On-Line Retrainable Neural Networks: Improving the Performance of Neural Networks in Image Analysis Problems

Anastasios D. Doulamis, *Member, IEEE*, Nikolaos D. Doulamis, *Member, IEEE*, and Stefanos D. Kollias, *Member, IEEE*

***Abstract*—A novel approach is presented in this paper for improving the performance of neural-network classifiers in image recognition, segmentation, or coding applications, based on a retraining procedure at the user level. The procedure includes: 1) a training algorithm for adapting the network weights to the current condition; 2) a maximum *a posteriori* (MAP) estimation procedure for optimally selecting the most representative data of the current environment as retraining data; and 3) a decision mechanism for determining when network retraining should be activated. The training algorithm takes into consideration both the former and the current network knowledge in order to achieve good generalization. The MAP estimation procedure models the network output as a Markov random field (MRF) and optimally selects the set of training inputs and corresponding desired outputs. Results are presented which illustrate the theoretical developments as well as the performance of the proposed approach in real-life experiments.**

***Index Terms*—Image analysis, MPEG-4, neural-network retraining, segmentation, weight adaptation.**

I. Introduction

PROBABLY the most important issue when designing and training artificial neural networks in real-life applications is network generalization. Many significant results have been derived during the last few years regarding generalization of neural networks when tested outside their training environment [23], [32]. Examples include algorithms for adaptive creation of the network architecture during training, such as pruning or constructive techniques, modular and hierarchical networks, or theoretical aspects of network generalization, such as the VC dimension. Specific results and mathematical formulations regarding error bounds and overtraining issues have been obtained when considering cases with known probability distributions of the data [4], [7], [16], [40]. Despite, however, the achievements obtained, most real-life applications do not obey some specific probability distribution and may significantly differ from one case to another mainly due to changes of their environment. That is why straightforward application of trained networks, to data outside the training set, is not always adequate for solving image recognition, classification or detection problems.

Manuscript received June 18, 1998; revised June 22, 1999 and October 4, 1999.

The authors are with the Electrical and Computer Engineering Department, National Technical University of Athens, Zografou 15773, Athens, Greece (e-mail: stefanos@cs.ntua.gr).

Publisher Item Identifier S 1045-9227(00)00875-4.

This is, for example, the case in remote sensing applications. In particular, recent studies, on the use of neural networks for classification and segmentation of images representing different geographical locations, have shown that neural networks trained at a specific location may result in as low as 25% classification accuracy if tested on another area. Instead, retraining of the networks, using data from both localities, was found necessary for achieving classification ratios of about 80% in either area [41]. This geographical generalization problem can be due to various reasons, including the fact that various locations of the earth can be quite different from each other, the fact that the conditions of image capturing are generally different, or even the fact that training sets may not be representative enough.

Another example of particular interest concerns applications to video processing and analysis. Neural networks have not played a significant role in the development of video coding standards, such as MPEG-1 and MPEG-2 [17], [18]. Nevertheless the forthcoming MPEG-4 and MPEG-7 standards [19], [28] referring to content-based video coding [1], [8], [10], storage, retrieval, and indexing [9], [11], [13] are based on physical object extraction from scenes, so as to handle multimedia information with an increased level of intelligence. Neural networks, with their superior nonlinear classification abilities, can play a major role in these forthcoming multimedia standards [22]. However, the above-mentioned problems in the generalization of neural networks when used in environments which are different from their training ones constitute a major obstacle, especially when dealing with applications where large variations of image and video scenes are frequently met. Apart from image and video coding, the above hold in a large variety of applications, including medical imaging [30], invariant object recognition [21], e.g., in factory environments and in robot vision, human face detection [12], [29], [39], and nonlinear system identification where the system to be identified is changing with time.

In most of the above cases, the adopted assumption of stationarity of the network training data is not valid. Consequently, the training set is not able to represent all possible variations or states of the operational environment to which the network is to be applied. Instead, it would be desirable to have a mechanism, which would provide the network with the capability to automatically test its performance and be automatically retrained when its performance is not acceptable. The retraining algorithm should update the network weights taking into ac-

count both the former network knowledge and the knowledge extracted from the current input data.

Adaptive training of neural networks in slowly varying nonstationary processes has been a topic of research within the neural-network community in the last few years [31], [33]. However, the proposed techniques have focused on the problem of weight adaptation, assuming that retraining is manually activated and that retraining input and desired output data are *a priori* known.

A novel approach is presented in this paper for improving the performance of neural networks when handling nonstationary image and video data, including:

- an automatic decision mechanism, which determines when network retraining, should take place;
- a maximum *a posteriori* (MAP) estimation technique which extracts knowledge from the current input data by modeling the network output as a Markov random field (MRF) and optimally selecting pairs of training inputs and corresponding desired outputs;
- an algorithm which retrains the network, adapting its weights to the current environment, by applying a nonlinear programming technique.

This paper is organized as follows. Formulation of the problem is given in Section II, while the retraining algorithm, the optimal selection of the training data and the mechanism for deciding whether retraining is needed are presented in Sections III–V, respectively. Application of the proposed approach to a real-life problem related to the new multimedia standards, which is extraction of the head and shoulder parts of humans from image sequences in videophone communications, is presented in Section VI. Conclusions and suggestions for further work are given in Section VII of the paper.

