

Chapter 2

A Fuzzy Kwan–Cai Neural Network for Determining Image Similarity and for the Face Recognition

2.1 Introduction

Similarity is a crucial issue in Image Retrieval [1–4]. It is relevant both for unsupervised clustering [5, 6] and for supervised classification [7]. In this study, we aim to provide an effective method for learning image similarity. To reach this aim we start of from the Fuzzy Kwan–Cai Neural Network (FKCNN) [8] and turn it into a supervised method for learning similarity. Unlike the classical unsupervised FKCNN [8], in which a class is represented by a single output neuron, the supervised FKCNN has more output neurons that each designate a class. These give a better performance than their unsupervised counterparts as in the case of the classical unsupervised FKCNN, a class is represented by a single output neuron while the supervised FKCNN has more output neurons that each designate a class. This concept is similar with the idea of replacing the binary decision about the membership (nonmembership) of a pattern to a class, with the introduction of a membership degree, between 0 and 1.

In order to evaluate the performance of our proposed neural network, it is compared with two baseline methods: Self-organizing Kohonen maps (SOKM) and k -Nearest Neighbors (k -NN).

The feasibility of the presented methods for similarity learning has been successfully evaluated on the Visual Object Classes (VOC) database [9], that consists of 10102 images in 20 object classes.

The concept of similarity is important not only in almost every scientific field but it has [10, 11] deep roots in philosophy and psychology. Our work however deals more with the measure of similarity in computer science domain (Information Retrieval to be more specific, that has focused on images, video, and to some extent audio).

“Measuring image similarity is an important task for various multimedia applications.”¹

¹Perkiö, J., and Tuominen, A., and Myllymäki, P., Image Similarity: From Syntax to Weak Semantics using Multimodal Features with Application to Multimedia Retrieval, *Multimedia Information Networking and Security*, 2009, 1, 213–219.

Use of adequate measures improves [10] the accuracy of information selection. As there are a lot of ways to compute similarity or dissimilarity among various object representations, they need to be categorized [10] as:

(1) *distance-based similarity measures*

Example of this approach include following models/methods: Minkowski distance, Manhattan/City block distance, Euclidean distance, Mahalanobis distance, Chebyshev distance, Jaccard distance, Dice’s coefficient, cosine similarity, Hamming distance, Levenshtein distance, soundex distance.

(2) *feature-based similarity measures (contrast model)*

This method (suggested by Tversky in 1977) constitutes an alternate to distance-based similarity measures, where the similarity is computed by common features of compared entities. The entities are more similar if they share more common features and dissimilar in case of more distinctive features. The following formula can be used to determine similarity between entities A and B:

$$s(A, B) = \alpha g(A \cap B) - \beta g(A - B) - \gamma g(B - A), \quad (2.1)$$

where α, β, γ are used to determine the respective weights of associated values, $g(A \cap B)$ represents the common features in A and B, $g(A - B)$ represents distinctive features of A and $g(B - A)$ that of entity B.

(3) *probabilistic similarity measures*

In order to calculate relevance among some complex data types, use of the following probabilistic similarity measure are required: maximum likelihood estimation, maximum a posteriori estimation.

(4) *extended/additional measures*: similarity measures based on fuzzy set theory [12], similarity measures based on graph theory, similarity-based weighted nearest neighbors [13, 14], similarity-based neural networks [11].

Next to the definition of similarity measures [15], “in the last few decades, the perception of similarity received a growing attention from psychological researchers.”²

As the investigation of similarity is crucial for many classification methods, more recently, learning similarity measure [16] has also attracted attention in the machine learning community [17, 18].

“The problem of classifying samples based only on their pairwise similarities can be divided into two subproblems: measuring the similarity between samples and classifying the samples based on their pairwise similarities.”³

²Melacci, S., and Sarti, L., and Maggini, M. and Bianchini, M., A Neural Network Approach to Similarity Learning, ANNPR 2008, LNAI 5064, 133–136.

³Cazzanti, L., Generative Models for Similarity-based Classification, 2007, http://www.mayagupta.org/publications/cazzanti_dissertation.pdf.

“Similarity-based classifiers estimate the class label of a test sample based on the similarities between the test sample and a set of labeled training samples, and the pairwise similarities among the training samples.”⁴

Similarity-based classification is useful for problems in Multimedia Analysis [19, 20], Computer Vision, Bioinformatics, Information Retrieval [21–23], and a broad range of other fields. “Among the various traditional approaches of pattern recognition [24], the statistical approach has been most intensively studied and used in practice. More recently, the addition of artificial neural network techniques theory have been receiving significant attention. The design of a recognition system requires careful attention to the following issues: definition of pattern classes, sensing environment, pattern representation, feature extraction and selection, cluster analysis, classifier design and learning, selection of training and test samples, and performance evaluation.”⁵

NNs methods hold great promise for defining similarity.

In this chapter [1], two similarity learning approaches based on Artificial Neural Networks (ANNs) are presented. We have been very motivated to write the present paper to highlight the fact that neural networks outperform statistical techniques in computing similarities, too. The main contributions of this chapter are:

- we provide a neural method for learning image similarity, by turning the classical unsupervised Fuzzy Kwan–Cai Neural Network (FKCNN) into a supervised method, which gives a better performance than its unsupervised counterparts;
- we build a novel algorithm based on an evaluation criteria to compare the performances of the three presented methods;
- we test the resulting similarity functions on the PASCAL Visual Object Classes (VOC) data set, which consists in 20 object classes;
- we perform a comparative study of the proposed similarity learning method and compare it with two baseline methods: Self-organizing Kohonen Maps (SOKM) and k-Nearest Neighbor rule (k-NN) in terms of their ability to learn similarity; these particular algorithms (k-NN and SOKM) have been chosen as benchmark for performance comparison of the improved FKCNN algorithm because of the fact that all of them (k-NN, which is a nonneural method, SOKM—an unsupervised neural network, FKCNN—a supervised neural network, resulted by improving the unsupervised FKCNN) are some *extended/additional measures* to compute similarity or dissimilarity among various object representations.
- we highlight the overall performance of FKCNN, being better for our task than SOKM and k-NN.

⁴Chen, Y., and Garcia, E.K., and Gupta, M.Y., and Rahimi, A., and Cazzanti, A., Similarity-based Classification: Concepts and Algorithms, *Journal of Machine Learning Research*, 2009, 10, 747–776.

⁵Basu, J.K., and Bhattacharyya, D., and Kim, T.H., Use of Artificial Neural Network in Pattern Recognition, *International Journal of Software Engineering and Its Applications*, 2010, 4(2), 22–34.

The ANNs are configured [25] for a specific application, such as pattern recognition [26] or data classification, through a learning process. “The advantages for using an ANN in similarity matching are twofold: one that we can combine multiple features extracted with different methods, and the second that the combination of these features can be nonlinear.”⁶

2.2 Related Work

In past few years, many papers have been proposed to efficiently and accurately estimate the similarity. Yu et al. (2012) provided [27] a novel semisupervised multiview distance metric learning (SSM-DML), which learns the multiview distance metrics both from the multiple feature sets and from the labels of unlabeled cartoon characters, under the umbrella of graph-based semisupervised learning. The effectiveness of the SSM-DML has been proven in the cartoon applications.

In the same year, Yu et al. (2012) achieved [28] a novel method which learns novel distance through hypergraph for transductive image classification. Hypergraph learning has been projected to solve the following difficulties: the existing graph-based learning methods can only model the pairwise relationship of images and they are sensitive to the parameter used in similarity calculation. In their proposed method, the authors generate hyperedges by linking images and their nearest neighbors and can automatically modulate the effects of different hyperedges (by learning the labels of unlabeled images and the weights of hyperedges).

Later, Yu et al. (2014) proposed [29] a novel multiview stochastic distance metric learning method, which is crucial in utilizing the complementary information of multiview data. In comparison with the existing strategies, the proposed approach adopts the high-order distance obtained from the hypergraph to replace pairwise distance in estimating the probability matrix of data distribution.

Kwan and Cai, 1997 have introduced [8] a four layer unsupervised fuzzy neural network to solve pattern recognition problems, proving the importance of combining of strengths of fuzzy logic and neural networks. The fuzzy neurons of the third layer allow the computing of the similarities corresponding to an input pattern to all the learned patterns. In the case of the classical unsupervised FKCNN, a class is represented by a single output neuron.

Hariri et al. (2008) modified [7] the structure of the unsupervised fuzzy neural network of Kwan and Cai, composing a new version (improved five-layer feedforward Supervised Fuzzy Neural Network = SFN) and used it for classification and identification of shifted and distorted training patterns. To show the identification capability of the SFN, they used fingerprint pattern to prove that their testing result is more considerable than early FKCNN.

⁶Chen, F., Similarity Analysis of Video Sequences Using an Artificial Neural Network, University of Georgia, 2003, http://atheneum.libs.uga.edu/bitstream/handle/10724/6674/chen_feng_200305_ms.pdf?sequence=1.

We started this work with the paper [30], which presents a neuro-fuzzy approach to face recognition using an improved version of Kwan and Cai fuzzy neural network [8]. We have modified the above-mentioned fuzzy net from an unsupervised network into a supervised one, called *Fuzzy Kwan–Cai Supervised Net* and we have applied it for the special task of *Face Recognition*. The supervised FKCNN has more output neurons that each designate a class. Classification of an image with unknown class is achieved through the association of the input pattern with the class corresponding to that neuron in the fourth layer which has the output equal to 1.

2.3 Background

2.3.1 Measure of the Class Similarities

Let us first carefully examine the notion of similarity, before defining the methods for computing similarity. We shall use the notion of class similarities as it will be necessary later, in the algorithm of the evaluation criteria. Let us suppose we have M classes ω_i , $i = \overline{1, M}$, with mean vectors μ_i and inner variances S_i . We denote with D_{ij} the distance between the feature vectors of the classes ω_i , ω_j ; usually one chooses μ_i as a feature vector of the class ω_i . We have a measure $R(S_i, S_j, D_{ij})$ of class similarities [31] if for each $i, j = \overline{1, M}$ it defines the *similarity between the classes* ω_i and ω_j such that:

- (1) $R(S_i, S_j, D_{ij}) \geq 0$ (the similarity measure among the classes is greater than or equal to zero);
- (2) $R(S_i, S_j, D_{ij}) = R(S_j, S_i, D_{ji})$, namely the similarity measure between two pattern classes is necessarily symmetric;
- (3) $R(S_i, S_j, D_{ij}) = 0$ if and only if $S_i = S_j = 0$ (the similarity measure is null if and only if the inner variances are null);
- (4) If $S_j = S_k$ and $D_{ij} < D_{ik}$, then $R(S_i, S_j, D_{ij}) < R(S_i, S_k, D_{ik})$, namely the similarity measure decreases as a consequence of the fact that the inner variances are the same and the distance between the classes, i.e., between the feature vectors increases;
- (5) If $D_{ij} = D_{ik}$ and $S_j > S_k$, then $R(S_i, S_j, D_{ij}) > R(S_i, S_k, D_{ik})$, i.e., the similarity measure increases in the case when the distances among the classes are equal and the inner variances increase.

Table 2.1 The important notations of this subsection

The used notation	Its significance
M	Number of the classes ω_i , $i = \overline{1, M}$
$R(S_i, S_j, D_{ij})$	A measure of class similarities
μ_i	Mean of the class ω_i
S_i	Inner variances of the samples belonging to the class ω_i
D_{ij}	Minkowski distance of order p between μ_i and μ_j ; For $p = 2$ it means the Euclidean distance

An example of a similarity measure which we shall use throughout this paper is [31]:

$$R_{ij} = R(S_i, S_j, D_{ij}) = \frac{S_i + S_j}{D_{ij}}, \quad (\forall) i, j = \overline{1, M}. \quad (2.2)$$

$$\mu_i = \frac{1}{N_i} \sum_{X_j \in \omega_i} X_j, \quad (2.3)$$

$$S_i = \left(\frac{1}{N_i} \sum_{X_j \in \omega_i} \|X_j - \mu_i\|^q \right)^{1/q}, \quad (2.4)$$

$$D_{ij} = \left(\sum_{k=1}^n |\mu_{ik} - \mu_{jk}|^p \right)^{1/p}, \quad (2.5)$$

D_{ij} being the Minkowski distance of order p between μ_i and μ_j ; for $p = 2$ it will result the Euclidean distance. For $q = 2$, S_i represents the inner variances of the samples belonging to the class ω_i against the mean μ_i of that class. If $p = q = 2$ we get the similarity measure introduced by Fisher [31].

