COUNTER-PROPAGATION (CPN) NETWORK

The data compression yields to the data reduction which is to be send or stored usually with the possibility of its full reproduction (decompression). Image data compression is used to encode large amounts of image data for transmission over a limited-capacity channels. Recently, neural networks algorithms have been developed for data compression, yielding superior performance over classical techniques.

For the efficient compression, the image data are passed through a network producing the binary vectors as their compressed version. After decompression the image data and output data are expected to be very close.

The compression ratio depends greatly on the tolerated amount of error. Algorithm is made of two major steps, namely network training (or learning) and processing (data compression and decompression)

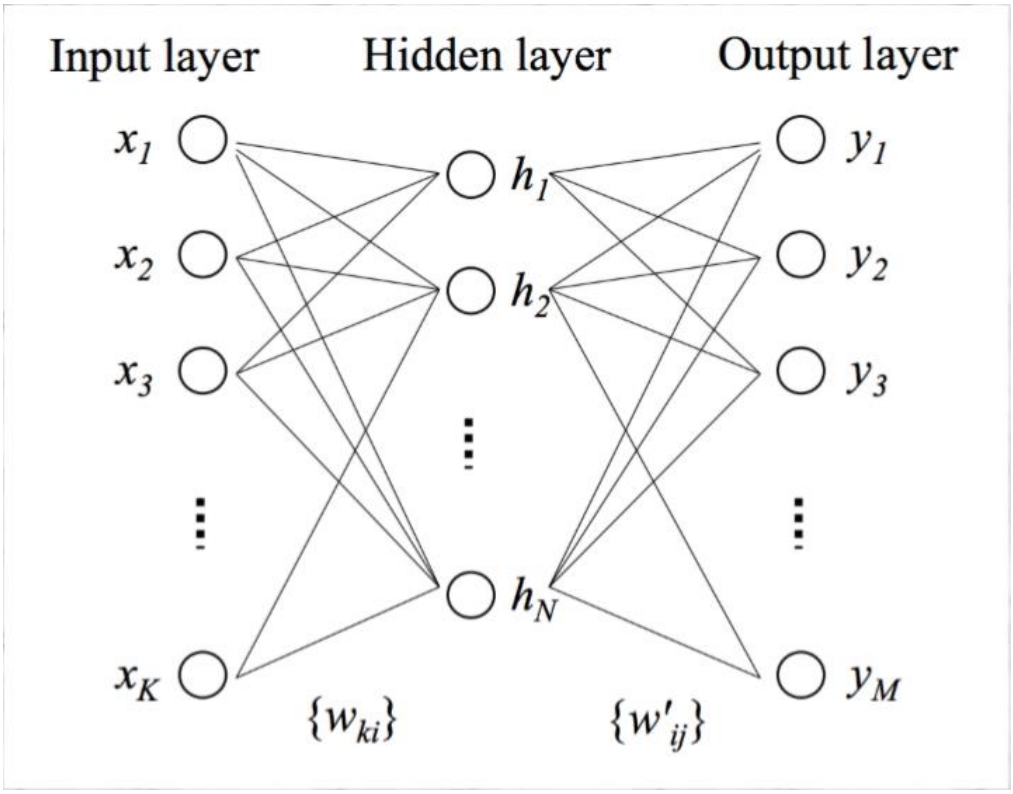


Figure 1: The model of multilayer neural network for a data compression

The input data consist of N vectors composed of n elements (i.e. $n = 64$) are sequentially applied to the input layer. The number of elements of an input vector is equal to the number of elements in this layer (i.e. 8x8) and also equal to number of pixels in the sample of a picture.

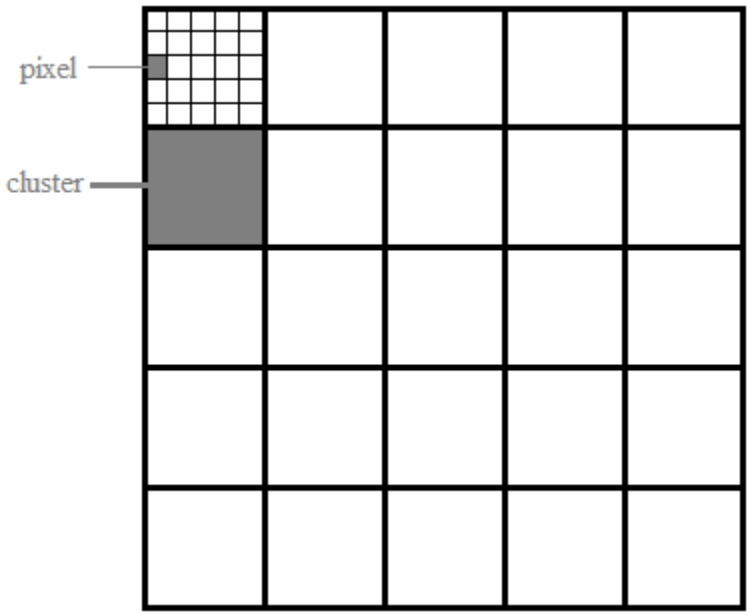
Assuming that each pixel has a gray level between 0 and 255 (0 – means absolutely dark, and 255 means absolutely white) the pixel can be stored with one byte (8 bits) of information.