II. Formulation of the Problem

Let $x_{i,j}$, $i = 0, \cdots, M_1 - 1$, $j = 0, \cdots, M_2 - 1$ denote the image intensity at pixel (i, j). In most image/video coding and analysis applications the image is processed in partitions, or blocks, of, say, 8 × 8 pixels. For classification purposes it is required to classify each block, or a transformed version of it of the same size, to one of, say, p available classes ω_i, $i = 1, 2, \cdots, p$. Let $\underline{x}_i$ be a vector containing the lexicographically ordered values of the ith block to be classified. A neural-network classifier will produce a p-dimensional output vector $\underline{y}(\underline{x}_i)$ defined as follows:

$$\underline{y}(\underline{x}_i) = \left[p^i_{\omega_1}\, p^i_{\omega_2}\, \cdots\, p^i_{\omega_p}\right]^T \tag{1}$$

where $p^i_{\omega_j}$ denotes the probability that the ith image block belongs to the jth class.

Let us first consider that a neural network has been initially trained to perform the previously described classification task using a specific training set, say, $S_b = \{(\underline{x}'_1, \underline{d}'_1), \cdots, (\underline{x}'_{m_b}, \underline{d}'_{m_b})\}$, where vectors $\underline{x}'_i$ and $\underline{d}'_i$ with $i = 1, 2, \cdots, m_b$ denote the ith input training vector, i.e., the ith image block or a transformed version of it, and the corresponding desired output vector consisting of p elements. Let $\underline{y}(\underline{x}_i)$ denote the network output when applied to the ith block of an image outside the training set. Whenever a change of the environment occurs, new network weights should be estimated through a retraining procedure, taking into account both the former network knowledge and the current situation. Let us consider retraining in more detail. Let $\underline{w}_b$ include all weights of the network before retraining, and $\underline{w}_a$ the new weight vector which is obtained through retraining. A training set S_c is assumed to be extracted from the current operational situation composed of, say, m_c image blocks of 8 × 8 pixels; $S_c = \{(\underline{x}_1, \underline{d}_1), \cdots, (\underline{x}_{m_c}, \underline{d}_{m_c})\}$ where $\underline{x}_i$ and $\underline{d}_i$ with $i = 1, 2, \cdots, m_c$ similarly correspond to the ith input and desired output retraining data. The retraining algorithm that is activated as described in Section V, whenever a change of the environment is detected, computes the new network weights $\underline{w}_a$, by minimizing the following error criterion with respect to the weights:

$$E_a = E_{c,a} + \eta E_{f,a} \tag{2}$$

with

$$E_{c,a} = \tfrac{1}{2} \sum_{i=1}^{m_c} \|\underline{z}_a(\underline{x}_i) - \underline{d}_i\|_2 \tag{2a}$$

and

$$E_{f,a} = \tfrac{1}{2} \sum_{i=1}^{m_b} \|\underline{z}_a(\underline{x}'_i) - \underline{d}'_i\|_2 \tag{2b}$$

where

- $E_{c,a}$ error performed over training set S_c ("current" knowledge);
- $E_{f,a}$ corresponding error over training set S_b ("former" knowledge);
- $\underline{z}_a(\underline{x}_i)$ output of the retrained network, corresponding to input vector $\underline{x}_i$ of the network consisting of weights $\underline{w}_a$;
- $\underline{z}_a(\underline{x}'_i)$ output of the retrained network, corresponding to input vector $\underline{x}'_i$ to the network consisting of weights $\underline{w}_a$.

Similarly, $\underline{z}_b(\underline{x}_i)$ would represent the output of the network, consisting of weights $\underline{w}_b$, when accepting vector $\underline{x}_i$ at its input. When retraining the network for the first time, $\underline{z}_b(\underline{x}_i)$ is identical to $\underline{y}(\underline{x}_i)$. Parameter η is a weighting factor accounting for the significance of the current training set compared to the former one and $\|\cdot\|_2$ denotes the L_2-norm.

In most real-life image or video analysis applications, training set S_c is *a priori* unknown; consequently estimation of S_c as well as detection of the change of the environment should be automatically provided by the system. As a result, apart from the retraining algorithm, two additional modules are embedded in the proposed architecture; a "change detection" decision mechanism and a MAP training set estimation procedure. The main objectives of these modules are briefly described next. A block-diagram of the overall architecture is presented in Fig. 1. The initially trained neural-network classifier and the retrained one are also included in the figure. The initially trained network is assumed to provide a coarse approximation of the correct classification irrespectively of changes of the environment. Possible misclassifications provided by this classifier are then corrected