We shall provide a notation Table 2.1 to list the important notations of this subsection.

2.3.2 Similarity-Based Classification

“Similarity-based classifiers are defined as those classifiers that require only a pairwise similarity—a description of the samples themselves is not needed.”⁷

⁷Mellouk, A., and Chebira, A., Machine Learning, 2009, InTech.

A simple similarity-based classifier is the k-Nearest Neighbor classification rule (k-NN), classifying an image into the class of the most similar images in the database [32]. The probability of error for the k-NN rule is less than the probability of error corresponding to the NN. The NN rule is suboptimal, i.e., it does not converge to Bayes optimal rule, but it turns out [33] that the probability of error of NN rule, asymptotically never exceeds twice the Bayes rate. Therefore, the probability of error for NN rule has the lower bound and the upper bound expressed in the formula [33]:

$$P^* \leq P \leq P^* \left(2 - \frac{c}{c-1} P^* \right),$$

where c is the number of classes and P^* denote the error rate of Bayes rule.

Self-organizing Kohonen maps are not designed for similarity-based classification. This neural network has, however, proved to be very useful in a wide range of problems, and it is considered especially suitable for clustering large high-dimensional data sets such as images, documents [34, 35], etc. It is unsupervised, in that it does not need any categorical information about the input data during its training process. After the training process, the achieved map is organized in such a way that similar data from the input space are mapped to the same unit or to neighboring units on the map.

The Kohonen neural network is a network whose learning process is based on [31]:

- *the principle of competition*: for a vector applied to the network input, one determines the winner neuron in the network, as the neuron with the best match;
- *the neighborhood principle*: one refines both the weight vector associated with the winner neuron and the weights associated with the surrounding neurons.

Figure 2.1 proves that the SOKM transforms the similarities among the input vectors into some neighborhood relationships between the neurons; it can be noticed that three similar input vectors from the class 1 are assigned to three neighboring neurons, etc.

The SOKM is characterized in that the neighboring neurons:

- are correlated;
- turn into some specific detectors of different classes of patterns;
- characterize the vectors applied to the network input.

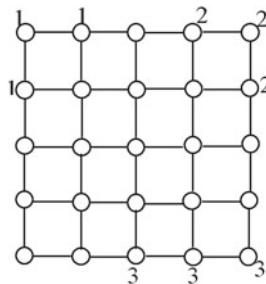


Fig. 2.1 Similarities among the input vectors

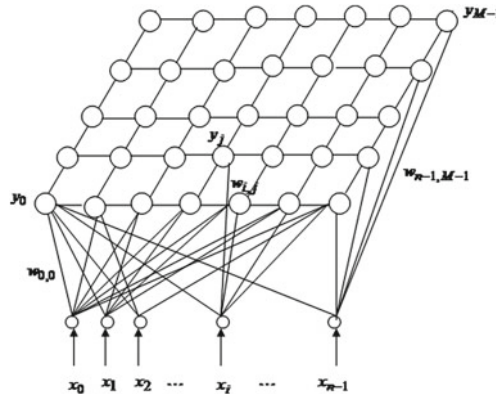


Fig. 2.2 Structure of a rectangular Kohonen network with M output neurons and n input neurons

The Kohonen network structure has two layers of neurons (see Fig. 2.2).

The neurons of the input layer correspond to the dimension of the input vector. The output layer contains M neurons, this number being greater than or equal to the number of clusters. These neurons are constrained to a regular grid, which usually is two-dimensional. All neurons in the input layer are connected to all neurons in the output layer. This neural network is self-organizing. The maps are trained in an unsupervised way using a competitive learning scheme.

The network training is one competitive, unsupervised, and self-organizing.

The training algorithm corresponding to the SOKM can be summarized below:

Step 1 Set $t = 0$. Initialize the learning rate $\eta(0)$ and the neighborhood of the neuron j , in the frame of the network, denoted V_j and establish the rule regarding their variations in time.

The weight values of the SOKM are initialized with some random values. Each neuron j of SOKM is characterized by a vector of weights $W_j = (w_{0j}, w_{1j}, \dots, w_{n-1j})^t \subset \mathfrak{R}^n, (\forall) j = \overline{0, M-1}, M$ being the number of neurons of SOKM and n representing the dimension of the input vectors.

Step 2 Apply the vector $X_p = (x_{p0}, x_{p1}, \dots, x_{p,n-1})^t \subset \mathfrak{R}^n, (\forall) p = \overline{1, N}$ (N being the number of the training vectors) at the network input.

Step 3 Compute the Euclidean distances between X_p and the weight vectors associated with the network neurons, at the time moment t , based on the formula:

$$d_j = \| X_p(t) - W_j(t) \|^2 = \sum_{i=0}^{n-1} (x_{pi}(t) - w_{ij}(t))^2, (\forall) j = \overline{0, M-1}. \quad (2.6)$$

Step 4 Find the winning neuron j^* of the SOKM, by computing the minimum distance among the vector X_p and the weight vectors W_j of the network, according to the relation [31]:

$$d_{j^*} = \min_{j=0}^{M-1} d_j, \quad (\forall) j = \overline{0, M-1}. \quad (2.7)$$

Step 5 Once the winning neuron is known, one refines the weights for the winning neuron and its neighbors, using the relation [31]:

$$W_{j+1}(t) = W_j(t) + \eta(t) \cdot (X_p(t) - W_j(t)), \quad j \in V_{j^*}, \quad (\forall) i = \overline{0, n-1}, \quad (2.8)$$

where

$$\eta(t) = \frac{\eta_0}{t^p} \exp\left(-\frac{\|r_k - r_{j^*}\|}{\sigma^2}\right) \quad (2.9)$$

is the learning rate and V_{j^*} represents the neighborhood of the neuron j^* .

In the relation (2.9) we have:

- $\eta_0 = \eta_0(t)$ and $\sigma = \sigma(t)$ is the value of the learning rate in the neighborhood center and respectively the parameter that controls the speed of the decrease of the learning rate depending on the neighborhood radius;
- r_{j^*} and r_k are the position vectors within the network of the central neuron of the neighborhood V_{j^*} and respectively of the neuron k for which we update the weights.

The neighborhood radius V_{j^*} can vary in time, being chosen to the beginning of refining to cover the whole network, after which it decreases monotonically in time.

Step 6 Set $p = p + 1$. If we finished the whole lot of the vectors, we have to check if the stop condition of the training process is satisfied, namely: after a fixed number of epochs or when we no longer get a significant change of the weights associated with the network neurons, according to [31]:

$$\|w_{ij}(t+1) - w_{ij}(t)\| < \varepsilon, \quad (\forall) i = \overline{0, n-1}, \quad (\forall) j = \overline{0, M-1}. \quad (2.10)$$

If the stop condition from (2.10) is not satisfied then we resume the training algorithm.

When we want to classify the lot of input vectors based on the similarities between the vectors, it will be made a training of the network until all the vectors belonging to a class will be associated to a single neuron of the network, which will represent that class.

The performances of SOKM depend on the network shape; it can be with the following architecture:

- (1) 1D: linear, circular;
- (2) 2D: planar (rectangular, hexagonal), spherical, toroidal;
- (3) 3D: spherical, toroidal, parallelepipedal.

As the similarity is not relevant only for the unsupervised clustering but for the supervised classification too, we shall turn the classical FKCNN, with four types of fuzzy neurons, into a supervised one.

When an input pattern is provided to the FKCNN input, this network first fuzzifies the respective pattern and then computes the similarities of this pattern to all of the learnt patterns.

In the case when a learnt previously pattern one presents at the FKCNN input, the network will treat the respective pattern as a known pattern, without to relearn it.

After computing the similarities, the FKCNN takes a decision by selecting the learnt pattern with the highest similarity and gives a nonfuzzy output.

2.3.3 Using Kohonen Algorithm in Vector Quantization

In the previous section, we have seen that the SOKM is a unsupervised method for learning similarity; SOKM also constitutes a nonlinear neural approach of feature selection (see Sect. 2.6.2). The aim of the present section is to describe how the SOKM can perform a Vector Quantization (VQ).

VQ has been noticed [36–39] as an efficient technique for image compression.

The VQ compression system consists of the two components [37, 40]:

- (1) VQ encoder;
- (2) VQ decoder.

New approaches based on neural networks as compression tools have been dealt by [41–43]. Particularly, the SOKM can be used as a vector quantizer for images [36–38].

We shall describe the algorithm which performs the VQ for the 256×256 images, using the SOKM [44]:

- Step 1* Split the original image into 1024 square blocks 8×8 and apply the DCT for each of them as in the case of the Algorithm 1.1.
- Step 2* Build a vector, whose components are the nine coefficients (10 retained without the first one) from each block, in a zigzag fashion (see the Algorithm 1.1) and cancel the rest of $(64 - 10)$ coefficients; in this way, it results the set of the nine-dimensional vectors X_i , $i = \overline{1, 1024}$, denoted $\aleph = \{X_1, \dots, X_{1024}\}$.
- Step 3* Design a SOKM with $32 \times 32 = 1024$ output neurons, which will be trained using the 1024 vectors achieved at the *Step 2*, that will be applicated one at a time, at the input network.
- Step 4* “Freeze” the weights resulted at the end of the learning algorithm of the SOKM.
- Step 5* Determine the prototype vectors for each neuron in the following way:
 - (a) apply again each vector X_i , $i = \overline{1, 1024}$ at the input of the SOKM to compute for its corresponding Euclidean distance to each neuron from the network with the formula (2.6);

- (b) find the minimum distance among the vector X_i , $i = \overline{1, 1024}$ and the weight vectors, that characterize each neuron of the SOKM, according to the relation (2.7);
- (c) design a statistics in order to determine how many vectors (each of them representing one block) from \aleph are associated to one neuron and then calculate the mean of these vectors to find the prototype vectors for each neuron.

Remark 2.1 Each class is characterized by a:

1. neuron j^* ;
2. prototype vector μ_{j^*} attached to this neuron;
3. 10-bit code word $C_{j^*} = (c_{1j^*}, \dots, c_{10j^*})$ (it corresponds to a network with 1024 output neurons); for example $C_{j^*} = \underbrace{(11 \dots 0)}_{10 \text{ times}}$

Step 6 Build the coded image by replacing (for each of the 1024 blocks) the first from the 10 retained coefficients with c_{1j^*} (the first bit of the code word, which represents the respective block).

Step 7 Apply the inverse DCT for each of the 1024 blocks to achieve the compressed image and convert its elements into integer values.

Step 8 Evaluate the performances of the VQ algorithm both from the visual point view and in terms of the PSNR, too.

2.3.4 Fourier Descriptors

The descriptors of different objects can be compared in order to give [45] a measurement of their similarity.

The Fourier descriptors have interesting properties in describing the shape of an object, by considering its boundaries.

Let γ be [31] a closed pattern, oriented counterclockwise, having the parametric representation: $z(l) = (x(l), y(l))$, where l is the length of an arc in a circle, along the curve γ , starting from an origin and $0 \leq l < L$, where L is the length of the boundary.

A point moving along the boundary generates the complex function $u(l) = x(l) + iy(l)$, which is a periodic function, having the period L .

Definition 2.1 The *Fourier descriptors* represents the coefficients corresponding to the decomposition of the function $u(l)$ in a complex Fourier series.

Similar to the implementation of a Fourier series to build a specific time signal consisting of cosine/sine waves of different amplitude and frequency, “the Fourier descriptor method uses a series of circles with different sizes and frequencies to build up at two-dimensional plot of a boundary”⁸ belonging to an object.

