Accepted for *Mathematics and Computers in Simulation.* Predicted publication: MACTOM 41(5-6) July 1996.

Developments and Applications of the Self-Organizing Map and Related Algorithms

Jari Kangas and Teuvo Kohonen

Helsinki University of Technology
Neural Networks Research Centre
Rakentajanaukio 2 C, FIN-02150, Espoo, FINLAND
tel: +358 0 451 3266, fax: +358 0 451 3277
email: Jari.Kangas@hut.fi, Teuvo.Kohonen@hut.fi

Abstract In this paper the basic principles and developments of an unsupervised learning algorithm, the Self-Organizing Map (SOM) and a supervised learning algorithm, the Learning Vector Quantization (LVQ) are explained. Some practical applications of the algorithms in data analysis, data visualization and pattern recognition tasks are mentioned. In the end of the paper new results are reported about increased error tolerance in the transmission of vector quantized images, provided by the topological ordering of codewords by the SOM algorithm.

I. INTRODUCTION

The Self-Organizing Map (SOM) [1][2][3] defines a nonparametric regression of a set of codebook vectors onto the input signal samples. It is a kind of nonlinear projection of a probability density function of high-dimensional input data onto a two-dimensional array. An important application of SOM is visualization of complex high-dimensional data, such as process states. Being a special clustering method, the SOM can also find abstractions from the raw data.

Learning Vector Quantization (LVQ) [2][3] is a group of algorithms applicable to statistical pattern recognition, in which the classes are described by a relatively small number of codebook vectors, properly placed within each zone such that the decision borders are approximated by the nearest-neighbor rule. Unlike in normal k-nearest-neighbor (k-nn) classification, the original samples are not used as codebook vectors, but they tune the latter. LVQ is concerned with the optimal placement of such codebook vectors into class zones.

The SOM and LVQ are paradigms in neural-network theory. Learning in the SOM is unsupervised, and in the LVQ supervised, respectively. Both of the above algorithms have been applied to a great many practical applications, and also special hardware, such as VLSI chips, have been designed for them. Freely available program packages [4] and [5] contain source codes for the algorithms.

This presentation expounds basic principles and special developments of the SOM and LVQ, and exemplifies their use by a few practical applications, such as speech recognition and analysis, interpretation of EEG data, visualization of machine faults, and classification of cloud types. A new application reported in this paper is transmission of vector-quantized images, whereby the topological ordering of codes by the SOM provides for high level of error tolerance.

II. THE PRINCIPLE OF THE SOM

There exist many versions of the Self-Organizing Map (SOM). The "basic" SOM defines a mapping from the input data space $\Re^n$ onto a regular two-dimensional array of nodes. With every node i, a parametric reference vector $m_i \in \Re^n$ is associated. An input vector $x \in \Re^n$ is compared with all the m_i, and the best match is defined as "response": the input is thus mapped onto this location. The comparison may be made in any metric; in practical applications the Euclidean distances $||x-m_i||$ usually define the best-matching node m_c:

$$\begin{aligned} ||x - m_c|| &= \min_i \{||x - m_i||\}, \quad \text{or} \\ c &= \arg\min_i \{||x - m_i||\}. \end{aligned} \tag{1}$$

An "optimal" mapping maps the probability density function $p(x)$ in the most "faithful" fashion, trying to preserve at least the local structures of $p(x)$. Definition of such m_i values, however, is far from trivial; a number of people have tried to define them as optima of some objective (energy) function (see e.g. [6], [7], [8], and [9]).

