Laboratory 2 - additional materials
Kohonen Network for image compression

1. Task specification for image compression exercise

Write a program which implements Kohonen neural network which then should be used for image compression. The structure of the Kohonen network is depicted in the snapshots form Fig. 1, Fig. 2 and Fig. 3.

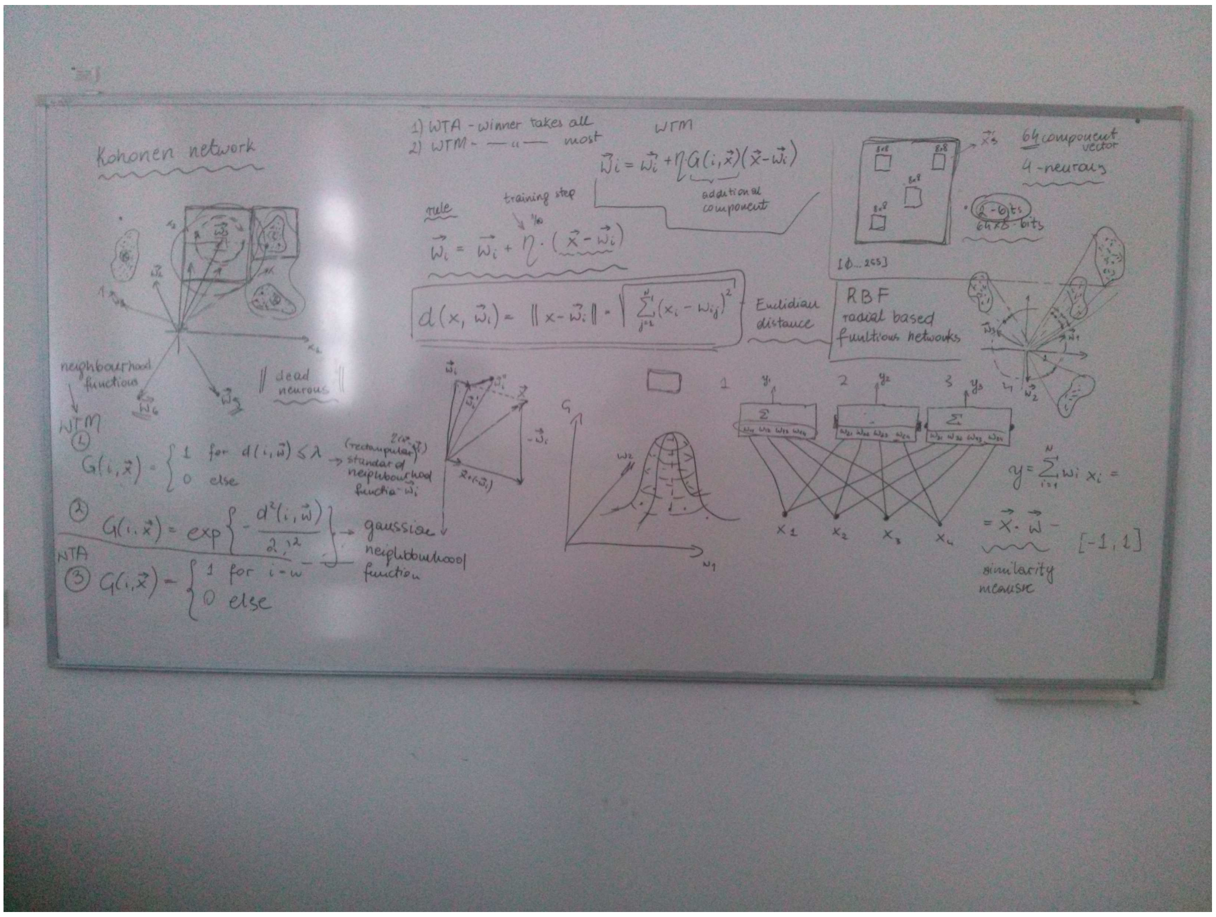


Fig. 1. The Kohonen network - snapshot 1

The program should use only 8-bit per pixel grayscale images (256 shades of gray) for simplicity and should be able to take into account the following parameters during its execution:

1. (square) size of the compression frame in pixels, which determines the number of neurons in the input layer, for example 4x4 or 8x8 or 16x16 pixels frames (16, 64 and 256 input neurons respectively)
2. number of neurons in the output (Kohonen) layer, which determines (along with the compression frame size) the compression ratio
3. small the training step
4. the number of random image frames which are used during the training

5. weather to normalize the training patterns or proceed with the training variant without normalization - this would affect the training method

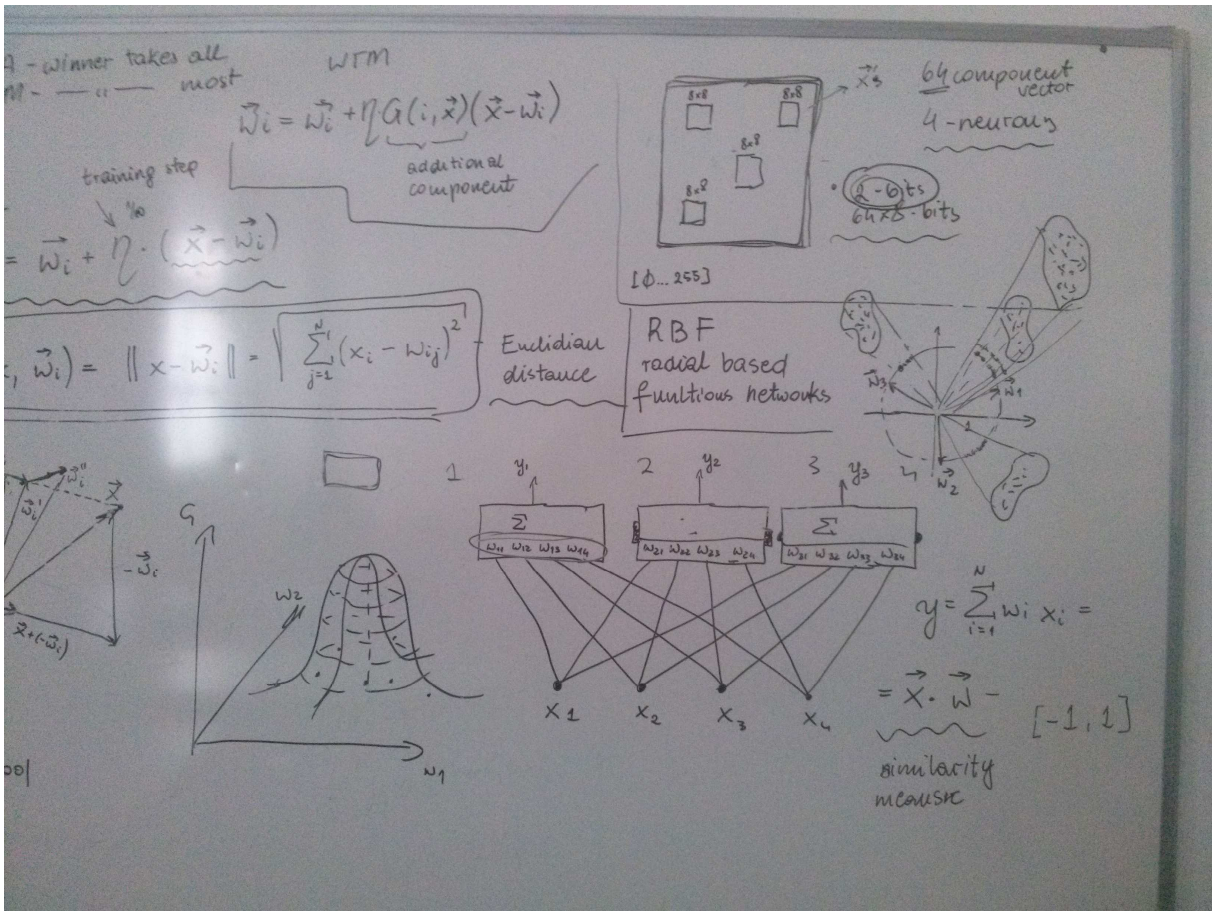


Fig. 2. The Kohonen network - snapshot 2

For a single configuration of the above parameters the program should run on a single image and for that single image it should output and report the following things:

1. the attained compression ratio, which is the ratio of the total number of bits needed to code of the original image pixel data to the total number of bits needed to code the compressed image pixel data
2. the value of the PSNR measure between the original image and the image which was decoded with the use of the trained Kohonen network, this measure can be calc
3. the decoded image in the form of a graphics file (of arbitrary format ex. *.bmp or *.jpg) to compare visually the decoded image with an input image

2. Remarks

Please note that in case of lack of the normalization you have to use external procedures when calculating the distances between the input signal and neurons' weights, i.e. you cannot use the network itself. In this case please use a standard Euclidian distance among two vectors. On the other hand, in case of the normalization you may use the Kohonen network for calculating those distances both during the training process and coding and decoding stages. Or even better, instead of using a standard Kohonen network you may use it's enhanced version, i.e. the Counter Propagation network depicted in Fig. 4. In this last case not only the calculation of the distances between the input signal and neurons' weights can be performed both during the training process and coding stage by the network itself, but also you can use the same network, i.e. it's last "lookup" layer, to perform coding of the input image after the training process has finished.

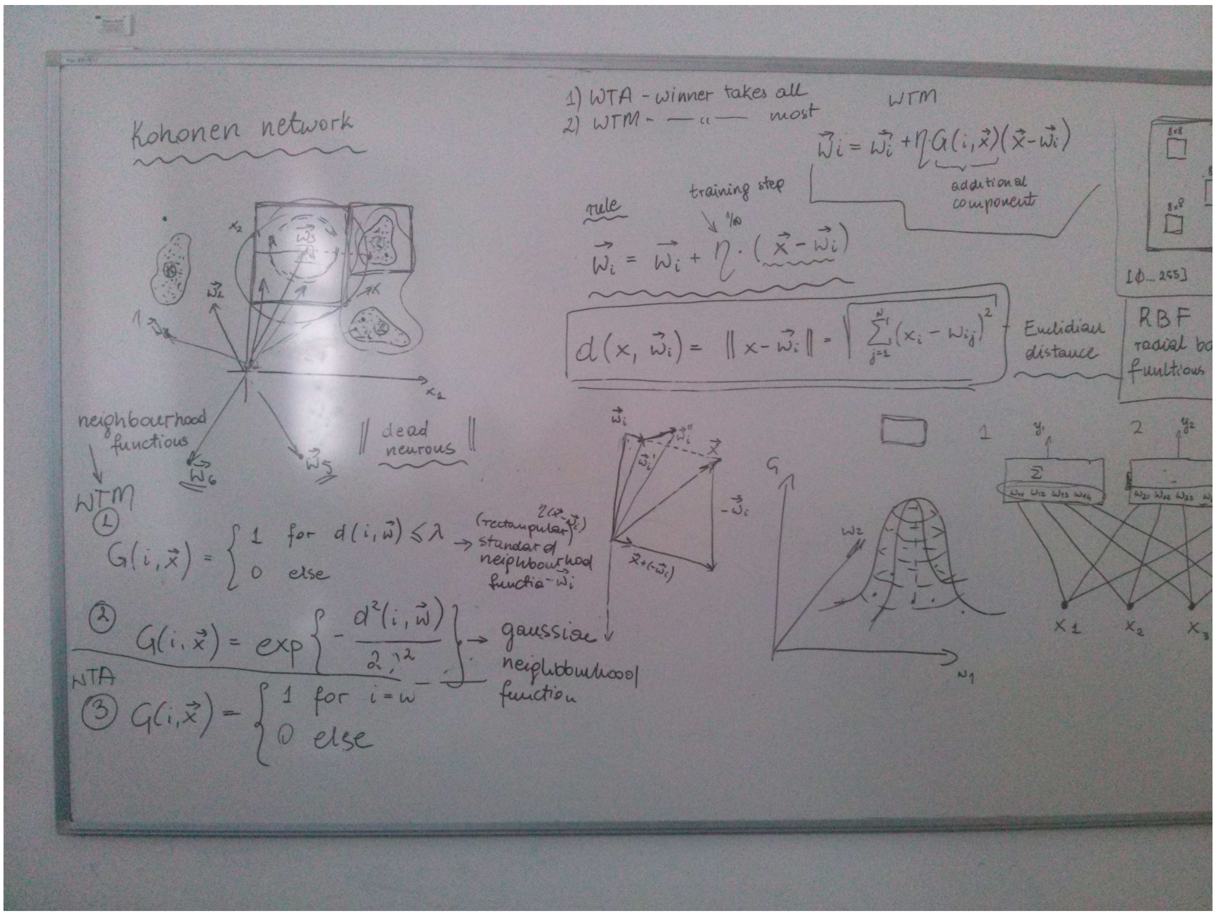
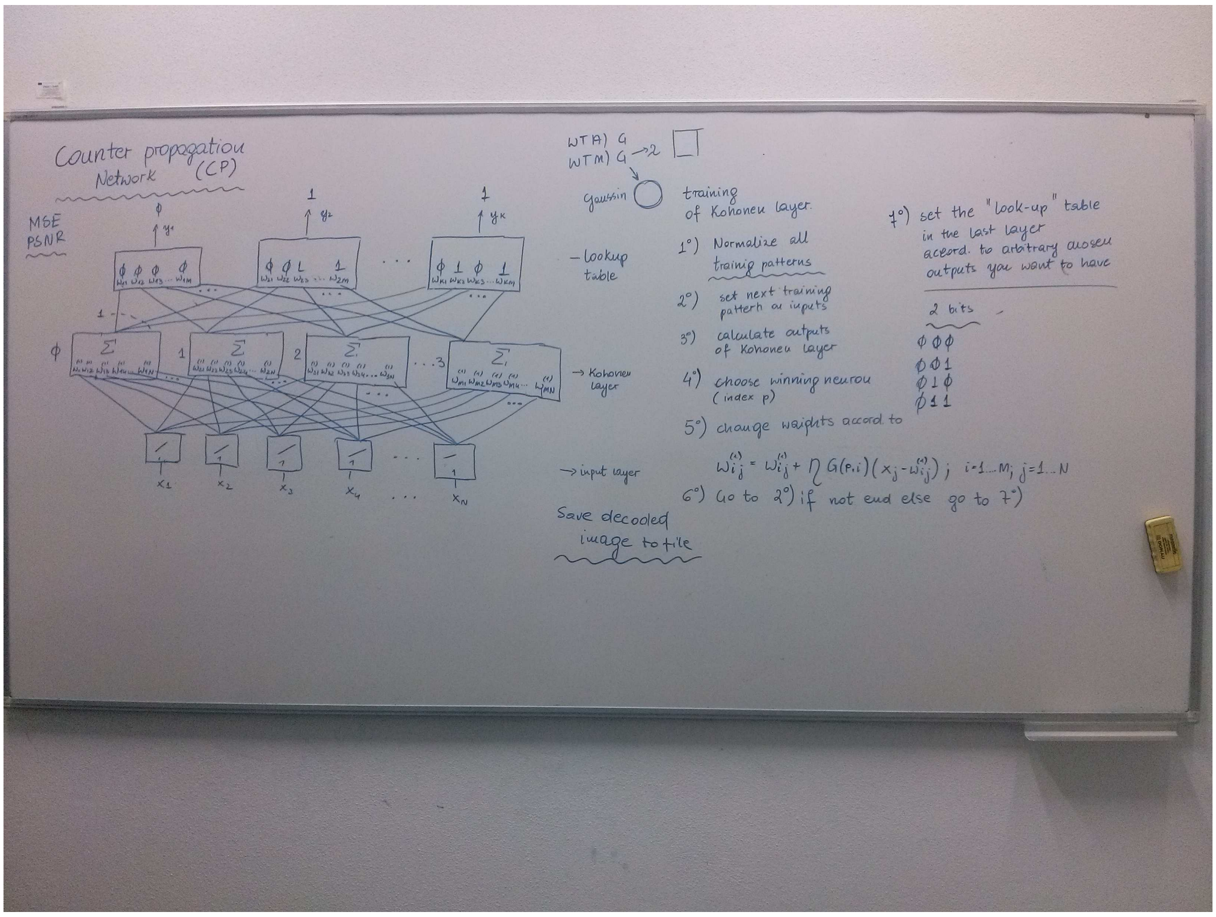