⁸Janse van Rensburg, F.J., and Treurnicht, J., and Fourie, C.J., The Use of Fourier Descriptors for Object Recognition in Robotic Assembly, 5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering, 2006.

The Fourier descriptors are computed using the formula [31, 46]:

$$a_n = \frac{1}{L} \int_0^L u(l) e^{-i \frac{2\pi}{L} nl} dl \tag{2.11}$$

such that

$$u(l) = \sum_{n=-\infty}^{\infty} a_n \cdot e^{i \frac{2\pi}{L} nl}. \tag{2.12}$$

We shall prove what becomes the formula (2.11) in the case of a polygonal contour, depicted in Fig. 2.3.

Denoting

$$\lambda = \frac{\|\overline{V_{k-1}M}\|}{\|MV_k\|}, \tag{2.13}$$

namely (see Fig. 2.4):

$$\lambda = \frac{l - l_{k-1}}{l_k - l}, \tag{2.14}$$

where $M(x_M, y_M)$, $V_{k-1}(x_{k-1}, y_{k-1})$, $V_k(x_k, y_k)$ and

$$\begin{cases} l_0 = 0, \\ l_k = \sum_{i=1}^k |V_i - V_{i-1}|; \end{cases} \tag{2.15}$$

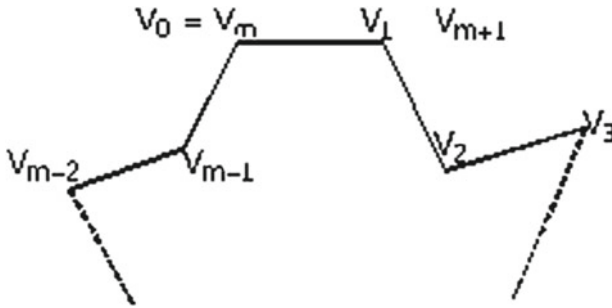


Fig. 2.3 Polygonal contour

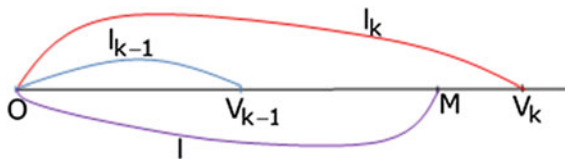


Fig. 2.4 Parametric representation of a segment

from (2.11) it results:

$$a_n = \frac{1}{L} \sum_{k=1}^m \int_{l_{k-1}}^{l_k} (x_M(l) + iy_M(l)) \cdot e^{-i\frac{2\pi}{L}nl} dl. \tag{2.16}$$

From (2.13) we deduce:

$$\begin{cases} x_M = \frac{x_{k-1} + \lambda x_k}{1 + \lambda}, \\ y_M = \frac{y_{k-1} + \lambda y_k}{1 + \lambda}. \end{cases} \tag{2.17}$$

We shall treat each coordinate pair as a complex number so that in Fig. 2.5:

Taking into account the previous assumption and the relations (2.17) and (2.14) we can notice that

$$\begin{aligned} x_M(l) + iy_M(l) &= \frac{x_{k-1} + \frac{l-l_{k-1}}{l_k-l} \cdot x_k}{1 + \frac{l-l_{k-1}}{l_k-l}} + i \frac{y_{k-1} + \frac{l-l_{k-1}}{l_k-l} \cdot y_k}{1 + \frac{l-l_{k-1}}{l_k-l}} \\ &= \frac{(x_{k-1} + iy_{k-1})l_k + l[(x_k - x_{k-1}) + i(y_k - y_{k-1})] - (x_k + iy_k)l_{k-1}}{l_k - l_{k-1}} \\ &= \frac{V_{k-1}l_k + l(V_k - V_{k-1}) - V_k l_{k-1}}{l_k - l_{k-1}}; \end{aligned}$$

hence, the formula (2.16) will become

$$\begin{aligned} a_n &= \frac{1}{L} \sum_{k=1}^m \frac{1}{l_k - l_{k-1}} \int_{l_{k-1}}^{l_k} [V_{k-1}l_k + l(V_k - V_{k-1}) - V_k l_{k-1}] \cdot e^{-i\frac{2\pi}{L}nl} dl \\ &= \frac{1}{L} \sum_{k=1}^m \frac{1}{l_k - l_{k-1}} \int_{l_{k-1}}^{l_k} [V_{k-1}l_k + l(V_k - V_{k-1}) - V_k l_{k-1}] \cdot e^{-i\frac{2\pi}{L}nl} dl = \end{aligned}$$

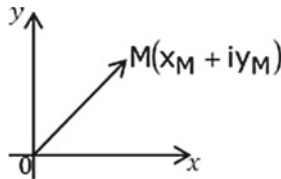


Fig. 2.5 Treating a point M as a complex number

$$a_n = \frac{1}{L} \sum_{k=1}^m \frac{1}{l_k - l_{k-1}} \left[(V_{k-1}l_k - V_k l_{k-1}) \cdot \int_{l_{k-1}}^{l_k} e^{-i\frac{2\pi}{L}nl} dl + (V_k - V_{k-1}) \cdot \underbrace{\int_{l_{k-1}}^{l_k} l \cdot e^{-i\frac{2\pi}{L}nl} dl}_I \right]. \quad (2.18)$$

By computing

$$\begin{aligned} I &= \int_{l_{k-1}}^{l_k} l \cdot e^{-i\frac{2\pi}{L}nl} dl = -\frac{1}{i\frac{2\pi}{L}n} \left(l_k \cdot e^{-i\frac{2\pi}{L}nl_k} - l_{k-1} \cdot e^{-i\frac{2\pi}{L}nl_{k-1}} \right) \\ &\quad + \frac{1}{\left(\frac{2\pi}{L}n\right)^2} \left(e^{-i\frac{2\pi}{L}nl_k} - e^{-i\frac{2\pi}{L}nl_{k-1}} \right) \end{aligned}$$

and substituting it into (2.18) we shall achieve

$$a_n = \frac{1}{L} \sum_{k=1}^m \frac{1}{l_k - l_{k-1}} \cdot T_k, \quad (2.19)$$

where

$$\begin{aligned} T_k &= -\frac{1}{i\frac{2\pi}{L}n} (V_{k-1}l_k - V_k l_{k-1}) \cdot \left(e^{-i\frac{2\pi}{L}nl_k} - e^{-i\frac{2\pi}{L}nl_{k-1}} \right) \\ &\quad - \frac{1}{i\frac{2\pi}{L}n} (V_k - V_{k-1}) \cdot \left(l_k e^{-i\frac{2\pi}{L}nl_k} - l_{k-1} e^{-i\frac{2\pi}{L}nl_{k-1}} \right) \\ &\quad + \frac{1}{\left(\frac{2\pi}{L}n\right)^2} (V_k - V_{k-1}) \cdot \left(e^{-i\frac{2\pi}{L}nl_k} - e^{-i\frac{2\pi}{L}nl_{k-1}} \right), \end{aligned}$$

namely

$$\begin{aligned} T_k &= -\frac{1}{i\frac{2\pi}{L}n} \left[V_k(l_k - l_{k-1}) \cdot e^{-i\frac{2\pi}{L}nl_k} - V_{k-1}(l_k - l_{k-1}) \cdot e^{-i\frac{2\pi}{L}nl_{k-1}} \right] \\ &\quad + \frac{1}{\left(\frac{2\pi}{L}n\right)^2} (V_k - V_{k-1}) \cdot \left(e^{-i\frac{2\pi}{L}nl_k} - e^{-i\frac{2\pi}{L}nl_{k-1}} \right). \quad (2.20) \end{aligned}$$

Therefore, on the basis of the relations (2.20) and (2.15) from which it results

$$l_k - l_{k-1} = |V_k - V_{k-1}|, \quad (\forall) k = \overline{1, m},$$

the formula (2.19), which allows us to compute the Fourier descriptors will be:

$$\begin{aligned}
 a_n = & -\frac{1}{2\pi in} \sum_{k=1}^m \left(V_k \cdot e^{-i\frac{2\pi}{L}nl_k} - V_{k-1} \cdot e^{-i\frac{2\pi}{L}nl_{k-1}} \right) \\
 & + \frac{L}{(2\pi n)^2} \sum_{k=1}^m \frac{V_k - V_{k-1}}{|V_k - V_{k-1}|} \cdot \left(e^{-i\frac{2\pi}{L}nl_k} - e^{-i\frac{2\pi}{L}nl_{k-1}} \right). \tag{2.21}
 \end{aligned}$$

The main advantage of using the Fourier Descriptor method for object recognition consists in the fact that the Fourier descriptors are [45] invariant to translation, rotation, and scale.

2.3.5 Fuzzy Neurons

First, we shall define a Fuzzy Neuron (FN) in general and then the four models of fuzzy neurons, that are the components of the fuzzy neural network, described in this chapter.

A FN (illustrated in Fig. 2.6) has [8]:

- N weighted inputs $x_i, i = \overline{1, N}$
- N weights $w_i, i = \overline{1, N}$
- M outputs $y_j, j = \overline{1, M}$.

Each output can be associated with a membership value to a fuzzy concept, in the meaning that it expresses to what degree, the pattern with the inputs $\{x_1, x_2, \dots, x_N\}$ belongs to a fuzzy set.

The equations that characterize a FN are [8]:

$$z = h(w_1x_1, w_2x_2, \dots, w_Nx_N), \tag{2.22}$$

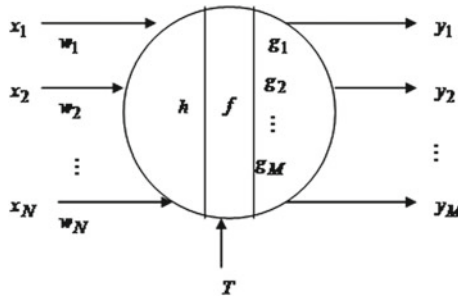


Fig. 2.6 A FN

$$s = f(z - T), \quad (2.23)$$

$$y_j = g_j(s), \quad j = \overline{1, M}, \quad (2.24)$$

where

- z is the input net of the FN,
- h constitutes the aggregation function,
- s means the state of the FN,
- f is the activation function,
- T is the activating threshold,
- $g_j, j = \overline{1, M}$ are the M outputs of the FN, that represents the membership functions of the input pattern $\{x_1, x_2, \dots, x_N\}$ to the M fuzzy sets.

Therefore, the FN can express and process the fuzzy information.

In general, the weights, the activating threshold, and the output functions that describe the interactions among the fuzzy neurons could be adjusted during the learning procedure. Hence, the fuzzy neurons are adaptive and a Fuzzy Neural Network (FNN) which has such neurons can learn from the environment.

The activation function and the activating threshold are the intrinsic features of a FN.

If different functions of f and h are used for different fuzzy neurons, then their properties will be different. Choosing different functions f and h we can achieve many types of fuzzy neurons.

The following four types of fuzzy neurons will be defined [8]:

- (A) **Input-FN** is a FN, which is used in the input layer of a FNN and that has only one input x such that

$$z = x; \quad (2.25)$$

- (B) **Maximum-FN (Max-FN)** is that FN, (represented in Fig. 2.7), which has a maximum function as a aggregation function, such that

$$z = \max_{i=1}^N(w_i x_i); \quad (2.26)$$

- (C) **Minimum-FN (Min-FN)** is that FN (represented in Fig. 2.8), which has a minimum function as a aggregation function, such that

$$z = \min_{i=1}^N(w_i x_i); \quad (2.27)$$

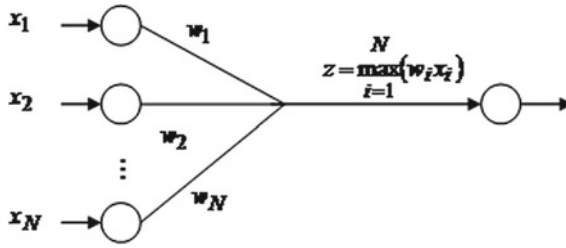


Fig. 2.7 A Max-FN

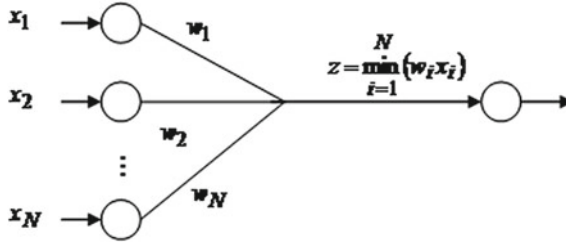


Fig. 2.8 A Min-FN

(D) **Competitive-FN (Comp-FN)** is a FN which has a variable threshold T and only one output, such that

$$y = g(s - T) = \begin{cases} 0 & \text{if } s < T, \\ 1 & \text{if } s \geq T. \end{cases} \tag{2.28}$$

$$T = t(c_1, \dots, c_k), \tag{2.29}$$

where

- s is the state of the FN;
- t is a threshold function;
- $c_k, k = \overline{1, K}$ are the competitive variables of the FN.