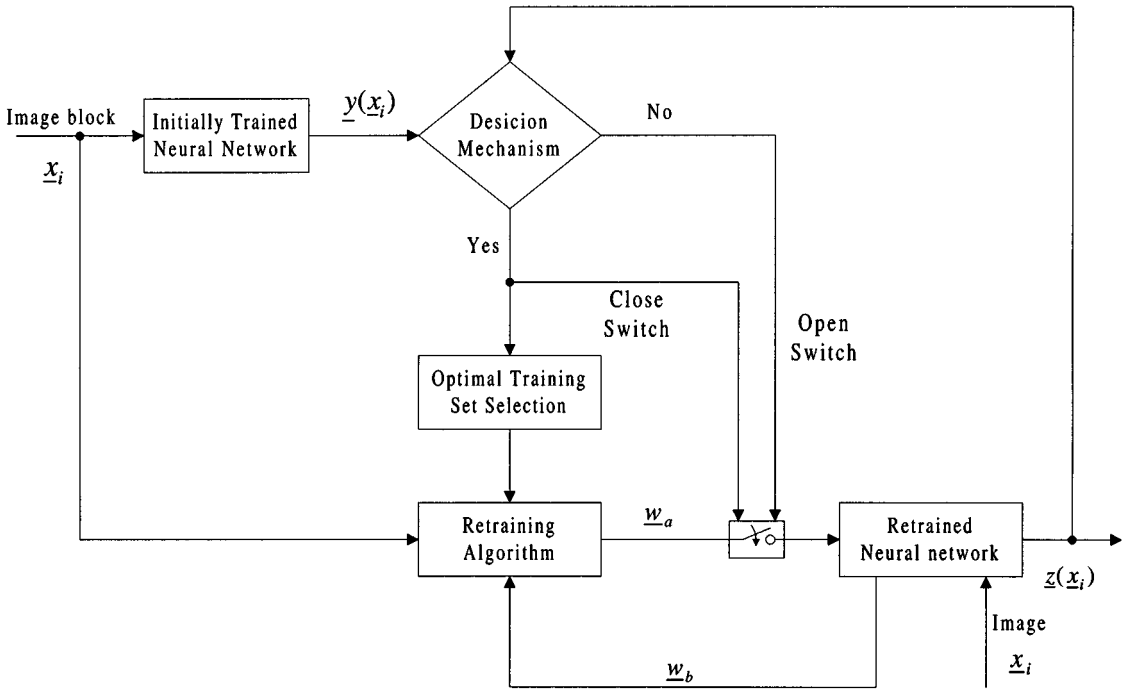

Fig. 1. The system architecture.

by the retrained network, which generates the final classification.

1) Decision Mechanism: The goal of this module is to determine when retraining should be activated, or equivalently, to detect the time instances when a change of the environment occurs. Whenever network performance is considered satisfactory (no change of the environment occurs), the network weights and structure remain the same. Instead, new network weights are calculated when network performance is not appropriate, by activating both the MAP estimation procedure and the retraining algorithm. Investigation of the performance of the initially trained network on the current data is the basis of the mechanism, as described in Section V.

2) MAP Estimation Procedure: This module optimally selects a training set of input and desired output data which optimally represent the current situation as described in Section IV. This set is used next by the retraining algorithm described in the following section.

III. The Retraining Technique

A. The Network Architecture

Let us, for simplicity, consider: 1) a two-class classification problem, where classes ω_1, ω_2 refer, for example, to foreground and background objects in an image and 2) a feedforward neural-network classifier which includes a single output neuron providing classification in two categories, one hidden layer consisting of q neurons, and accepts image blocks of, say, J pixels at its input. Let us also ignore neuron thresholds. Extension to classification problems and networks of higher complexity can be performed in a similar way. Fig. 2 presents the structure of the network. As mentioned before, $\underline{w}_b$ and $\underline{w}_a$ denote the corresponding network weights, before and after retraining has been performed to a specific image or video frame. Let $\underline{w}^1_{\{a,b\}}$ denote a vector containing the $q \times 1$ weights between the output and hidden neurons and $\underline{w}^0_{k,\{a,b\}}$, $k = 1, 2, \cdots, q$ denote the $J \times 1$ vector of weights between the kth hidden neuron and the network inputs, where subscripts $\{a, b\}$ refer to either the situation "after" or "before" retraining, respectively. An illustration of weights $\underline{w}^0_{k,\{a,b\}}$ and $\underline{w}^1_{\{a,b\}}$ is presented in Fig. 2. Then

$$\underline{w}_{\{a,b\}} = \left[\left(\underline{w}^0_{1,\{a,b\}}\right)^T \cdots \left(\underline{w}^0_{q,\{a,b\}}\right)^T \left(\underline{w}^1_{\{a,b\}}\right)^T\right]^T \quad (3)$$

is a vector containing all network weights. For a given input vector $\underline{x}_j$, corresponding to the jth image block, the output of the final neural-network classifier is given by

$$z_{\{a,b\}}(\underline{x}_j) = f\left(\left(\underline{w}^1_{\{a,b\}}\right)^T \cdot \underline{u}_{\{a,b\}}(\underline{x}_j)\right) \quad (4)$$

where $f(\cdot)$ denotes the activation function of the output neuron, e.g., a sigmoidal function, and

$$\underline{u}_{\{a,b\}}(\underline{x}_j) = \left[u_{1,\{a,b\}}(\underline{x}_j) \cdots u_{q,\{a,b\}}(\underline{x}_j)\right]^T \quad (5)$$

is a vector containing the hidden neuron outputs when the network weights are $\underline{w}_b$ (before retraining) or $\underline{w}_a$ (after the retraining). The network output in (4) is scalar, since we have assumed a single network output. The output of the ith neuron of the first hidden layer can be written in terms of the input vector and the weights $\underline{w}^0_{i,\{a,b\}}$ as follows:

$$u_{i,\{a,b\}}(\underline{x}_j) = f\left(\left(\underline{w}^0_{i,\{a,b\}}\right)^T \cdot \underline{x}_j\right). \quad (6)$$