Kohonen network
1) WTA - winner takes all
2) WTM - " most
WTM
$\vec{w}_i = \vec{w}_i + \eta G(i,\vec{x})(\vec{x}-\vec{w}_i)$
additional component
rule
training step
$\vec{w}_i = \vec{w}_i + \eta \cdot (\vec{x}-\vec{w}_i)$
$d(x,\vec{w}_i) = \|x-\vec{w}_i\| = \sqrt{\sum_{j=1}^{N}(x_i - w_{ij})^2}$
Euclidian distance
RBF
radial ba
functions
[0...255]
neighbourhood functions
dead neurons
WTM
$G(i,\vec{x}) = \begin{cases} 1 & \text{for } d(i,\vec{w}) \leq \lambda \\ 0 & \text{else} \end{cases}$
(rectangular) standard neighbourhood function
$G(i,\vec{x}) = \exp\left\{-\frac{d^2(i,\vec{w})}{2\lambda^2}\right\}$
gaussian neighbourhood function
WTA
$G(i,\vec{x}) = \begin{cases} 1 & \text{for } i = w \\ 0 & \text{else} \end{cases}$
x_1
x_2
x_3
y_1
y_2

Fig. 3. The Kohonen network - snapshot 3


Counter propagation Network (CP)
MSE
PSNR
ϕ
y_1
1
y_2
1
y_K
ϕ ϕ ϕ ϕ
$w_{11}\ w_{12}\ w_{13} \dots w_{1M}$
ϕ ϕ L 1
$w_{21}\ w_{22}\ w_{23} \dots w_{2M}$
ϕ 1 ϕ 1
$w_{K1}\ w_{K2}\ w_{K3} \dots w_{KM}$
1
ϕ
1
2
...3
Σ
$w^{(1)}_{11}\ w^{(1)}_{12}\ w^{(1)}_{13}\ w^{(1)}_{14} \dots w^{(1)}_{1N}$
$w^{(1)}_{21}\ w^{(1)}_{22}\ w^{(1)}_{23}\ w^{(1)}_{24} \dots w^{(1)}_{2N}$
$w^{(1)}_{31}\ w^{(1)}_{32}\ w^{(1)}_{33}\ w^{(1)}_{34} \dots w^{(1)}_{3N}$
$w^{(1)}_{M1}\ w^{(1)}_{M2}\ w^{(1)}_{M3}\ w^{(1)}_{M4} \dots w^{(1)}_{MN}$
x_1
x_2
x_3
x_4
...
x_N
WTA) G
WTM) G → 2
Gaussin
– lookup table
→ Kohonen layer
→ input layer
Save decoded image to file

training of Kohonen layer

1°) Normalize all training patterns

2°) set next training pattern as inputs

3°) calculate outputs of Kohonen layer

4°) choose winning neuron (index p)

5°) change weights accord. to

$$w^{(1)}_{ij} = w^{(1)}_{ij} + \eta\, G(p,i)\left(x_j - w^{(1)}_{ij}\right); \quad i=1 \dots M; \ j=1 \dots N$$

6°) Go to 2°) if not end else go to 7°)

7°) set the "look-up" table in the last layer accord. to arbitrary chosen outputs you want to have

2 bits

ϕ ϕ ϕ
ϕ ϕ 1
ϕ 1 ϕ
ϕ 1 1

Fig. 4. The Counter Propagation network - snapshot 1