During learning, those nodes that are topographically close in the array will learn from the same input. Without mathematical proof we state that useful values of the m_i can be found as convergence limits of the following learning process, whereby the initial values of the $m_i(0)$ can be arbitrary (random or regular):

$$m_i(t+1) = m_i(t) + h_{ci}(t)[x(t) - m_i(t)], \tag{2}$$

where t is the discrete-time coordinate, and $h_{ci}(t)$ is a neighborhood kernel defined over the lattice points. Usually $h_{ci}(t) = h(||r_c - r_i||, t)$, where $r_c \in \Re^2$ and $r_i \in \Re^2$ are the location vectors of nodes c and i, respectively, in the array. With increasing $||r_c - r_i||$ and time, $h_{ci}(t) \to 0$. The average width and form of h_{ci} defines the "stiffness" of the "elastic surface" to be fitted to the data points. Notice that it is usually not necessary to describe the exact form of $p(x)$, especially if x is very-high-dimensional; it is more important to be able to automatically find those dimensions and domains in the signal space where x has significant amounts of sample values.

Two definitions of $h_{ci}(t)$ are usually given. The simpler of them refers to a neighborhood set $N_c(t)$ around node c, whereby $h_{ci}(t) = \alpha(t)$ if $i \in N_c(t)$ and $h_{ci}(t) = 0$ if $i \notin N_c(t)$, where $\alpha(t)$ is some monotonically decreasing function

of time ($0 < \alpha(t) < 1$). Another widely applied neighborhood kernel can be written

$$h_{ci}(t) = \alpha(t) \cdot \exp\left(-\frac{||r_c - r_i||^2}{2\sigma^2(t)}\right) \quad , \tag{3}$$

where $\alpha(t)$ is as above, and $\sigma(t)$ defines the width of the kernel corresponding to the radius of $N_c(t)$ above; $\sigma(t)$ is some monotonically decreasing function of time.

The last step in training is calibration of the map, to be able to identify images of input data items on it. By inputting a number of typical, manually analyzed data sets and looking where the best matches lie, at least some of the nodes can be labeled to delineate a "coordinate system", a set of characteristic reference points on the map. Since the mapping is assumed to be continuous along the "elastic surface", the unknown data can be interpreted by means of interpolation and extrapolation from the calibrated points.

III. THE PRINCIPLE OF THE LVQ

The Self-Organizing Map algorithm is directly applicable, e.g., to visualization applications. If an application requires pattern recognition, then different algorithms are needed. The SOM creates an approximation of the probability density function of all the input samples, whereas the classification task requires an approximation of the optimal decision borders between the classes.

Learning Vector Quantization (LVQ) is a group of algorithms applicable to statistical pattern recognition, in which the classes are described by a relatively small number of codebook vectors. In the following three variations for the algorithms, the LVQ1 [2], the LVQ2.1 [10] and the LVQ3 [2], are explained.

A. The LVQ1 algorithm

Assume that a number of 'codebook vectors' m_i are placed into the input space to approximate various domains of the input vector x. Usually several codebook vectors are assigned to each class, and the input vector x is then classified to the same class to which the nearest m_i belongs (m_c as given by Equation (1)). Values of the m_i that approximately minimize the misclassification errors in the above nearest-neighbor classification can be found in the following learning process:

$$\begin{aligned}
m_c(t+1) &= m_c(t) + \alpha(t)[x(t) - m_c(t)] \\
&\quad \text{if } x \text{ and } m_c \text{ belong to the same class,} \\
m_c(t+1) &= m_c(t) - \alpha(t)[x(t) - m_c(t)] \\
&\quad \text{if } x \text{ and } m_c \text{ belong to different classes,} \\
m_i(t+1) &= m_i(t) \text{ for } i \neq c.
\end{aligned} \tag{4}$$

Here $0 < \alpha(t) < 1$, and $\alpha(t)$ usually decreases monotonically with time. Initially we might have $\alpha(0) < 0.1$.

B. The LVQ2.1 algorithm

The classification decision is identical with that of the LVQ1. In learning, however, two nearest codebook vectors to x, m_i and m_j, are now updated. One of them must belong to the correct class and the other to a wrong class, respectively. Moreover, x must fall in a 'window', which is defined around the midplane of m_i and m_j: When d_i and d_j are the Euclidean distances of x from m_i and m_j, respectively, x falls in a 'window' of relative width w if

$$\min(\frac{d_i}{d_j}, \frac{d_j}{d_i}) > s, \text{ where } s = \frac{1-w}{1+w}. \tag{5}$$

A 'window' width w of 0.2 to 0.3 is recommendable.