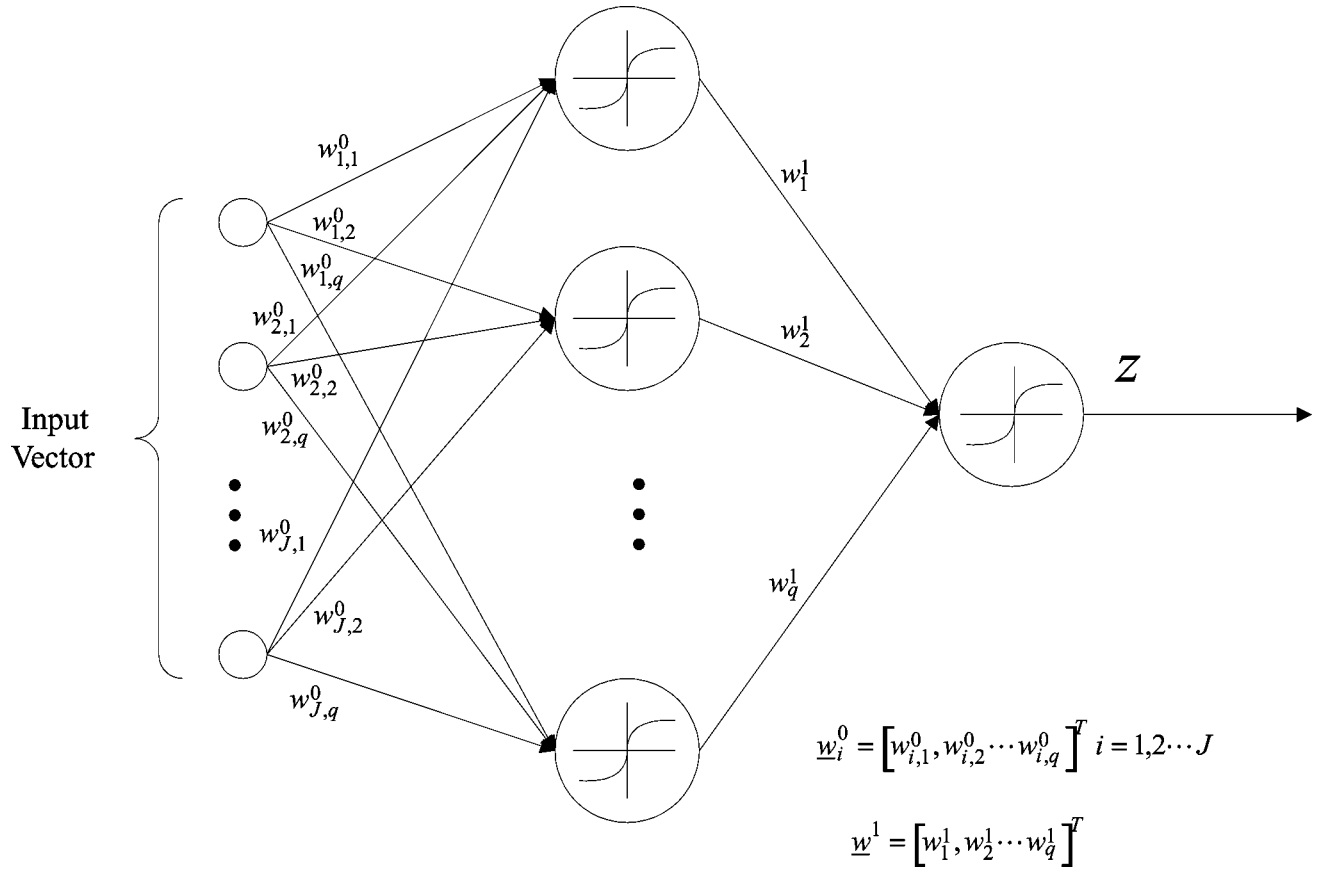

Fig. 2. The neural-network structure.

Thus (5) can be expressed as

$$\underline{u}_{\{a,b\}}(\underline{x}_j) = \underline{f}\left(\left(\boldsymbol{W}^0_{\{a,b\}}\right)^T \cdot \underline{x}_j\right) \tag{7}$$

where

$\boldsymbol{W}^0_{\{a,b\}}$ Jxq matrix defined as $\boldsymbol{W}^0_{\{a,b\}} = [\underline{w}^0_{1,\{a,b\}} \cdots \underline{w}^0_{q,\{a,b\}}]$

$\underline{f}(\cdot)$ vector-valued function the elements of which represent the activation function of a corresponding hidden neuron. The hyperbolic tangent sigmoidal function is used in the rest of the paper.