2.4 Fuzzy Kwan–Cai Neural Network

In this study, we aim to provide an effective method for learning image similarity. To reach this aim we start from the Fuzzy Kwan–Cai Neural Network (FKCNN) and turn it into a supervised method for learning similarity. The new approach gives a better performance than its unsupervised counterparts [8] as in the case of the classical unsupervised FKCNN, a class is represented by a single-output neuron, while the supervised FKCNN has more output neurons that each designate a class. This concept is similar with the idea of replacing the binary decision about the

membership (nonmembership) of a pattern to a class, with the introduction of a membership degree, between 0 and 1.

Using the following four types of fuzzy neurons [8]: **Input-FN**, **Maximum-FN (Max-FN)**, **Minimum-FN (Min-FN)**, **Competitive-FN (Comp-FN)** as basis we can develop our Fuzzy Kwan–Cai neural network [8] for learning similarity.

2.4.1 Architecture of FKCNN

The FKCNN is a feedforward fuzzy neural network with four layers each containing a specific type of neuron, as shown in Fig. 2.9. This is a parallel system designed both for image recognition [30] and for the Information Retrieval: it simultaneously processes all the pixels of an input image, namely it is a parallel system. Its each layer is built by a specific type of fuzzy neurons.

The first layer of the network is the input layer, which ensures the provision of patterns, that the network has to recognize. This layer contains Input-FNs and each neuron corresponds to a feature of the input pattern. The Input-FNs are arranged in a 2D structure and their number is equal to the number of features that belong to the input pattern. Assuming that each pattern has $N_1 \times N_2$ features, it results that the first layer has $N_1 \times N_2$ neurons. The algorithm of the (i, j) -th Input-FN in the first layer is

$$s_{ij}^1 = z_{ij}^1 = x_{ij}, \quad (\forall i = \overline{1, N_1}, j = \overline{1, N_2}), \quad (2.30)$$

$$y_{ij}^1 = \frac{s_{ij}^1}{P_{vmax}}, \quad (\forall i = \overline{1, N_1}, j = \overline{1, N_2}), \quad (2.31)$$

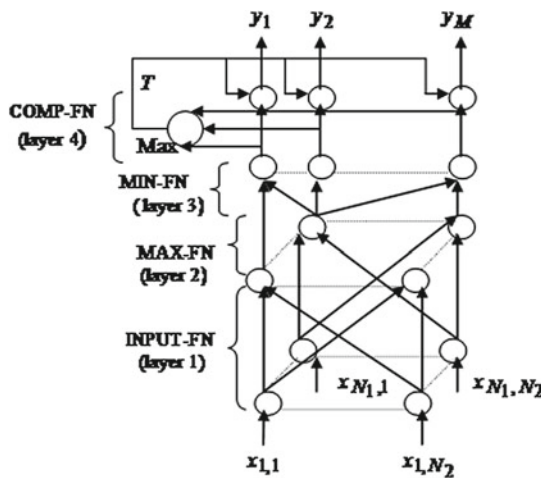


Fig. 2.9 Architecture of FKCNN

where x_{ij} is the (i, j) -th feature of an input pattern ($x_{ij} \geq 0$) and P_{vmax} is the maximum feature among all input patterns.

From Eq. (2.31) we can notice that the first layer receives image data and transforms the features corresponding to the input image into normalized values within the interval $[0, 1]$.

The second layer is also organized as a 2D grid and consists of $N_1 \times N_2$ Max-FNs. The purpose of this layer is to fuzzy the input patterns through a weight function $w(m, n)$.

The state of the (p, q) -th Max-FN in this layer is given [8] by

$$s_{pq}^{[2]} = \max_{i=1}^{N_1} \left(\max_{j=1}^{N_2} (w(p-i, q-j)y_{ij}^1) \right), \quad (\forall) p = \overline{1, N_1}, q = \overline{1, N_2}, \quad (2.32)$$

where $w(p-i, q-j)$ is the weight connecting the (i, j) -th Input-FN in the first layer to the (p, q) -th Max-FN of the second layer.

The weight function has [8] the following analytical expression:

$$w(m, n) = e^{-\beta^2(m^2+n^2)}, \quad (\forall) m = \overline{-(N_1-1), (N_1-1)}, n = \overline{-(N_2-1), (N_2-1)}. \quad (2.33)$$

Using the fuzzification weight function given by (2.33), “each Max-FN in the second layer behaves like a lens focusing on a specific feature of the input pattern.”⁹

It can keep into account different neighbor features of the central one, too. The parameter β controls the number of features seen by a Max-FN and its value is determined by the learning algorithm.

Each FN in the second layer has M different outputs, one for each FN of the third layer. The outputs of the (p, q) -th Max-FN in the second layer are given [8] by the relation:

$$y_{pqm}^{[2]} = g_{pqm}(s_{pq}^{[2]}), \quad (\forall) p = \overline{1, N_1}, q = \overline{1, N_2}, m = \overline{1, M}, \quad (2.34)$$

where $y_{pqm}^{[2]}$ is the m -th output of the (p, q) -th Max-FN in the second layer, that is connected to the m -th Min-FN in the third layer.

The output function $g_{pqm}(s_{pq}^{[2]})$ will be determined through a learning algorithm. For simplicity, the fuzzy output functions of neurons in the second layer will be commonly chosen as isosceles triangles, having the height equal to 1 and the base length equal to α (see Fig. 2.10).

⁹Kwan, H. K. and Cai, Y., A Fuzzy Neural Network and its Application to Pattern Recognition, IEEE Trans. on Fuzzy Systems, 1997, 2(3), 185–193.

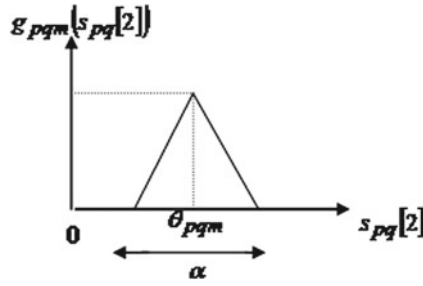


Fig. 2.10 The output function of a Max-FN

Then [8]:

$$y_{pqm}^{[2]} = g_{pqm}(s_{pq}^{[2]}) = \begin{cases} 1 - 2|s_{pq}^{[2]} - \theta_{pqm}|/\alpha & \text{if } \alpha/2 \geq |s_{pq}^{[2]} - \theta_{pqm}| \geq 0, \\ 0 & \text{otherwise,} \end{cases} \quad (2.35)$$

(\forall) $\alpha \geq 0$, $p = \overline{1, N_1}$, $q = \overline{1, N_2}$, $m = \overline{1, M}$, θ_{pqm} being the central point of the base of function $g_{pqm}(s_{pq}^{[2]})$.

The third layer contains Min-FNs; as each of them represents a learnt pattern, the number M of the Min-FNs in the third layer could be determined only after the learning procedure is finished. The output of the m -th Min-FN in the third layer is given [8] by the relation:

$$y_m^{[3]} = s_m^{[3]} = \min_{p=1}^{N_1} \left(\min_{q=1}^{N_2} (y_{pqm}^{[2]}) \right), \quad (\forall) m = \overline{1, M}, \quad (2.36)$$

where $s_m^{[3]}$ represents the state of the m -th Min-FN, in the third layer.

The fourth layer is the output layer and it contains M Comp-FNs, one for each of the M learnt patterns. It provides nonfuzzy (crisp) outputs. If an input image is most similar to the m -th learnt pattern, then the output of the m -th Comp-FN becomes 1, while the other outputs are equal to 0. The relations that characterize the Comp-FN in the fourth layer are [8]:

$$s_m^{[4]} = z_m^{[4]} = y_m^{[3]}, \quad (\forall) m = \overline{1, M}, \quad (2.37)$$

$$y_m^{[4]} = g(s_m^{[4]} - T) = \begin{cases} 0, & \text{if } s_m^{[4]} < T \\ 1, & \text{if } s_m^{[4]} = T, \end{cases} \quad (2.38)$$

(\forall) $m = \overline{1, M}$. The activation threshold T of all the Comp-FNs in the fourth layer is

$$T = \max_{m=1}^M (y_m^{[3]}), \quad (\forall) m = \overline{1, M}. \quad (2.39)$$

2.4.2 Training Algorithm of FKCNN

In the learning phase, the following parameters will be determined by the learning procedure:

- the parameters of the output functions of the Max-FNs in the second layer, namely θ_{pqm} , (\forall) $p = \overline{1, N_1}$, $q = \overline{1, N_2}$, $m = \overline{1, M}$,
- the number of neurons for the third and the fourth layers.

To do so, we define a parameter T_f , $0 \leq T_f \leq 1$ that represents the fault tolerance of the FKCNN. Let K be the total number of training patterns. For $k = \overline{1, K}$, the steps of the learning procedure are:

- Step 1* Create $N_1 \times N_2$ Input-FNs in the first layer of the FKCNN and $N_1 \times N_2$ Max-FNs in the second layer. Choose a value for α , $\alpha \geq 0$ and a value for β .
- Step 2* Set $M = 0$ and $k = 1$.
- Step 3* Set $M = M + 1$. Create the M -th Min-FN in the third layer and the M -th Comp-FN in the fourth layer. Determine the central point corresponding to the M output functions for the (p, q) -th Max-FN in the second layer, denoted θ_{pqM} , using the formula:

$$\theta_{pqM} = s_{pqM}^{[2]} = \max_{i=1}^{N_1} \left(\max_{j=1}^{N_2} (w(p-i, q-j)x_{ijk}) \right), \quad (\forall) p = \overline{1, N_1}, \quad q = \overline{1, N_2}, \quad (2.40)$$

- $X_k = \{x_{ijk}\}$ being the k -th training pattern.
- Step 4* Set $k = k + 1$. If $k > K$ the learning algorithm finishes. Otherwise, introduce the k -th training pattern to network input and compute the outputs of the fourth layer of the FKCNN (with M Min-FNs in the third layer and M Comp-FNs in the fourth layer). Set

$$\sigma = 1 - \max_{j=1}^M y_{jk}^{[3]}, \quad (2.41)$$

where $y_{jk}^{[3]}$ is the output of the j -th Min-FN in the third layer, for the k -th training pattern X_k .

If $\sigma \leq T_f$ then go to *Step 4*. If $\sigma > T_f$ then go to *Step 3*.

In one training epoch, all patterns in the training set are sequentially given as input to the FKCNN.

The procedure of transforming the unsupervised fuzzy neural network into a supervised one is similar to that used in several applications of the Self-Organizing Map (Kohonen) [47, 48].

After the unsupervised learning algorithm, one calibrates [30] the FKCNN to learn similarity through partial supervision. For example, if we have a training set consisting of K images, belonging to P classes, one assumes that after training, the number of FNs of the third and fourth levels is M .

Then, after calibration, each of the M outputs corresponding to the Comp-FNs has a label, representing one of the P classes (generally $K \geq M \geq P$). The FNs of the fourth layer are then labeled as follows: neurons whose output is equal to 1 are labeled with the class corresponding to the corresponding input pattern.