The hidden layer is composed from q neurons ($q << n$, i. e. $q = 16$) and realizes data compression. In the above-mentioned example, we get reduction factor equal to 4. The input layer and hidden layer can be treated as a transmitter.

In the output layer there is n neurons again, and the decompression has place reproducing the original 64 elements pattern samples.

The output layer can be treated as a receiver. Usually the most popular system of a network learning is the is the back-propagation algorithm.

The input picture is divided into clusters composed of pixels.



It was shown that neural network, called the counter-propagation network, can perform for some applications even better than the back-propagation one. The architecture of the counter-propagation network is a combination of the self-organizing map of Kohonen and the outstar structure of Grossberg. It has two layers similar to feedforward networks BUT it has different learning strategy.

The counter-propagation (CPN) network is composed of two feedforward layers: the Kohonen layer and the Grossberg layer. The number of neurons in these layers can differ. There is a total interconnection between layers; every Kohonen neuron is connected with each Grossberg neuron. The Kohonen layer is connected to the network input, and is operating under the rule winner takes all (WTA). CPN is useful in pattern mapping and associations, data compression, and classification.

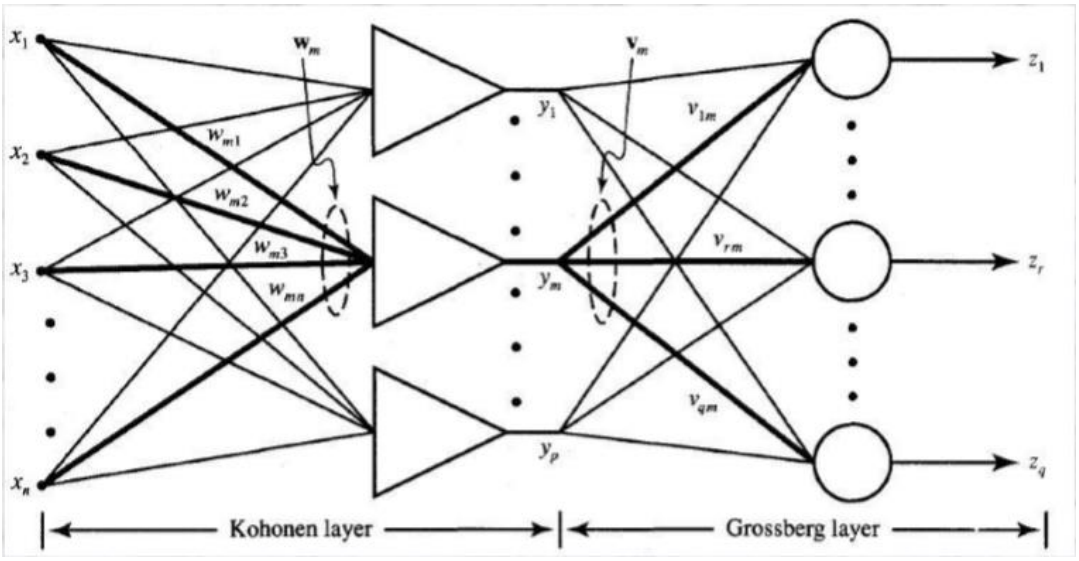


Figure 2: The architecture of counter-propagation neural network

For the normalized vector input signal $\boldsymbol{X} = [x_1, x_2, \ldots, x_k]$ each Kohonen neuron k_i obtains at its input the weighted sum

$$NET_i = \sum_{j=1}^{k} w_{ij} x_j \qquad \text{for } i = 1,2,\ldots,N,$$

where

N is the number of Kohonen neurons,

w_{ij} is the weight between jth input node and the ith Kohonen neuron,

x_j has the value of normalized component of the real input.

The WTA winner is the neuron with the maximal value of NET_i. This neuron generates on its output the signal equal to 1 blocking the rest of neurons and activates the Grossberg neurons via the weights v_{ij}. Grossberg neurons G_i obtain the weighted sum

$$G_i = \sum_{j=1}^{N} v_{ij} k_j$$

where

N is the number of Kohonen neurons,

v_{ij} is the connection value between jth Kohonen neuron and the ith Grossberg neuron,

k_j is the output value of jth Kohonen neuron ($k_j \in \{0,1\}$.