B. The Retraining Algorithm

The goal of the training procedure is to minimize (2) and estimate the new network weights $\underline{w}$, i.e., $\boldsymbol{W}^0_a$ and $\underline{w}^1_a$, respectively. Let us first assume that a small perturbation of the network weights (before retraining) $\underline{w}_b$ is enough to achieve good classification performance. Then

$$\boldsymbol{W}^0_a = \boldsymbol{W}^0_b + \Delta\boldsymbol{W}^0 \quad \text{and} \quad \underline{w}^1_a = \underline{w}^1_b + \Delta\underline{w}^1 \tag{8}$$

where $\Delta\boldsymbol{W}^0$and $\Delta\underline{w}^1$ are small increments. This assumption leads to an analytical and tractable solution for estimating $\underline{w}_a$, since it permits linearization of the nonlinear activation function of the neuron, using a first-order Taylor series expansion.

Equation (2) indicates that the new network weights are estimated taking into account both the current and the previous network knowledge. To stress, however, the importance of current training data in (2), one can replace (2a) by the constraint that the actual network outputs are equal to the desired ones, that is

$$z_a(\underline{x}_i) = d_i \quad i = 1, \cdots, m_c, \qquad \text{for all data in } S_c. \tag{9}$$

Equation (9) indicates that the first term of (2), corresponding to error $E_{c,a}$, takes values close to zero, after estimating the new network weights.

It is shown in Appendix B that, through linearization, solution of (9) with respect to the weight increments is equivalent to a set of linear equations

$$\underline{c} = \boldsymbol{A} \cdot \Delta\underline{w} \tag{10}$$

where $\Delta\underline{w} = [(\Delta\underline{w}^0)^T \; (\Delta\underline{w}^1)^T]^T$, $\Delta\underline{w}^0 = \vec{\{}\Delta\boldsymbol{W}^0\}$, with $\vec{\{}\Delta\boldsymbol{W}^0\}$ denoting a vector formed by stacking up all columns of $\Delta\boldsymbol{W}^0$; vector $\underline{c}$ and matrix $\boldsymbol{A}$ are appropriately expressed in terms of the previous network weights. The size of $\Delta\underline{w}$ is $N_w = (J+1)q$ for the network presented in Fig. 2. In particular, $\underline{c} = [z_a(\underline{x}_1) \cdots z_a(\underline{x}_{m_c})]^T - [z_b(\underline{x}_1) \cdots z_b(\underline{x}_{m_c})]^T$, expressing the difference between network outputs after and before retraining for all input vectors in S_c. Based on (9), vector $\underline{c}$ can be written as

$$\underline{c} = [d_1 \cdots d_{m_c}]^T - [z_b(\underline{x}_1) \cdots z_b(\underline{x}_{m_c})]^T. \tag{11}$$

Equation (10) is valid only when weight increments $\Delta\underline{w}$ are small quantities. In Appendix A, (A11) and (A13), it is shown

that, given a tolerated error value, proper bounds ϑ and ϕ can be computed for the weight increments, for each input vector $\underline{x}_i$ in S_c

$$-\vartheta(\underline{x}_i) \leq \left(\Delta \underline{w}_k^0\right)^T \cdot \underline{x}_i \leq \vartheta(\underline{x}_i), \qquad k = 1, 2, \cdots q$$

and

$$-\phi(\underline{x}_i) \leq (\Delta \underline{w})^T \cdot \underline{a}(\underline{x}_i) \leq \phi(\underline{x}_i) \tag{12}$$

where $\underline{a}$ is a vector depending on the former weights and on the input vector as described in Appendix B. Equation (12) should be satisfied for all $\underline{x}_i$ in S_c, so as to assure that each linear equation in (10) is valid within the given tolerated error. A less strict approach would require that the mean value of the inner products in (12), for all $\underline{x}_i$, be smaller than the mean value of the bounds.

The size of vector $\underline{c}$ is smaller than the number of unknown weights $\Delta \underline{w}$, since in general a small number, m_c, of training data, is chosen, either through a clustering or a principal component analysis technique. Thus, many solutions exist for (10), since the number of unknowns is much greater than the respective number of equations. Uniqueness, however, is imposed by an additional requirement, which takes into consideration the previous network knowledge. Among all possible solutions that satisfy (10) the one which causes a minimal degradation of the previous network knowledge is selected as the most appropriate. It is considered that the network weights before retraining, i.e., $\underline{w}_b$, have been estimated as an optimal solution over data of set S_b. Furthermore, the weights after retraining provide a minimal error over all data of the current set S_c, according to (9). Thus, minimization of the second term of (2), which expresses the effect of the new network weights over data set S_b, can be considered as minimization of the absolute difference of the error over data in S_b with respect to the previous and the current network weights. Similar conclusions are drawn in [31]. This means that the weight increments are minimally modified, resulting in the following error criterion:

$$E_S = \|E_{f,a} - E_{f,b}\|_2 \tag{13}$$

with $E_{f,b}$ defined similarly to $E_{f,a}$, with $\underline{z}_a$ replaced by $\underline{z}_b$ in the right-hand side of (2b).