The learning algorithm is given by:

$$\begin{aligned} m_i(t+1) &= m_i(t) - \alpha(t)[x(t) - m_i(t)], \\ m_j(t+1) &= m_j(t) + \alpha(t)[x(t) - m_j(t)], \end{aligned} \tag{6}$$

where m_i and m_j are the two closest codebook vectors to x, whereby x and m_j belong to the same class, while x and m_i belong to different classes, respectively. Furthermore x must fall into the 'window'.

C. The LVQ3 algorithm

The LVQ2 algorithm was based on the idea of *differentially* shifting the decision borders towards the Bayes limits, while no attention was paid to what might happen to the location of the m_i in the long run. Therefore it seems necessary to ensure that the m_i continue approximating the class distributions. The following algorithm takes care of this:

$$\begin{aligned} m_i(t+1) &= m_i(t) - \alpha(t)[x(t) - m_i(t)], \\ m_j(t+1) &= m_j(t) + \alpha(t)[x(t) - m_j(t)], \end{aligned} \tag{7}$$

if the conditions for LVQ2-learning apply, and

$$m_k(t+1) = m_k(t) + \epsilon\alpha(t)[x(t) - m_k(t)], \tag{8}$$

for $k \in \{i, j\}$, if x, m_i, and m_j belong to the same class. In a series of experiments, applicable values of ϵ between 0.1 and 0.5 were found. The optimal value of ϵ seems to depend on the size of the window, being smaller for narrower windows.

D. Differences between the basic algorithms

The three options for the LVQ-algorithms yield almost similar accuracies. The LVQ1 and the LVQ3 define a more robust process, whereby the codebook vectors assume stationary values in extended learning periods. For the LVQ1 the learning rate can approximately be optimized for quick convergence. In the LVQ2.1, the relative distances of the codebook vectors from the class borders are optimized. The LVQ2.1 should only be used in a differential fashion, using a small value of learning rate and a relatively low number of training steps.

IV. APPLICATIONS OF SOM AND LVQ ALGORITHMS

In the following we mention some applications of the SOM and LVQ algorithms. For a more comprehensive list of application papers, see [3] or [11].

A. Speech recognition and analysis

In [12] it was shown that the SOM created a representation of the spectra of the speech samples; different spectra were mapped to different parts of the map. After fine tuning with the LVQ algorithm, the maps can be used in speech recognition systems [13][14]. The visualization properties of the SOM have been utilized also in voice analysis applications [15]. In these the trajectory of successive best matching units in the map have been used as an indication of the voice quality.

B. Interpretation of EEG data

Designing analysis tools for the EEG signals is difficult, as the most descriptive features seem to be largely unknown. In [16] the Self-Organizing Map was used to monitor the EEG signal as a trajectory on a two-dimensional display. It was shown that one is able to identify certain overall states from the EEG signal. The SOM was used also in [17], where the sleep EEG was analyzed. Decisive features in the sleep EEG can be identified by clustering the data by the SOM and observing the transitions between the clusters.

C. Visualization of machine faults

The SOM algorithm might also be used to visualize multi-dimensional online measurements from any dynamic process. In [18] the applicability of the SOM algorithm for process state monitoring in a chemical process was explained. Another preliminary study of the usage of the SOM algorithm was explained in [19], where error states were detected based on the quantization error of the weight vectors vs. the input feature vector. A similar system was also explained in [20].

D. Classification of cloud types and timber

The SOM algorithm can be used for the analysis of stochastic textures. In [21] some application areas were described. These included an application where cloud types were determined from satellite images [22]. Another application was the inspection of timber in saw-mills [23] for the determination of quality and price.