Because only one of Kohonen neurons is active (output signal equal to 1) the Grossberg layer generates at output the vector signal equal to the weight vector $\boldsymbol{v}_k$ determined by the weights values between the Kohonen neuron k and Grossberg neurons.

The Kohonen layer operates as a vector classifier, activating the neuron representing the class of similar input vectors. The Grossberg layer generates the vector $\boldsymbol{Y}$ associated with this class.

Every Grossberg neuron realizes the outstar function. When the winning neuron in the Kohonen layer has the number k_j, the output vector signal from the Grossberg layer is the binary value of k_j (for example, for k_j = 5 the output vector $\boldsymbol{Y} = [1\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$).

The number of Grossberg neurons depends on the number of Kohonen neuron. For $0 < \boldsymbol{N} \leq 256$ we have $\boldsymbol{M}$ = 8 (one byte) Grossberg neurons and for $\boldsymbol{N}$ > 256 (but of course $\boldsymbol{N} \leq$ 65536), $\boldsymbol{M}$ = 16 (two bytes).

Learning

Before compression it is necessary to learn the network. The learning procedure can be based in practice on any learning image (recommended are images with varied gray levels). The learning picture is segmented into clusters composed of 2×2, 3×3 or 4×4 pixels. Each pixel has a gray level between 0 and 63.

Dividing a picture into n rows and m columns generates $n \times m$ clusters. Each cluster is represented by a k-dimensional vector $\boldsymbol{X} = [x_1, x_2, \ldots, x_k]$ where k is the number of pixels in the cluster. The vector $\boldsymbol{X}$ has to be normalized before use.

Elements of $\boldsymbol{X}$ have the values depending on the pixels gray level. The Grossberg layer generates the number of the winning Kohonen neuron. This information defines the cluster number, which is able to reconstruct the components of the vector $\boldsymbol{X}' \cong \boldsymbol{X}$.

At the first stage the weights for the Kohonen layer are generated. The method consists of the division of learning vectors into classes composed of similar vectors. At the beginning there is only one Kohonen neuron. Its weights are assumed to be identical with the components of the first learning vector $\boldsymbol{W}_1 = \boldsymbol{X}_1$. Next, for each following learning vector $\boldsymbol{X}_i$ the products $\boldsymbol{X}_i \boldsymbol{W}_k$ are calculated, where $\boldsymbol{W}_k$, $(k = 1,2,\ldots)$, are the input weights of every Kohonen neuron already formed.

If $(1 - \boldsymbol{X}_i \boldsymbol{W}_k) < B$

where $B(B \in (1;0))$, defines the difference between the learning vectors, then ith learning vector is included into the class k, and the weights of kth neuron are modified. If this condition is not satisfied for any existing Kohonen neuron, the new Kohonen neuron is added with weights equal to the components of $\boldsymbol{X}_i$ representing this new class. The algorithm is repeated until the last learning vector is included in the proper class.

In the Grossberg layer we have the supervised learning procedure. For every Kohonen neuron the special vector $\boldsymbol{d}_i,\ i = 1,2,\dots,N$ is generated. This vector is the binary notation of the number of Kohonen neuron (for example for the neuron 6 we have $\boldsymbol{d}_6 = [\,0\ 1\ 1\ 0\ 0\ 0\ 0\ 0]$.

The weights $\boldsymbol{V}_k$ are calculated according to the outstar rule

$$v_{jk}(t+1) = v_{jk}(t) + \eta(d_j - v_{jk}(t))$$

where η is a learning coefficient,

v_{jk} is the connection value between kth Kohonen neuron and ith Grossberg neuron.

The compression method

The method follows the conventional image processing scheme which is on an orthogonal basis: By decomposing a large pictorial data into frames of subimages which are then analyzed for extraction of frequencies and identification of pictorial features in order to build a mapping table which can be used to reconstruct the image with the coordinate information. When using a CPN for this purpose, the Kohonen network identifies the pattern class each subimage belongs to and generates a class index by the winning node. The indices generated are of the same sequence that the subimages are fed in. The outstar network takes part in approximating class vectors representing these classes. The class index sequence generated and the weight matrix of the outstar layer are used to restore the image

References and further reading:

warsaw university of technology faculty of mathematics and information science/ Neural Networks course, http://www.mini.pw.edu.pl/~macukow/wspolne/nn/Lecture14.pdf [visited on January - 2020]

Chang, W., Soliman, H. S., & Sung, A. H. (1992, October). Image data compression using counterpropagation network. In *[Proceedings] 1992 IEEE International Conference on Systems, Man, and Cybernetics* (pp. 405-409). IEEE.