It can be shown [31] that (13) takes the form of

$$E_S = \tfrac{1}{2} (\Delta \underline{w})^T \cdot \boldsymbol{K}^T \cdot \boldsymbol{K} \cdot \Delta \underline{w} \tag{14}$$

where the elements of matrix $\boldsymbol{K}$ are expressed in terms of the previous network weights $\underline{w}_b$ and the training data in S_b. Thus, the problem results in minimization of (14) subject to constraints (10) and (12).

The number of data belonging to set S_c is in general much smaller than the number of network weights. This is due to the fact that these data have been selected from the current condition using a clustering or principal component analysis technique to discard data of similar content and obtain more reliable solutions. Thus, network overfitting could arise if only these current data were taken into account. However, in our method, the network weights are estimated using data from both the current set S_c and the previous set S_b, as (10) and (14) indicate. The number of elements of S_b is in general greater than the number of network weights since this set includes all *a priori* network knowledge. Consequently, overfitting issues are not encountered in the proposed retraining algorithm, since the number of training data, i.e., the number of data in S_c and S_b, is greater than the respective network weights. In Section V, where experimental results are presented, specific values for the numbers of data in sets S_c and S_b are provided.

The error function defined by (14) is convex since it is of squared form [25]. The constraints in (10) are linear equalities, while the constraints in (12) are linear inequalities. Thus, the solution should lie on the hyper-surface defined by (10), satisfy the inequalities in (12) and simultaneously minimize the error function given in (14). A variety of methods can be used to estimate the weight increments based on minimization of (14) subject to (10) and (12). In this paper we adopt the gradient projection method, which is presented next.

As described in Section V, the decision mechanism ascertains whether a small adaptation of the network weights is sufficient to provide accurate classification. In case that such a small adaptation is not adequate, the above training algorithm cannot be used, since the activation functions cannot be approximated by a first-order Taylor series. In this case, neither (9) nor (13) can be expressed in a simple form and thus the new network weights should be estimated through conventional minimization of (2) by applying, for example, the backpropagation algorithm. Since both the current and previous network training data sets are taken into account for updating the weights, overfitting issues are not encountered in this case too.

Each time the decision mechanism ascertains that retraining is required, a new training set S_c is created, which represents the current condition. Then, new network weights are estimated taking into account both the current information (data in S_c) and the former knowledge (data in S_b). Since the set S_c has been optimized over the current condition, it cannot be considered suitable for following or future states of the environment. This is due to the fact that data obtained from future states of the environment may be in conflict with data obtained from the current one. On the contrary, it is assumed that the training set S_b, which is in general provided by a vendor, is able to roughly approximate the desired network performance at any state of the environment. Consequently, in every network retraining phase, a new training set S_c is created and the previous one is discarded, while new weights are estimated based on the current set S_c and the old one S_b, which remains constant throughout network operation.

C. The Gradient Projection Method

The philosophy of the gradient projection method is, starting from a feasible point, to move in a direction which decreases E_S and simultaneously satisfies the constraints (10) and (12). Let us give some definitions first [25]. A point is called feasible when it satisfies all the constraints. An inequality is characterized as an active constraint when it is about to be violated at a feasible point, in the sense that it turns to equality at this point. Otherwise the inequality is an inactive constraint. For example, the inequality $\Delta \underline{w} \cdot \underline{a}(\underline{x}_i) \leq \phi(\underline{x}_i)$ in (12) becomes active at a feasible point $\Delta \underline{w}$ when $\Delta \underline{w} \cdot \underline{a}(\underline{x}_i) = \phi(\underline{x}_i)$ at this current

point. Otherwise it is considered to be inactive. In the following we denote as q_a the number of active inequalities. By convention we consider any equality constraint to be active at a feasible point. Thus, the total number of active constraints, i.e., equalities and active inequalities, is equal to $m_c + q_a$ (m_c equalities and q_a active inequalities).

Only the active constraints define the direction which causes a decrease of function E_S. As a result, a working set is formed at each feasible point which includes all active constraints. Then, the method tries to move in a direction which decreases the error function E_S while keeping the working constraints active. This is accomplished by projecting the negative gradient of E_S onto the subspace that is tangent to the surface defined by these constraints. If the resulting vector is nonzero, it determines the direction for the next step. To compute this projection, let matrix $\boldsymbol{A}_q$ be composed of the above-defined working set. Matrix $\boldsymbol{A}_q$ contains all rows of matrix $\boldsymbol{A}$ (since equalities are always active at any feasible point) and additional rows which include the active inequalities. Thus, $\boldsymbol{A}_q$ is of $(m_c+q_a)\times N_w$, where we recall that N_w corresponds to the total number of network weights. Let us denote by $\underline{h}(n)$ the projection of the negative gradient of E_S in the nth iteration step of the algorithm. Then, adaptation of the weight increments is made as follows:

$$\Delta\underline{w}(n+1) = \Delta\underline{w}(n) + \mu(n)\underline{h}(n) \tag{15}$$

where $\mu(n)$ is a scalar that determines the rate of convergence. Using the methodology of [25] we can estimate vector $\underline{h}(n)$ as

$$\underline{h}(n) = -\boldsymbol{P}\nabla E_S = -\boldsymbol{Q}\Delta\underline{w} \tag{16a}$$

with

$$\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{A}_q^T(\boldsymbol{A}_q\boldsymbol{A}_q^T)^{-1}\boldsymbol{A}_q \quad \text{and} \quad \boldsymbol{Q} = \boldsymbol{P}\boldsymbol{K}^T\boldsymbol{K}\Delta\underline{w} \tag{16b}$$

using ∇E_s computed from (14) at the nth iteration.