Classification of an image with unknown class is achieved through the association of the input pattern with the class corresponding to that neuron in the fourth layer which has the output equal to 1.

2.4.3 Analysis of FKCNN

The first level of the FKCNN has to input, “the feature glows of the respective pattern and transmits these signals to the second layer, into normalized values, from the interval $[0, 1]$.”¹⁰ The aim of the second layer is to fuzzify the input pattern. Each Max-FN in the second layer is connected to all the Input-FNs of the first layer through the weight function $w(m, n)$ and the state of a fuzzy Max-FN is given by the maximum of the weighted inputs.

The second layer of the FKCNN performs the fuzzification of the lower level features belonging to the input pattern. The degree of fuzzification of the input pattern through the second layer depends on the parameter β . The smaller β is, the more Max-FN in the second layer are affected by a lower level feature of the input pattern. If β is too small, then FKCNN can not separate some distinct training patterns, while if the β is too large the network loses its ability to recognize some displaced or distorted patterns. The values of the parameter β should be chosen such that the FKCNN can separate all the distinct training patterns and moreover the network has to have an acceptable recognition rate.

Each Max-FN from the second layer has M different outputs; therefore M will define the number of the FNs in the third layer. The m -th output of the (p, q) -th Max-FN neuron in the second layer, i.e., $y_{pqm}^{[2]}$ “expresses to what extent the fuzzy concept about that zone the component values around the (p, q) -th component of the input pattern are similar to the component values around the (p, q) -th component of the m learnt pattern.” (see footnote 10).

The output function $g_{pqm}(s_{pq}^{[2]})$ is a membership function of this fuzzy set and it contains some data with regard to the component values around the (p, q) -th component

¹⁰Kwan, H. K. and Cai, Y., A Fuzzy Neural Network and its Application to Pattern Recognition, IEEE Trans. on Fuzzy Systems, 1997, 2(3), 185–193.

of the m pattern, which it has already learnt. The training algorithm uses θ_{pqM} at *Step 4*, to store such data; hence FKCNN can remember all learnt forms.

The fuzzy neurons of the third layer allows to compute the similarities of the input pattern with all the learnt patterns. As in the third layer one uses Min-FNs, the similarity of the input pattern $X = \{x_{ijk}\}$ to the m -th learnt pattern can be computed using [8] the formula:

$$y_m^{[3]} = \begin{cases} \min_{p,q} (1 - 2|s_{pq}^{[2]} - \theta_{pqm}|/\alpha) & \text{if } \max_{p,q} (|s_{pq}^{[2]} - \theta_{pqm}|) \leq \alpha/2, \\ 0 & \text{otherwise,} \end{cases} \quad (2.42)$$

(\forall) $m = \overline{1, M}$ and $s_{pq}^{[2]}$ is the state of the (p, q) -th Max-FN in the second layer, when $X = \{x_{ijk}\}$ are given as network input. The relation (2.42) shows that α is a visibility parameter (scope parameter) and its value affects the computation of the similarities. When the input pattern X is one of the previously learnt patterns, then one of the similarities $y_m^{[3]}$, $m = \overline{1, M}$ will be equal to 1. In the case when the input pattern X is not any of the learnt patterns then all the M similarities will be less than 1.

The output layer of the FKCNN is used to produce the defuzzification and to give some nonfuzzy outputs. The maximum similarity will be chosen as an activation threshold of all the Comp-FNs in the fourth layer. If $y_m^{[3]}$ has the maximum value among all the outputs of the FNs in the third layer, then the output of m -th Comp-FN in the fourth layer is equal to 1 and the outputs of the other Comp-FNs in this layer will be equal to 0.

The learning algorithm is developed in four stages: input data (layer 1), fuzzification (layer 2), fuzzy deduction (layer 3), and defuzzification (layer 4).

When a previously learned pattern is given as input, the network will treat the respective pattern as a known pattern, without relearning it. The way of treating an input pattern (either distinct from all patterns or a learned one) is determined both by the similarities of the FKCNN, computed to all the learnt patterns and the parameters α , β and T_f : α and β affect the computing of the similarities and T_f is the fault tolerance of the FKCNN. If one of the computed similarities to the input pattern is greater than or equal to $1 - T_f$, then this pattern will be treated as a learnt pattern; otherwise the pattern is treated as a new form.

2.5 Experimental Evaluation

2.5.1 Data Sets

We shall use the images from the PASCAL data set to evaluate our methods. The PASCAL Visual Object Classes (VOC) challenge is [9] a benchmark in visual object category recognition and detection, providing the vision and machine learning

communities with a standard data set of images and annotation. In our work we use 10102 images half of them (chosen randomly) for training and the other half for testing. The VOC data sets contain significant variability in terms of object size, orientation, pose, illumination, position, and occlusion. It consists of 20 object classes: aeroplane, bicycle, bird, boat, bottle, bus, car, cat chair, cow, dog, horse, motorbike, person, sheep, sofa, table, potted plant, train, and tv/monitor. The ColorDescriptor engine [49] was used to extract the image descriptors from all the images. To evaluate our methods we use images from the PASCAL data set. The PASCAL Visual Object Classes (VOC) challenge is [9] a benchmark in visual object category recognition and detection, providing the vision and machine learning communities with a standard data set of images and annotation. In our work, we use 10102 images half of them for training and the other half for testing. Figure 2.11 shows 50 images from the VOC data base.

In our work [1], we have used 64 descriptors for each image from the training and test set. Hence, the number of the neurons of the first layer corresponding to FKCNN is 8×8 and 64 in the case of SOKM.

2.5.2 Evaluation Criteria

We aim to compare the performance of the three methods we have defined k-NN, SOKM, and FKCNN in terms of their ability to learn similarity. Following (2.2) we need [1] the following stages:

Stage 1 (compute the similarities among the patterns). For each test image T_k , ($\forall k = \overline{1, 5051}$), classified in the class C'_i , ($\forall i = \overline{1, M}$) we compute its similarities S_{kp} ($\forall k, p = \overline{1, 5051}$) regarding the training images I_1, \dots, I_{5051} , that belong to the classes C_1, \dots, C_M .

Stage 2 (find the similarities among the classes). Determine with (2.2) the similarity R_{ij} between the classes C'_i and C_j (corresponding to the test image T_k and respectively associated to a training image I_p ($\forall p = \overline{1, 5051}$)).

Stage 3 (evaluation criteria). Check if for $S_{k,p_j} \leq S_{k,p_l}$ we have $R_{ij} \leq R_{il}$ or when $S_{k,p_j} > S_{k,p_l}$ we have $R_{ij} > R_{il}$, i being the class where the test image T_k belongs and j respectively l being the classes corresponding to the two training images I_{p_j} and I_{p_l} .

To determine how well the methods perform we have to [1]:

- apply the procedure for all $N = 5051 \cdot (5051 - 1)$ pairs of images;
- find the percentage for which the similarities were correctly computed, using the formula:

$$p [\%] = \frac{n}{N} \cdot 100,$$

n being the number of correctly evaluated images in the view of their similarities.

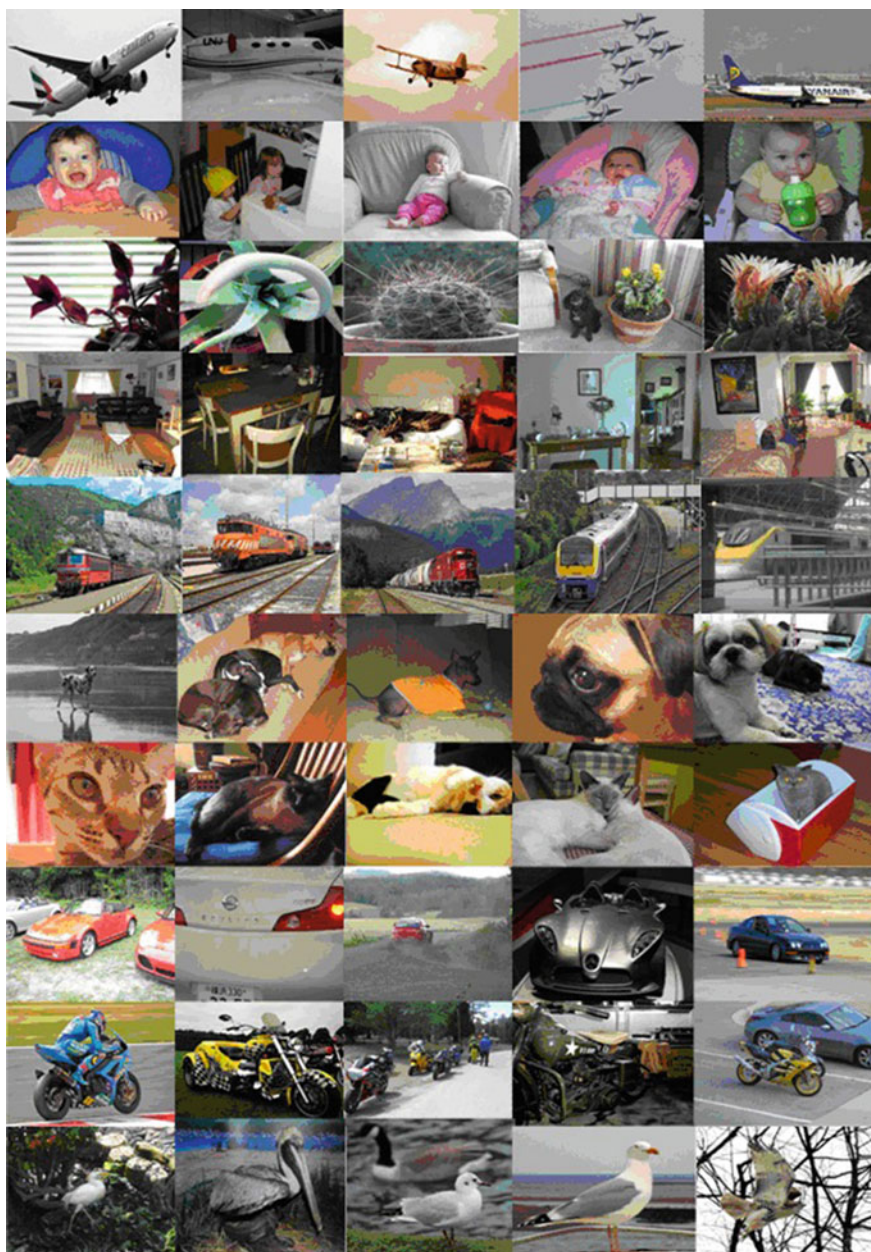


Fig. 2.11 50 images from the VOC data base

2.5.3 Experimental Results

For our experiments we set $\alpha = 1$, $\beta = 1$ as the parameters of the FKCNN. With different values of the threshold T_f we obtain the results in Table 2.2 and Fig. 2.12:

Figure 2.12 shows that the best value of M (corresponding to the number of neurons from the third and fourth layers of FKCNN) is 22 which is close to the number of classes in the data set which is 20.

In the case of using SOKM (trained during some epochs) we evaluate the performance of the algorithm for computing similarities in the following way:

- (1) find the coordinates of the winning neuron;
- (2) compute the similarity between a test image and a training image based on the formula (2.6), where W_j is the weight vector of that neuron associated to the respective training image;

Table 2.2 The overall performance of FKCNN: the values of p [%] obtained using the FKCNN, for different values of T_f , when $\alpha = 1$, $\beta = 1$

T_f	M	p [%]
0.41	16	99.8
0.396	19	99.7
0.39	20	99.8
0.38	21	99.8
0.37	22	99.9
0.36	23	99.8
0.35	24	99.8
0.3212	30	99.7

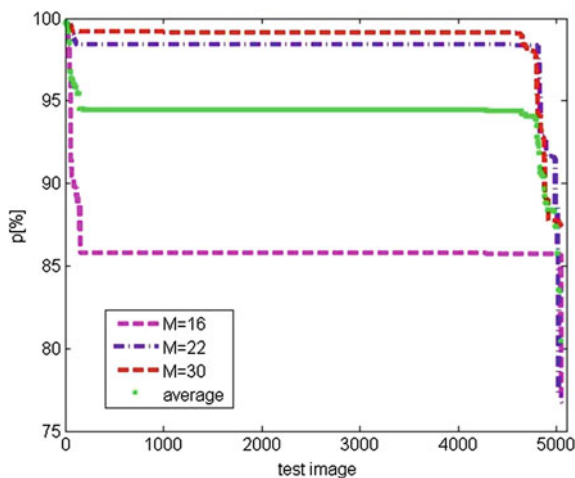


Fig. 2.12 The percentages for which the similarities were correctly computed using FKCNN

- (3) check if we achieve a smaller similarity for a training image that is closer to that test image.

