calculations quality of the obtained results was acceptable after 5000 epochs.

6 Conclusion

Proposed neural network represents a significant extension of the concept of network with convolutional layers. It use the current CNN idea of similarity between the weights for individual neurons in the layer, but breaks with their direct sharing concept. Based on the observation on the correlation between, the values of weights and coordinates of neurons inputs and the coordinates of the neurons themselves, it can be stated that the CWNN network can implement transformations not available for the CNN network. At the same time, the network retains main advantage of CNN, which is the small number of parameters that should be optimized in the learning process. This paper propose a new structure of the neural network and its learning method. The concept of network has been verified by checking its ability to implement typical global pattern transformations. The results confirms the ability of the CWNN to perform any global transformations. Presented research has been conducted based on a single-layer CWNN. Further research will focus on creation of networks with multiple layers, and ability to combine these layers with convolution layers, as well as with standard layers with a full pool of connections. This should give a chance to develop new solutions in the area of deep networks, which will allow to get competitive results in more complex tasks.

Acknowledgments. This research was supported in part by PL-Grid Infrastructure under Grant PLGJAMA2017.

References

1. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: NIPS 2015 (Spotlight), vol. 2, pp. 2017–2025 (2015)
2. Ferreira, A., Giraldib, G.: Convolutional Neural Network approaches to granite tiles classification. Expert Syst. Appl. **84**, 1–11 (2017)
3. Krizhevsky, A., Sutskever I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS-2012, pp. 1097–1105 (2012)
4. Zhang, Y., Zhao, D., Sun, J., Zou, G., Li, W.: Adaptive convolutional neural network and its application in face recognition. Neural Process. Lett. **43**(2), 389–399 (2016)
5. Radwan, M.A., Khalil, M.I., Abbas, H.M.: Neural networks pipeline for offline machine printed Arabic OCR. Neural Process. Lett. (2017). https://doi.org/10.1007/s11063-017-9727-y
6. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556. http://arxiv.org/abs/1409.1556. Accessed 19 May 2018
7. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient based learning applied to document recognition. IEEE **86**(11), 2278–2324 (1998)
8. Abdel-Hamid, O., Mohamed, A., Jiang, H., Deng, L., Penn, G., Yu, D.: Convolutional Neural Networks for speech recognition. IEEE/ACM Trans. Audio Speech Lang. Process. **22**(10), 1533–1545 (2014)
9. Wang, Y., Zu, C., Hu, G., et al.: Automatic tumor segmentation with Deep Convolutional Neural Networks for radiotherapy applications. Neural Process. Lett. (2018). https://doi.org/10.1007/s11063-017-9759-3
10. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
11. Golak, S.: Induced weights artificial neural network. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3697, pp. 295–300. Springer, Heidelberg (2005). https://doi.org/10.1007/11550907_47
12. Cire, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. Arxiv preprint arXiv:1202.2745 (2012)
13. Christian, I., Husken, M.: Empirical evaluation of the improved Rprop learning algorithms. Neurocomputing **50**, 105–123 (2003)
14. Nguyen, D., Widrow, B.: Improving the learning speed of 2-layer neural networks by choosing initial values of adaptive weights. In: IJCNN, pp. III-21–26 (1989)