Starting from an initial feasible point, the gradient projection algorithm computes the next feasible point based on (15) until the projected gradient $\underline{h}(n)$ is close to zero, i.e., $\underline{h}(n) \approx 0$. In this case, however, it is possible to improve the solution (estimate a new feasible point that causes a greater decrease of E_S) by relaxing one active constraint, i.e., one active inequality. This happens since the Kuhn–Tucker conditions are not satisfied for the original problem [25]. A vector, say $\underline{\lambda}$, is calculated by the following equation:

$$\underline{\lambda} = -\left(\boldsymbol{A}_q \cdot \boldsymbol{A}_q^T\right)^{-1} \cdot \boldsymbol{A}_q \cdot \nabla E_s. \tag{17}$$

Each element of $\underline{\lambda}$, denoted as λ_i, corresponds to the ith active constraint of the working set [25]. In case that there are negative elements of $\underline{\lambda}$, the Kuhn–Tucker conditions are not satisfied and a new improved feasible solution can be obtained by relaxing the respective constraint from the working set. This is performed by deleting the corresponding row of $\boldsymbol{A}_q$, so that the respective constraint becomes inactive. In this case the projection matrix $\boldsymbol{P}$ should be recomputed since matrix $\boldsymbol{A}_q$ has modified. A new iteration then commences until the projected gradient $\underline{h}(n)$ reaches a value close to zero. If all elements of $\underline{\lambda}$ are nonnegative, the Kuhn–Tucker conditions are satisfied and the process is terminated since it is not possible to further improve the obtained solution. It should be mentioned that the times that the working set is changed are equal to number of inequalities since equalities are always considered active. Thus, the number of iterations required for getting all elements of $\underline{\lambda}$ positive so that the Kuhn–Tucker conditions are satisfied, is bounded by the number of inequalities.

The time required to adapt network weights is very crucial, especially in real-time applications. This is, for example, the case in video processing, where each video frame is transmitted every 40 (33) ms for the PAL (NTSC) system and thus network retraining should be completed within this small interframe period. The computational complexity required to update each network weight independently using the gradient projection method is proportional to the number of network weights N_w as shown next.

At each iteration of the algorithm, the inner product of matrix $\boldsymbol{P}$ with the gradient of function E_S is calculated (16a) along with the weight updating given by (15). Since function E_S is of squared form, the gradient $\nabla E_s = \boldsymbol{K}^T\boldsymbol{K}\Delta\underline{w}$ and thus (16a) involves a simple multiplication of vector $\Delta\underline{w}$ with matrix $\boldsymbol{Q} = \boldsymbol{P}\boldsymbol{K}^T\boldsymbol{K}\Delta\underline{w}$, which is of size $N_w \times N_w$. This requires $O(N_w)$ operations assuming that each element of $\underline{h}(n)$ can be implemented in parallel. It should be mentioned that matrix $\boldsymbol{Q}$, for a fixed number of working constraints is calculated once since $\boldsymbol{P}$ is modified each time the working constraints are changed and matrix $\boldsymbol{K}$, which depends on the previous network weights $\underline{w}_b$, is available before the algorithm begins. Furthermore, the projection matrix $\boldsymbol{P}$ does not need to be entirely recomputed each time the working constraints are modified. This is due to the fact that the active constraints in the working set change by one at a time and thus it is possible to recalculate matrix $\boldsymbol{P}$ from the previous one by an updating procedure as described in [25, p. 333]. This is a significant property of the gradient projection method and greatly reduces its computational load.

The convergence rate of the gradient projection method is of the same order with that of steepest descent, for a constant learning rate $\mu(n)$; i.e., it converges to the optimal solution linearly as can be shown using eigenvalue analysis in [25, p. 342]. Faster convergence rates can be achieved by selecting the value $\mu(n)$ to be the largest step toward the global minimum. This can be accomplished by minimizing the following equation with respect to learning rate $\mu(n)$:

$$\mu(n) = \underset{\mu(n)\in R}{\arg\min}\left\{E_S\left(\Delta\underline{w}(n) + \mu(n)\underline{h}(n)\right)\right\}. \tag{18}$$

After minimization, the optimal value of learning rate $\mu(n)$ is the following:

$$\mu(n) = -\frac{\underline{h}^T(n)\cdot\nabla E_S}{\underline{h}^T(n)\cdot\boldsymbol{K}^T\cdot\boldsymbol{K}\cdot\underline{h}(n)}. \tag{19}$$

The number of iterations required by the algorithm to converge is further restricted by the maximum permitted time, say T_D, in which retraining should be accomplished. For instance,

in case of on line classification of video frames, $T_D = 40$ ms for a PAL system, within which selection of the current training set S_c, weight adaptation procedure as well as network testing using the new weights should be completed. Since the time required for network retraining is significantly larger than the other times, we can assume in the following that the maximum permitted time for weight adaptation is approximately equal to T_D. Thus, the number of iterations of the gradient projection method should be smaller than T_D/T_s, where T_s indicates the average computational time for one iteration. If, after T_D/T_s iterations, the solution is not close to the optimal one, the currently estimated weights are used to perform classification. Weight updating will be continued in the following frames, for further improving the network performance. Similarly, in a large weight adaptation case, where a type of backpropagation algorithm is used for estimating the network weights, the number of iterations is also restricted by the maximum permitted time T_D. If a solution cannot be reached within T_D, processing moves to the following frames, assuming that no further retraining is required.