V. SOMS IN ERROR TOLERANT TRANSMISSION OF VECTOR QUANTIZED IMAGES

Image compression is an essential task for image storage and transmission [24]. A review of compression methods using Vector Quantization (VQ) techniques has recently appeared [25]. VQ methods offer good performance when

high compression rates are needed. In VQ approaches, adjacent pixels are taken as a single block that is mapped into a finite set of codewords. In the decoding stage the codewords are replaced by corresponding model vectors. The set of codewords and the associated model vectors together is called a codebook (see Figure 1). In VQ the correlations that exist between adjacent pixels in a block are naturally taken into account, and with a comparatively small codebook one achieves a small quantization error in the reconstructed image.

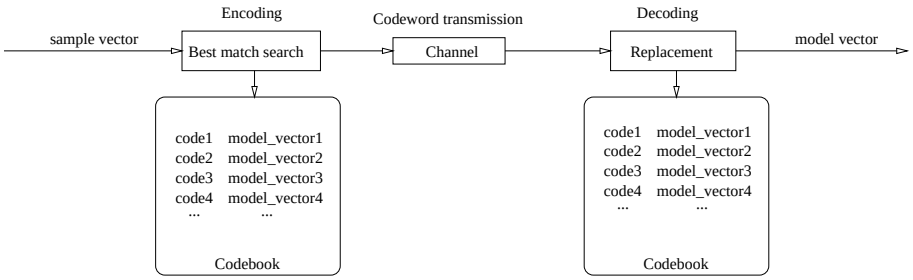

Figure 1: *Vector quantization in image compression application. A sample vector (a subimage) is compared with all model vectors in the codebook and the best-matching one is selected. The corresponding codeword is sent to the transmission channel. In the decoding stage the model vector associated with the codeword is searched from the codebook and used in building the reconstructed image. The two codebooks are identical.*

A. SOMs in vector quantization of images

The basic idea in vector quantization is to minimize the mean quantization error. Perhaps the best known VQ method is the Linde-Buzo-Gray (LBG) algorithm [26], which iteratively searches for a fixed number of clusters in the training data. The cluster centroids are used as the codebook model vectors, while the codewords for them can be selected arbitrarily.

The Self-Organizing Map (SOM) algorithm has recently been suggested for vector quantization [27][28]. In the SOM, the weight vectors of the neurons form the codebook while the coordinates of the SOM array can directly be used as the codewords. The distortion errors in using codebooks produced by both the SOM algorithm and the LBG algorithm are rather similar [27][28]; however, in [28], the SOM algorithm was found to be more robust with respect to initialization than the LBG algorithm.

The most important difference between the LBG and SOM algorithms, however, is the following. The LBG algorithm does not define any order in the codebook; the codewords for model vectors can be selected arbitrarily. Contrary to that, the codebook trained by the SOM algorithm has acquired an internal order; adjacent codebook entries have similar codewords and similar model vectors. This order can be utilized to increase error tolerance and to improve image compression efficiency [29][30][31][32]. Because the similarities between adjacent blocks in the original image are converted to similarities between the corresponding codewords, prediction coding of the codewords can

further be used to improve compression rates by 15 to 20 percent [31].

B. Error tolerance using SOMs

Assume that a vectorial data item x is converted to some codeword y and then back to the decoded vector x'. If the codeword y is transmitted through a noisy transmission channel, the errors may change y to some other codeword y'. When simple, unordered VQ methods are used, the erroneous codeword will usually be decoded into a whole block of completely erroneous pixels.

In an error-tolerant codebook, the degrading effects of the channel errors are minimized by a proper selection of codewords for the model vectors: *With increasing degree of errors in the codewords, the similarity of the decoded model vectors is made to change gradually.*

In [7] it was shown that it is possible to derive a stochastic gradient descent algorithm for the codebook design, based on the theory of errors. The algorithm (where the training neighborhood was defined through the likelihood of changing the selected codeword to other codewords) turned out to be identical to the Self-Organizing Map training algorithm. *The SOM algorithm therefore leads to a codebook in which the distortion effects are automatically taken into account.*

The benefit of an error-tolerant codebook in data transmission were first demonstrated in [33], in an experiment where speech amplitude values were quantized and random bit substitutions for codewords could happen. The SOM algorithm was used to order the codebook entries. It was shown that the ordered codebook gave a 3 dB reduction in the reconstruction distortion compared to random codeword schemes.