We want to determine the robustness with respect to choosing the right parameters.

Table 2.3 and respectively Table 2.4, Figs. 2.13 and 2.14 illustrate the percentages for which the similarities were correctly computed using SOKM, as function of number of epochs used in training epochs and respectively by changing the weights.

Although, sometimes we can achieve good results using SOKM, it has an unstable behavior as Figs. 2.13 and 2.14 show. We note that the instability of SOKM is determined by the following two aspects: by training the SOKM in five epochs, the value of p [%] changes, especially in the cases when the weights increase; in these cases, p [%] will have smaller values than the values achieved when the weights decrease; in the situations where the SOKM is trained with a different number of epochs, the values of p [%] will fluctuate from one epoch to another.

Figure 2.15 and Table 2.5 indicate that for our task FNNKC and SOKM perform better than the nonneural method k-NN, with 22 neighbors.

We observe that:

- the nonneural k-NN method computes more wrong similarities than the neural methods;
- using the FKCNN we achieved the biggest number of correct similarities (i.e., 5038) between a test image and the training images compared to the other two methods;
- FKCNN is also most performant as the biggest sum of the number of correct similarities is achieved by it (namely, for each test image we achieved a big number

Table 2.3 The percentages for which the similarities were correctly computed using SOKM, after some training epochs

$Nrep$	p [%]
1	77
2	92.2
5	98.4
10	75.3
15	75.4
25	70.5

Table 2.4 The percentages for which the similarities were correctly computed using SOKM, by changing the weights

Interval where the weight belongs	p [%]
$[-0.002, 0.002]$	97.9
$[-0.2, 0.2]$	72.2
$[-0.05, 0.05]$	73
$[-0.006, 0.006]$	94.9
$[-0.09, 0.09]$	73.6

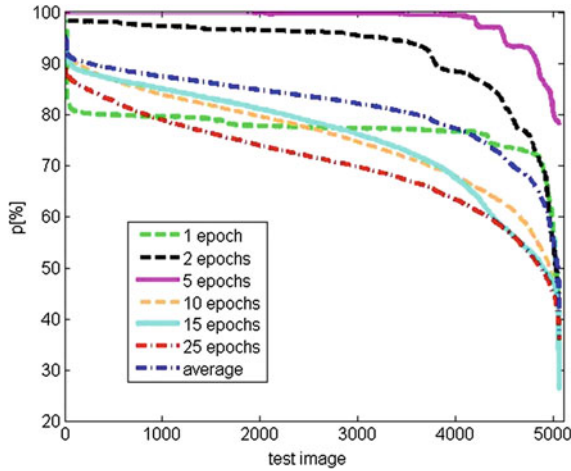


Fig. 2.13 The percentages for which the similarities were correctly computed using SOKM, after some training epochs

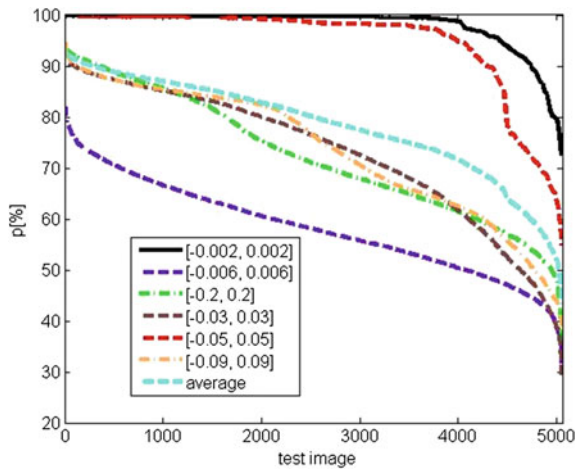


Fig. 2.14 The percentages for which the similarities were correctly computed using SOKM, by changing the weights

- of correct similarities with FKCNN, while using SOKM we can not have almost the same number of correct similarities for each image);
- the graph corresponding to the k-NN method is an almost constant function, while the functions of the neural methods have a descending behavior.

In Fig. 2.16 we can notice that the FKCNN is always better than SOKM as we have obtained the maximum value of $p = 99.86\%$ using FKCNN (see Table 2.2). The SOKM has better performance for the easy elements, the ones that are in the

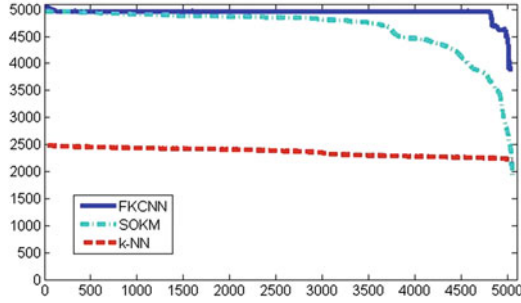


Fig. 2.15 The overall performance of the three proposed methods for computing similarities

Table 2.5 The percentage of those test images for which works well our procedure (for $M = 20$)

Used model	p [%]
FKCNN	99.75
k-NN	72.2
Euclidean distance (Cosine similarity)	50.12 (50.14)
SOKM	98.80

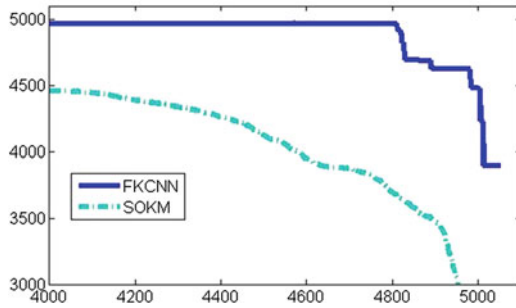


Fig. 2.16 Close-up of the overall performance of Fig. 2.15 of the neural methods for computing similarities

beginning of the list. FCKNN does a better job at the tail where the more difficult and likely more important elements in terms of tasks like clustering/classification lie.

The results indicate that the neural methods FKCNN and SOKM are performing better for our task than k -NN (see Fig. 2.15). SOKM sometimes gives good results, but this depends highly on the right parameter settings. Small variations induced large drops in performance. The overall performance of FKCNN is better (see Fig. 2.16). The main advantage of FKCNN consists in the fact that we can obtain good results that are robust to changes in the parameter settings.

2.6 Face Recognition

Face Recognition (FR) has been studied for many years and has practical and/or potential applications in areas such as security systems, criminal identification, video telephony, medicine, and so on. In comparison to other identification techniques, FR has the advantage of being nonintrusive and requiring very little cooperation; it has become a hot topic recently as better hardware and better software have become available.

On the other side, the hybrid systems of *fuzzy logic* and *neural networks* (often referred as *fuzzy neural networks*) represent exciting models of computational intelligence with direct applications in pattern recognition, approximation, and control.

Face Recognition represents a segment of a larger technological areas called Biometric Technology (BT). The BT has modern practical applications in the following fields: Medicine, Justice and Police, Finance and Banks, Border control, Voice and Visual communications, Access control for sensitive areas, as indicated [50] in Fig. 2.17.

BT is widely used for the applications that include: face recognition, voice recognition, signature recognition, hand geometry, iris, Automated Fingerprint Identification System (AFIS), and non-AFIS.

As an industry, the biometrics is in its early stages; according to a study of the *Grand View Research* (a market research and consulting company),¹¹ the revenues for this industry have a significant growth, such that it is expected to reach USD 24.59 billion by 2020 (see Fig. 2.18).

FR is an automated approach to identify humans through the unique features of their faces. It needs

- (A) a camera to take a digital image of a subject's face;
- (B) some algorithms to extract the facial features from the respective image, resulting a template. Various approaches have been proposed [51] to select the facial features from an image:
 - (a) *Geometry based Technique*-the selection is based on the size and the relative position of important components of images.

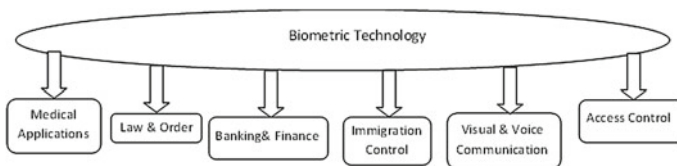


Fig. 2.17 Main modern BT applications

¹¹Grand View Research, 2014, <http://www.grandviewresearch.com/industry-analysis/biometrics-industry>.

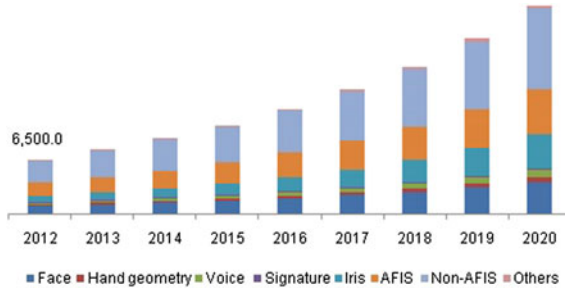


Fig. 2.18 Global biometrics technology market by application (USD Million), 2012–2020 (see footnote 11)

- (b) *Appearance based approach* has the main purpose to keep the important information of image and reject the redundant information, like examples being the Principal Component Analysis (PCA), Discrete Cosine Transformation (DCT), Independent Component Analysis (ICA), Linear Discriminant Analysis (LDA).
- (c) *Template Based Techniques* uses an appropriate energy function to extract the facial features based on the previously designed templates; the minimum energy is achieved for the best match of the template in facial image.
- (d) *Color based approach*—the skin color is used to isolate the face area from the nonface area in an image.

(C) the comparison of the template formed at step (B) to the templates in the considered database.

The performance of the face recognition systems is affected by a lot of factors such as:

- variations in lighting conditions;
- camera angles;
- changes in position and expression of the face.

A very important stage of the FR process is called *eigenfaces* and it consists in the determination of the best facial features, which discriminates the features of a subject from those of another face.

We have experimented [30] our model using the *ORL Database of Faces*, provided by the AT&T Laboratories from Cambridge University.

The database has 400 images, corresponding to 40 subjects (namely, 10 images for each of the 40 classes). We have divided the whole gallery into a training lot (200 pictures) and a test lot (200 pictures). Each image has the size of 92×112 pixels with 256 levels of gray. For the same subject (class), the images have been taken at different hours, lighting conditions, and facial expressions, with or without glasses. For each class, one chooses five images for training and five images for test.

Figure 2.19 illustrates 100 images from the ORL face database.



Fig. 2.19 100 images from the ORL face database

2.6.1 Applying the Fuzzy Kwan–Cai Neural Network for Face Recognition

We have performed the software implementation of the FKCNN to be experimented for the face recognition task, both in the case of 100 images (10 classes) [30, 52], and 400 images (40 classes) [52], too. A half images from the used database constitutes the training lot of the FKCNN and the other half, its test lot.

The images belonging to the training and respectively from the test lot (being different from those that are in training lot) have been applied alternatively and in only one learning epoch.

After the training algorithm of the FKCNN, it is necessary to preserve its parameters and go to the network calibration (partial supervision). This stage also means a testing on the training lot (normally, we have to obtain here a recognition score of 100%).

The output neurons that yield the maximum value 1 constitutes the label corresponding to the training input image.

We need the following algorithm to classify a test image:

- (1) find the label of that neuron of the fourth level of the FKCNN, whose output is equal to 1;
- (2) check if the label of the neuron coincides with the label of the respective input image to see if the recognition is correctly; otherwise it is erroneously.