C. Present experiments with image data

One of the authors has recently suggested SOMs for the transmission of image data [34]. The two identical codebooks, one for encoding and the other for decoding, respectively, consisted of 512 entries for 4 by 4 pixel subimages. A three-dimensional SOM was used for the codebook, and the three coordinates of the neurons in the array (with three bits each) constituted the codeword to be transmitted through the channel.

Transmission through the channel was defined as follows: Each codeword (nine bits) was divided into three three-bit subwords, each of which was converted into an eight-level analog signal. In the simple channel noise model used in this experiment the signal was assumed to deviate randomly by one level either upward or downward, with a probability of p. The probability of at least one error to occur in a given codeword was then $P(p) = 1.0 - (1.0 - \frac{7}{4}p)^3$.

This same codebook could be used for comparison, for the randomly-ordered codebook experiment, by relabeling the model vectors. A new, unique, random nine-bit codeword was redefined to each entry.

The codebooks were trained using 28 photographic images of faces (both males and females) and tested using another face image which was not contained in the training set. All the images were 512 by 680 pixel, 256 gray level images.

Distortion in the decoded images was measured using a peak signal-to-noise ratio (PSNR). It was defined as

$$\mathrm{PSNR} = 10 \log \frac{255^2}{\mathrm{MSE}} \ \mathrm{dB}, \tag{9}$$

where MSE is the mean square error in the reconstructed image. The PSNR values after transmission through the noisy channel model having different probabilities of error are tabulated on Table 1.

Probability p of an error	PSNR in SOM code	PSNR in random code	Difference in PSNRs
No error	34.31	34.31	0.00
0.0001	34.29	33.56	0.73
0.0010	34.15	30.08	4.07
0.0100	32.83	21.94	10.89
0.1000	27.29	12.46	14.83

Table 1: *Peak signal-to-noise ratios (in dB) for different levels of noise.*

The images seen in Figure 2 have been transmitted through a moderately noisy channel where the probability for an error was $p = 0.01$ ($P(0.01) = 0.052$, i.e. over 5 % of the codewords were changed). In the image with random order in codewords the errors are usually rather severe; e.g., in the middle of dark areas there are white blocks, and dark blocks in white areas, respectively. In the image with error-tolerant coding the errors are of different nature; for instance, in the dark area the erroneous blocks are never white, but "almost" dark. The subjectively experienced qualities of the images differ significantly, although the same frequencies of errors were present in both of the images.

In Figure 3 another exemplary pair of images is shown. The probability for an error was $p = 0.1$ ($P(0.1) = 0.438$, i.e. almost half of the codewords were changed).

D. Conclusions

From the results one can clearly see that degradation of images due to errors in codewords can significantly be reduced if the codewords are ordered. The SOM algorithm can be used to order the codewords automatically. The present simulations demonstrated that the codebooks designed by the SOM are superior to unordered VQ codebooks.

REFERENCES

[1] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

Figure 2: *The vector quantized images after transmission through a moderately noisy ($p = 0.01$) channel. The image with ordered codebook is on the left and the image with unordered codebook is on the right.*

[2] T. Kohonen. The self-organizing map. *Proceedings of IEEE*, 78:1464–1480, 1990.

[3] T. Kohonen. Self-Organizing Maps. *Springer Series in Information Sciences*, Vol. 30, 1995.

[4] SOM_PAK: The Self-Organizing Map Program Package. Obtainable via anonymous ftp from the internet address "cochlea.hut.fi", 1995.

[5] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola. LVQ_PAK: A program package for the correct application of Learning Vector Quantization algorithms. In *Proceedings of Int Jt Conf on Neural Networks*, pages I–725–730, 1992.