The score of the correct recognition can be computed using the formula:

$$R [\%] = \frac{\text{the number of the correct recognitions}}{\text{the total number of the test images}} \times 100. \quad (2.43)$$

Remark 2.2 The same procedure and formula is also applied for the training lot, in the calibration stage to validate the FKCNN learning.

The following parameters are chosen in order to guarantee a good behavior of the learning algorithm:

- the parameter α characterizing the nonlinear output functions of the MAX-FN's in the second layer;
- the parameter β , characterizing the fuzzification function;
- T_f -the fault tolerance of the FKCNN.

We shall further evaluate the effects of the incoming parameters α , β , and T_f on the recognition rate and on the number M of output neurons obtained during the learning procedure.

Some experimental results are given in Table 2.6.

From Table 2.6 we can evaluate that the very good recognition score of 94% is obtained by the FKCNN on the test lot of 50 images, for the following two optimum combinations of parameters:

- (1) $\alpha = 1.5, \beta = 0.014, T_f = 0.052$;
- (2) $\alpha = 2, \beta = 0.014, T_f = 0.039$.

Table 2.6 The recognition rate R [%] of the FKCNN on the test lot (50 images of 10 classes) as a function of the parameters α , β , and T_f

α	β	T_f	R [%]
1.5	0.015	0.076	90
	0.0145	0.061	92
	0.014	0.052	94
	0.0135	0.051	92
	0.013	0.048	84
2	0.015	0.057	90
	0.0145	0.045	92
	0.014	0.039	94
	0.0135	0.038	92
	0.013	0.036	84

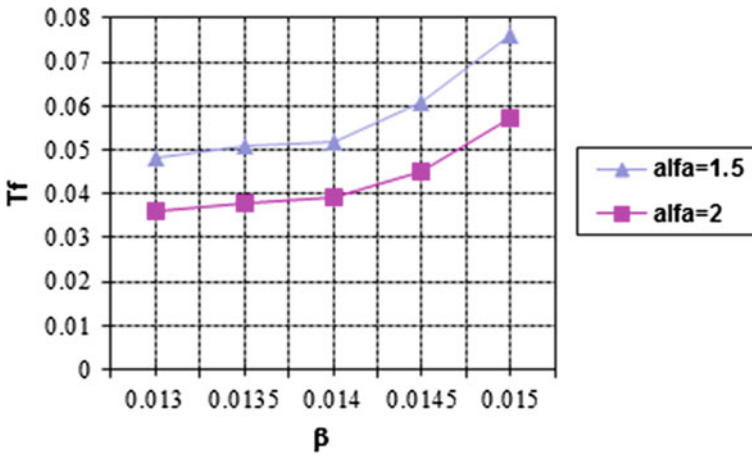


Fig. 2.20 The fault tolerance T_f as a function of the parameter β , in order to obtain good recognition rates

The fault tolerance T_f has been chosen in order to obtain a number of MIN-FN's on the third layer equal to the number of images belonging to the training lot.

The FKCNN is very sensitive [30] to the selection of the parameter β . For example, if $\alpha = 2$ then a modification of the parameter β with only 0.001 (from 0.014 to 0.013) produces the decreasing of the recognition rate with 10% (from 94 to 84%).

Conversely, the parameter $\alpha = 2$ has a reduced influence on the recognition score.

From Fig. 2.20 we can notice that to obtain good recognition rates (for example $R \geq 84\%$, according to the results given in Table 2.6), when the parameter β increases, one has to increase the fault tolerance T_f , too.

The matrix C shows the confusion matrix for the best recognition score ($R = 94\%$, $\alpha = 2$, $\beta = 0.014$, $T_f = 0.039$).

$$\begin{array}{c}
 \text{Assigned class} \\
 \mathbf{C} =
 \end{array}
 \begin{array}{c}
 \text{Real class} \\
 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10
 \end{array}
 \begin{pmatrix}
 80 & 0 & 0 & 0 & 0 & 0 & 20 & 0 & 0 & 0 \\
 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 20 & 80 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 80 & 20 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100
 \end{pmatrix}$$

The matrix C illustrates that in the case when $\alpha = 2$, $\beta = 0.014$, $T_f = 0.039$, the FKCNN correctly recognizes 47 images of the 50 from the test lot.

In Fig. 2.21 are presented the labels that FKCNN assigned to the images of the test lot.



Fig. 2.21 Labeling of test lot for $\alpha = 2$, $\beta = 0.045$, $T_f = 0.114$

Table 2.7 The recognition rate R [%] of the FKCNN on the test lot (200 images of 40 classes) as a function of the parameters α , β , and T_f

α	β	T_f	R [%]
1.5	0.05	0.167	80.5
	0.045	0.153	82.5
	0.004	0.133	82.5
	0.035	0.114	82.5
	0.03	0.096	79.5
2	0.05	0.125	80.5
	0.045	0.114	82.5
	0.04	0.1	82.5
	0.035	0.085	82.5
	0.03	0.072	79.5

Table 2.7 proves that we have also achieved [52] with the FKCNN, a recognition rate of 100 % over the training lot for a database of 40 classes, as in the hypothesis of using 10 classes (see Table 2.6).

From Table 2.7 we can notice that a good recognition score of 82.5 % can be achieved by the FKCNN on the test lot of 200 images, by choosing the following parameters:

- (1) $\alpha = 1.5$ with
 - (a) $\beta = 0.045$, $T_f = 0.153$;
 - (b) $\beta = 0.04$, $T_f = 0.133$;
 - (c) $\beta = 0.035$, $T_f = 0.114$;
- (2) $\alpha = 2$ with
 - (a) $\beta = 0.045$, $T_f = 0.114$;
 - (b) $\beta = 0.04$, $T_f = 0.1$;
 - (c) $\beta = 0.035$, $T_f = 0.085$;

The confusion matrix D depicted in Fig. 2.22 corresponds to the case: $\alpha = 2$, $\beta = 0.045$, $T_f = 0.114$ from Table 2.7.

Figure 2.23 shows the way how are chosen the values of the fault tolerance T_f as a function of β , for $\alpha = 1.5$ and $\alpha = 2$, in the case when $M = 40$.

Figure 2.24 shows how to choose the values of T_f as a function of β , for $\alpha = 1.5$ and $\alpha = 2$ ($M = 10$ and $M = 40$).

From Fig. 2.24 we can deduce that the FKCNN is less sensitive to the choice of β in the case of experimenting this neural approach for a database of 40 classes (consisting in 400 images) than when we applied the FKCNN for the classification of 10 classes of images.

Figure 2.25 presents the recognition rates over the test lot, being functions of the β values, for $\alpha = 1.5$ and $\alpha = 2$ ($M = 10$ and $M = 40$).

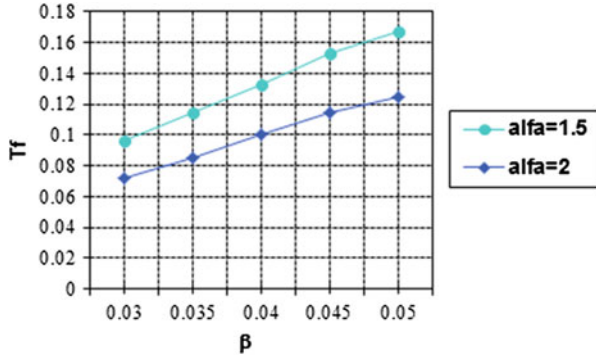
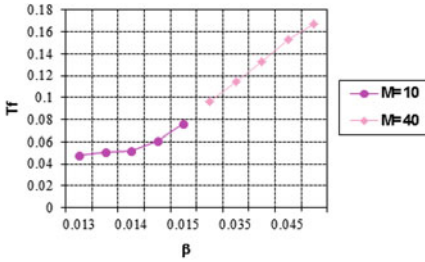
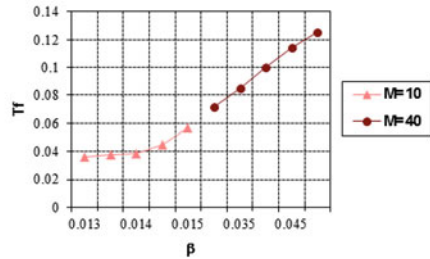


Fig. 2.23 The fault tolerance T_f as a function of the parameter β , in order to obtain good recognition rates

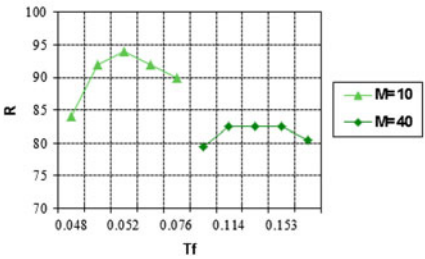


(a) $\alpha = 1.5$ ($M = 10$ and $M = 40$)

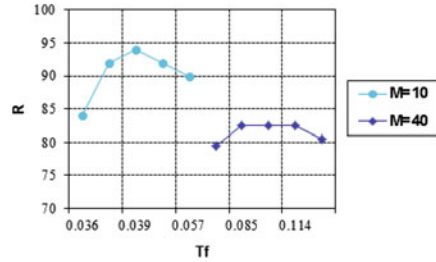


(b) $\alpha = 2$ ($M = 10$ and $M = 40$)

Fig. 2.24 The values of T_f as a function of β



(a) $\alpha = 1.5$ ($M = 10$ and $M = 40$)



(b) $\alpha = 2$ ($M = 10$ and $M = 40$)

Fig. 2.25 The values of T_f as a function of β

The FKCNN is also sensitive to the choice of T_f (less sensitive than sensitive to the choice of β) in the meaning that for α and β fixed, if the value of T_f will be lower than that from Table 2.6 and respectively Table 2.7 then FKCNN will not be able to distinguish between the different images from the training lot.

From Tables 2.6 and 2.7, it results that:

- in case when the values of β decreases and α remains constant, the value of T_f has to decrease in order to achieve good recognition scores on the test lot;
- if we preserve the value of β and diminish the value of α , then T_f has to increase in order that the FKCNN to distinguish the images applied at its input;
- when we experimented FKCNN for a database of 40 classes we have to choose some bigger values of β and T_f than in the situation of working with 10 classes to achieve good recognition rates over the test lot.

The FKCNN approach is a special one as its learning algorithm takes place throughout a single training epoch and its parameters do not need to be refined.

The advantage of using the FKCNN for face recognition consists in the fact that we can obtain good recognition rates over the test lot only through a training epoch, but for the optimum choice of the parameters α , β , and T_f .

2.6.2 Applying Kohonen Maps for Feature Selection

The feature selection stage can be achieved using the Self-organizing Kohonen map (SOKM), which performs [53] the transformation of the original space of the features, in a two-dimensional space, associated to a planar rectangular network.

The projection of the vectors from the initial space is carried out through their association with one of the neurons in the SOKM and the two-dimensional projection will be given by the coordinates of that neuron in the network.

If the training algorithm of the SOKM stops before the network to succeed net classification of the input vectors, then the vectors corresponding to the same class will be associated to some neighboring neurons within the network. The projection of the n -dimensional vectors from the input space into a two-dimensional space is represented by the plane of the network neurons and it makes in the following way [53]:

- (a) one assigns a coordinate system to the rectangular network;
- (b) determines the projection of an input vector like being given by the coordinates of the neuron, which is associated with the respective vector within the network.

The use of this neural network in the case of the projection in the two-dimensional space aims to highlight the performances of Kohonen network in comparison with situations when in the feature selection are used some nonneural projection algorithms.

Unlike the classification of the n -dimensional vectors using the SOKM, the projection of the vectors in the plane has the advantage that it is much faster (it does not

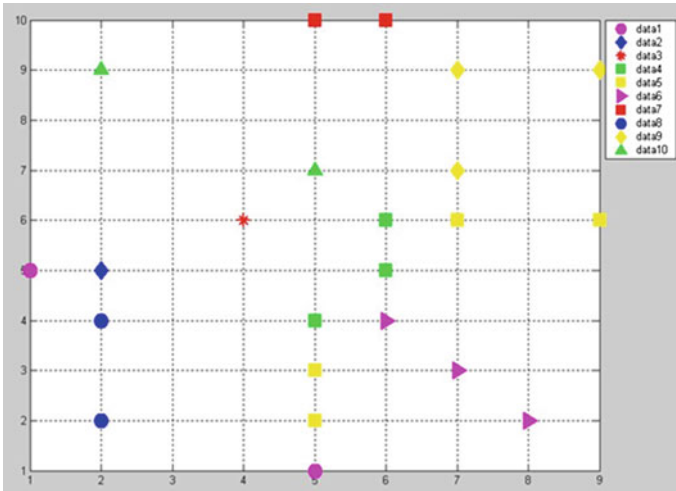


Fig. 2.26 Feature selection using SOKM

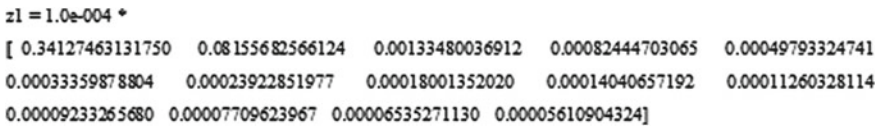


Fig. 2.27 Change of the weights associated with the network neurons

require a large number of iterations) and the resulting two-dimensional vectors can be further classified using a simpler algorithm.

We built a Matlab program to illustrate the training of the SOKM using the 50 images from the ORL face database (see Fig. 2.19) as its input patterns. We have chosen that the value of the neighborhood radius is equal to 10 at the beginning of the refining; after a fixed number of 14 training epochs we represented in Fig. 2.26 the projection result.

We computed

$$z1_{it} = \min_{i=1}^{112 \times 92} \|w_{ijk}(it + 1) - w_{ijk}(it)\|, (\forall) j, k = \overline{1, 10}$$

to depict (in Fig. 2.27) the change of the weights associated with the network neurons, from one epoch it to another $it + 1$.

References

1. I. Iatan and M. Worring. A fuzzy Kwan–Cai neural network for determining image similarity. *BioSystems (Under Review)*, 2016.
2. G. P. Nguyen and M. Worring. Interactive access to large image collections using similarity-based visualization. *Journal of Visual Languages and Computing*, 19:203–224, 2008.
3. G. P. Nguyen and M. Worring. Optimization of interactive visual-similarity-based search. *ACM Transactions on Multimedia Computing Communications and Applications*, 4 (1):1–23, 2008.
4. G. P. Nguyen, M. Worring, and A. W. M. Smeulders. Similarity learning via dissimilarity space in CBIR. In *Proceedings of the ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 107–116, 2006.
5. G. Strong and M. Gong. Similarity-based image organization and browsing using multi-resolution self-organizing map. *Image and Vision Computing*, 29:774–786, 2011.
6. S.S. Chowhan. Iris recognition using fuzzy min-max neural network. *International Journal of Computer and Electrical Engineering*, 3 (5):743–747, 2011.
7. M. Hariri, S.B. Shokouhi, and N. Mozayani. An improved fuzzy neural network for solving uncertainty in pattern classification and identification. *Iranian Journal of Electrical & Electronic Engineering*, 4 (3):79–93, 2008.
8. H. K. Kwan and Y. Cai. A fuzzy neural network and its application to pattern recognition. *IEEE Trans. on Fuzzy Systems*, 2 (3):185–193, 1997.
9. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. A fuzzy neural network and its application to pattern recognition. *IEEE Trans. on Fuzzy Systems*, 88:303–338, 2010.
10. B. Zaka. Theory and applications of similarity detection techniques. http://www.iicm.tugraz.at/thesis/bilal_dissertation.pdf, 2009.
11. K. Suzuki, H. Yamada, and S. Hashimoto. A similarity-based neural network for facial expression analysis. *Pattern Recognition Letters*, 28:1104–1111, 2007.
12. C.M. Hwang, M.S. Yang, W.L. Hung, and M.G. Lee. A similarity measure of intuitionistic fuzzy sets based on the Sugeno integral with its application to pattern recognition. *Information Sciences*, 189:93–109, 2012.
13. Y. Chen, E.K. Garcia, M.Y. Gupta, A. Rahimi, and A. Cazzanti. Similarity-based classification: Concepts and algorithms. *Journal of Machine Learning Research*, 10:747–776, 2009.
14. S. Wang, Q. Huang, S. Jiang, Q. Tian, and L. Qin. Nearest-neighbor method using multiple neighborhood similarities for social media data mining. *Neurocomputing*, 2012.
15. S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21 (9):871–883, 1999.
16. G.D. Guo, A.N. Jain, and W.Y. Ma. Learning similarity measure for natural image retrieval with relevance feedback. *IEEE Transactions on Neural Networks*, 13 (4):811–820, 2002.
17. A. Mellouk and A. Chebira. *Machine Learning*. InTech, 2009.
18. T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. pages 1–14. ECCV - European Conference on Computer Vision, <http://hal.inria.fr/docs/00/72/23/13/PDF/mensink12eccv.final.pdf>, 2012.
19. N. A. Chinchor, and Wong P. C. Thomas, J. J., M. G. Christel, and W. Ribarsky. Multimedia analysis + visual analytics = multimedia analytics. *Computer Graphics and Applications, IEEE*, 30 (5):52–60, 2010.
20. W. Lin, D. Tao, J. Kacprzyk, Z. Li, E. Izquierdo, and H. Wang. *Multimedia Analysis, Processing and Communications*. Springer-Verlag Berlin Heidelberg, 2011.
21. X. Li, C. G. M. Snoek, and M. Worring. Learning tag relevance by neighbor voting for social image retrieval. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 180–187, 2008.
22. J. Perkiö, A. Tuominen, and P. Myllymäki. Image similarity: From syntax to weak semantics using multimodal features with application to multimedia retrieval. *Multimedia Information Networking and Security*, 1:213–219, 2009.

23. A. Huang. Similarity measures for text document clustering. In *NZCSRSC*, pages 49–56, 2008.
24. V. Dutt, V. Chadhury, and I. Khan. Different approaches in pattern recognition. *Computer Science and Engineering*, 1 (2):32–35, 2011.
25. R.C. Chakraborty. Fundamentals of neural networks. http://www.myreaders.info/html/artificial_intelligence.htm, 2010.
26. J.K. Basu, D. Bhattacharyya, and T.H. Kim. Use of artificial neural network in pattern recognition. *International Journal of Software Engineering and Its Applications*, 4 (2):22–34, 2010.
27. Yu J., Wang M., and Tao D. Semisupervised multiview distance metric learning for cartoon synthesis. *IEEE Transactions on Image Processing*, 21 (11):4636–4648, 2012.
28. Yu J., Tao D., and Wang M. Adaptive hypergraph learning and its application in image classification. *IEEE Transactions on Image Processing*, 21 (7):3262–3272, 2012.
29. Yu J., Rui Y., Tang Y.Y., and Tao D. High-order distance-based multiview stochastic learning in image classification. *IEEE Transactions on Cybernetics*, 44 (12):2431–2442, 2014.
30. V. Neagoe and I. Iatan. A neuro-fuzzy approach to face recognition. In *Proceedings of 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), 14-18 July 2002, Orlando, Florida, XIV*, pages 120–125, 2002.
31. V. E. Neagoe and O. Stănaşilă. *Pattern Recognition and Neural Networks (in Romanian)*. Ed. Matrix Rom, Bucharest, 1999.
32. I. Iatan. Iris recognition by non- parametric techniques. In *Proceedings of International Conference Trends and Challenges in Applied Mathematics, June 20-23, Bucharest*, pages 220–223, 2007.
33. A. Krzyzak. Welcome to pattern recognition homepage. Course Web Page: http://www.cs.concordia.ca/~comp473_2/fall2005/Notes.htm, 2005.
34. F. P. Romero, A. Peralta, A. Soto, J. A. Olivas, and J. Serrano-Guerrero. Fuzzy optimized self-organizing maps and their application to document clustering. *Soft Computing*, 14:857–867, 2010.
35. T.N. Yap. Automatic text archiving and retrieval systems using self-organizing kohonen map. In *Natural Language Processing Research Symposium*, pages 20–24, 2004.
36. I. Bose and C. Xi. Applying Kohonen vector quantization networks for profiling customers of mobile telecommunication services. In *The Tenth Pacific Asia Conference on Information Systems (PACIS 2006)*, pages 1513–1526, 2006.
37. M. Ettaouil, Y. Ghanou, K. El Moutaouakil, and M. Lazaar. Image medical compression by a new architecture optimization model for the Kohonen networks. *International Journal of Computer Theory and Engineering*, 3 (2):204– 210, 2011.
38. M. Ettaouil and M. Lazaar. Compression of medical images using improved Kohonen algorithm. *Special Issue of International Journal of Computer Applications on Software Engineering, Databases and Expert Systems SEDEXS*, pages 41– 45, 2012.
39. M. Lange, D. Nebel, and T. Villmann. *Partial Mutual Information for Classification of Gene Expression Data by Learning Vector Quantization*, pages 259– 270. *Advances in Self-Organizing Maps and Learning Vector Quantization*. Springer, 2014.
40. A.N. Netravali and B.G. Haskell. *Digital Pictures: Representation and Compression*. Springer, 2012.
41. V. Neagoe. A neural approach to compression of hyperspectral remote sensing imagery. In B. Reusch, editor, *Computational Intelligence, Theory and Applications, International Conference, 7th Fuzzy Days, Dortmund, Germany, October 1-3, 2001*, volume 2206 of *Lecture Notes in Computer Science*, pages 436–449, 2001.
42. N.A. AL-Allaf Omaima. Codebook enhancement in vector quantization image compression using backpropagation neural network. *Journal of Applied Sciences*, 11:3152–3160, 2011.
43. R. Lamba and M. Mittal. Image compression using vector quantization algorithms: A review. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3 (6):354–358, 2013.
44. V.E. Neagoe. Pattern recognition and artificial intelligence (in Romanian), lecture notes, Faculty of Electronics, Telecommunications and Information Technology, University Politehnica of Bucharest. 2000.

45. F.J. Janse van Rensburg, J. Treurnicht, and C.J. Fourie. The use of fourier descriptors for object recognition in robotic assembly. In *5th CIRP International Seminar on Intelligent Computation in Manufacturing Engineering*. https://www.academia.edu/754136/The_Use_of_Fourier_Descriptors_for_Object_Recognition_in_Robotic_Assembly, 2006.
46. R.A. Tuduca. *Signal Theory*. Bren, Bucharest, 1998.
47. T. Kohonen. *Self-Organizing Maps*. Berlin: Springer-Verlag, 1995.
48. V. E. Neagoie. Concurrent self-organizing maps for automatic face recognition. In *Proceedings of the 29th International Conference of the Romanian Technical Military Academy, November 15-16, 2001, Bucharest, Romania*, pages 35–40, 2001.
49. K. E. A. van de Sande, T. Gevers, and C. G. M. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32 (9):1582–1596, 2010.
50. M.A. Anjum. *Improved Face Recognition using Image Resolution Reduction and Optimization of Feature Vector*. PhD thesis, National University of Sciences and Technology (NUST) Rawalpindi Pakistan, 2008.
51. S. Dhawan and H. Dogra. Feature extraction techniques for face recognition. *International Journal of Engineering, Business and Enterprise Applications*, 2 (1):1–4, 2012.
52. I. Iatan. *Neuro- Fuzzy Systems for Pattern Recognition (in Romanian)*. PhD thesis, Faculty of Electronics, Telecommunications and Information Technology- University Politehnica of Bucharest, PhD supervisor: Prof. dr. Victor Neagoie, 2003.
53. I. Iatan. Unsupervised neural models and their applications for the feature selection and pattern classification (in Romanian). Master's thesis, Faculty of Mathematics and Computer Science- University of Craiova, PhD supervisor: Prof. dr. Victor Neagoie, June 1998.



<http://www.springer.com/978-3-319-43870-2>

Issues in the Use of Neural Networks in Information
Retrieval

Iatán, I.F.

2017, XIX, 199 p. 88 illus., 44 illus. in color., Hardcover

ISBN: 978-3-319-43870-2