[6] H. Ritter and K. Schulten. Kohonen self-organizing maps: exploring their computational capabilities. In *Proceedings of IEEE Int Conf on Neural Networks, San Diego*, volume I, pages 109–116, 1988.

[7] S. P. Luttrell. Derivation of a class of training algoritms. *IEEE Transactions on Neural Networks*, 1(2):229–232, June 1990.

[8] T. Kohonen. Self-Organizing Maps: Optimization approaches. In *Artificial Neural Networks*, pages II–981–990. North-Holland, 1991.

[9] E. Erwin, K. Obermayer, and K. Schulten. Self-organizing maps: Ordering, convergence properties and energy functions. *Biological Cybernetics*, 67(1):47–55, 1992.

[10] T. Kohonen. Improved versions of Learning Vector Quantization. In *Proceedings of Int Jt Conf on Neural Networks, San Diego, June 17-21*, pages I–545–550, 1990.

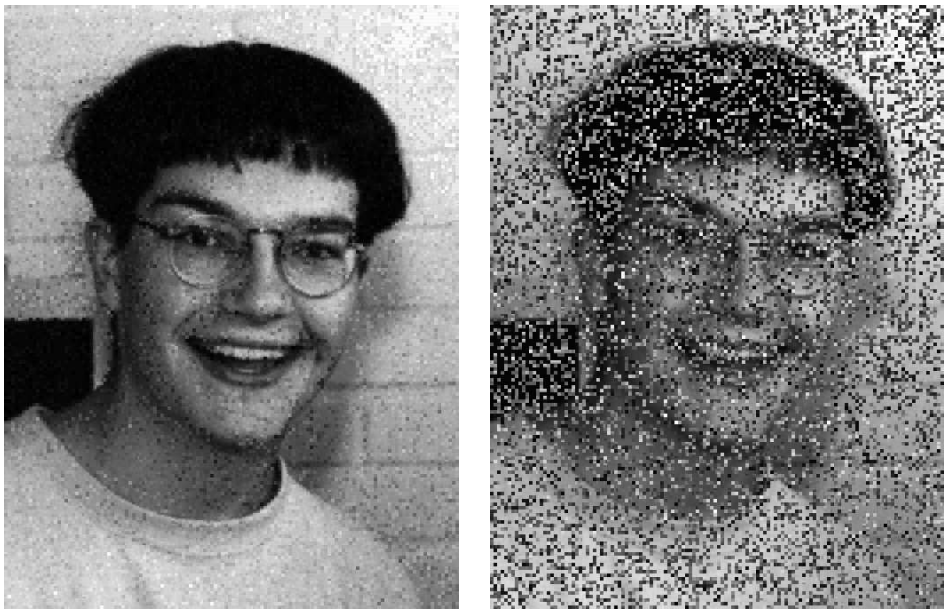

Figure 3: *The vector quantized images after transmission through a very noisy ($p = 0.1$) channel. The image with ordered codebook is on the left and the image with unordered codebook is on the right.*

[11] Reference list of 1213 studies of the Self-Organizing Map (SOM) and Learning Vector Quantization (LVQ). Obtainable via anon. ftp from the internet address "cochlea.hut.fi", 1993.

[12] T. Kohonen. Clustering, taxonomy, and topological maps of patterns. In *Proc of the 6th IEEE Int Conf on Pattern Recognition*, pages 114–128, 1982.

[13] T. Kohonen. The 'neural' phonetic typewriter. *Computer*, 21(3):11–22, 1988.

[14] K. Torkkola, J. Kangas, P. Utela, S. Kaski, M. Kokkonen, M. Kurimo, and T. Kohonen, Status Report of the Finnish Phonetic Typewriter Project. In *Artificial Neural Networks*, pages 771–776, Amsterdam, 1991. North-Holland.

[15] L. Leinonen, J. Kangas, K. Torkkola, and A. Juvas. Dysphonia detected by pattern recognition of spectral composition. *Journal of Speech and Hearing Research*, 35:287–295, April 1992.

[16] S. Kaski and S.-L. Joutsiniemi. Monitoring EEG signal with the self-organizing map. In *Proceedings of Int Conf on Artificial Neural Networks*, pages 974–977, London, 1993. Springer-Verlag.

[17] S. Roberts and L. Tarassenko. Analysis of the sleep EEG using a multi-layer network with spatial organisation. *IEE Proceedings F [Radar and Signal Processing]*, 139(6):420–425, December 1992.

[18] V. Tryba and K. Goser. Self-Organizing Feature Maps for process control in chemistry. In *Artificial Neural Networks*, pages 847–852, Amsterdam, 1991. North-Holland.

[19] J. Alander, M. Frisk, L. Holmström, A. Hämäläinen, and J. Tuominen. Process error detection using self-organizing feature maps. In *Artificial Neural Networks*, pages II–1229–1232. North-Holland, 1991.

[20] M. Kasslin, J. Kangas, and O. Simula. Process state monitoring using self-organizing maps. In *Artificial Neural Networks, 2*, pages II–1531–1534. North-Holland, 1992.

[21] O. Simula and A. Visa. Self-organizing feature maps in texture classification and segmentation. In *Artificial Neural Networks, 2*, pages II–1621–1628. North-Holland, 1992.

[22] O. Simula, A. Visa, and K. Valkealahti. Operational cloud classifier based on the topological feature map. In *Proceedings of Int Conf on Artificial Neural Networks, Amsterdam*, pages 899–902, London, 1993. Springer-Verlag.

[23] I. Yläkoski and A. Visa. A two-stage classifier for wooden boards. In *Proc. of The 8th Scandinavian Conf on Image Analysis*, pages I–637–641, 1993.

[24] A. K. Jain. Image data compression: A review. *Proceedings of the IEEE*, 69(3):349–389, 1981.

[25] N. M. Nasrabadi and R. A. King. Image coding using vector quantization: A review. *IEEE Transactions on Communications*, 36(8):957–971, 1988.

[26] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1:4–29, April 1984.

[27] N. M. Nasrabadi and Y. Feng. Vector quantization of images based upon the Kohonen self-organization feature maps. *Neural Networks*, 1(1 SUPPL):518, 1988.

[28] J. D. McAuliffe, L. E. Atlas, and C. Rivera. A comparison of the LBG algorithm and Kohonen neural network paradigm for image vector quantization. In *Proc. ICASSP-90, Int. Conf. on Acoustics, Speech and Signal Processing*, pages IV–2293–2296. IEEE, 1990.

[29] E. A. Riskin, L. E. Atlas, and S.-R. Lay. Ordered neural maps and their applications to data compression. In B. H. Juang, S. Y. Kung, and C. A. Kamm, editors, *Proc. Workshop on Neural Networks for Signal Processing*, pages 543–551, Piscataway, NJ, 1991. IEEE Service Center.

[30] S. Carrato, G. L. Siguranza, and L. Manzo. Application of ordered codebooks to image coding. In *Neural Networks for Signal Processing. Proc. 1993 IEEE Workshop*, pages 291–300, Piscataway, NJ, 1993. IEEE Service Center.

[31] G. Burel and J.-Y. Catros. Image compression using topological maps and MLP. In *Proc. ICNN'93, Int. Conf. on Neural Networks*, volume II, pages 727–731, Piscataway, NJ, 1993. IEEE Service Center.

[32] S. Carrato. Image vector quantization using ordered codebooks: Properties and applications. *Signal Processing*, 40(1):87–103, 1994.

[33] D. S. Bradburn. Reducing transmission error effects using a self-organizing network. In *Proceedings of Int Jt Conf on Neural Networks*, pages II–531–537, Piscataway, NJ, 1989. IEEE.

[34] J. Kangas. Self-organizing maps in error tolerant transmission of vector quantized images. Technical Report A21, Helsinki Univ. of Tech., Lab. of Comp. and Inf. Science, SF-02150 Espoo, Finland, 1993.