IV. Optimal Selection of the Network Retraining Data

Whenever the decision mechanism ascertains that network retraining is required, a training set S_c should be generated so that the retraining algorithm uses it to adapt the network weights to the current environment. A MAP estimation procedure is proposed next for constructing this training set S_c, i.e., for determining proper pairs $(\underline{x}_i, \underline{d}_i)$, with $\underline{x}_i$ denoting input image blocks of the examined image $\underline{x}$ and $\underline{d}_i$ respective desired outputs. Let $\underline{S}$ be a vector containing the probabilities that each image block belongs to one of the p available classes. Then, $\underline{S} = [\underline{s}(\underline{x}_1)^T \underline{s}(\underline{x}_2)^T \cdots \underline{s}(\underline{x}_L)^T]^T$ where L is the number of blocks of image $\underline{x}$ and $\underline{s}(\underline{x}_i) = [p^i_{\omega_1}\ p^i_{\omega_2} \cdots p^i_{\omega_p}]^T$. A high value $p^i_{\omega_j}$ indicates that it is highly probable that the respective image block $\underline{x}_i$ belongs to class ω_j and can therefore be included in the retraining set, being classified to category ω_j. Let also vectors $\underline{Z}$ and $\underline{Y}$ include the outputs of the retrained and initially trained networks, respectively, for all blocks $\underline{x}_i$ of image $\underline{x}$, i.e.,

$$\underline{Z} = \left[\underline{z}(\underline{x}_1)^T\ \underline{z}(\underline{x}_2)^T \cdots \underline{z}(\underline{x}_L)^T\right]^T$$

and

$$\underline{Y} = \left[\underline{y}(\underline{x}_1)^T\ \underline{y}(\underline{x}_2)^T \cdots \underline{y}(\underline{x}_L)^T\right]^T. \tag{20}$$

For a current image $\underline{x}$ and an output $\underline{Y}$ being provided by the initially trained network, our target is: 1) to estimate vector $\underline{S}$, and consequently the current training set S_c; 2) to compute the weights of the final network classifier $\underline{w}_a$ (through the retraining algorithm); and 3) to obtain the final classification output $\underline{Z}$ corresponding to image $\underline{x}$. This can be obtained through maximization of the following conditional likelihood function:

$$\left\{\hat{\underline{Z}}, \hat{\underline{w}}, \hat{\underline{S}}\right\} = \arg\max_{\underline{Z}, \underline{w}, \underline{S}} L(\underline{Z}, \underline{w}, \underline{S}/\underline{Y}, \underline{x}) \tag{21}$$

where L denotes a log-likelihood function. In (21) $\hat{\underline{w}}$ denotes the weights $\underline{w}_a$ provided by the retraining algorithm and $\hat{\underline{Z}}$ is the classification output of the retrained neural-network classifier which is applied to the input image $\underline{x}$.

It is straightforward to show that the corresponding conditional probability density can be written as follows:

$$\begin{aligned}&\Pr(\underline{Z}, \underline{w}, \underline{S}/\underline{Y})\\ &\quad = \Pr(\underline{S}/\underline{Y}) \Pr(\underline{Z}, \underline{w}/\underline{Y}, \underline{S})\\ &\quad = \Pr(\underline{S}/\underline{Y}) \Pr(\underline{w}/\underline{Y}, \underline{S}) \Pr(\underline{Z}/\underline{S}, \underline{w}, \underline{Y})\end{aligned} \tag{22}$$

where we have ignored the dependence on the current image $\underline{x}$, since it is involved in all equations.

Estimation of the network weights $\underline{w}$ is independent of vector $\underline{Y}$. Moreover, network output $\underline{Z}$ only depends on the values of the weights $\underline{w}$ (and the input image). Hence, it can be concluded that

$$\begin{aligned}&\max_{\underline{Z}, \underline{w}, \underline{S}} \Pr(\underline{Z}, \underline{w}, \underline{S}/\underline{Y})\\ &\quad = \max_{\underline{Z}, \underline{w}, \underline{S}} \{\Pr(\underline{S}/\underline{Y}) \Pr(\underline{w}/\underline{S}) \Pr(\underline{Z}/\underline{w})\}.\end{aligned} \tag{23}$$

Using (4), (5), and (7), it is easy to compute a nonlinear function $g(\cdot)$ that models the operation of the neural network. Then

$$\underline{Z} = g(\underline{w}, \underline{x}). \tag{24}$$

Thus, if the optimal weights $\hat{\underline{w}}$ have been estimated through network retraining, the final classification of current image $\underline{x}$ is unique. Consequently

$$\Pr(\underline{Z}/\underline{w}) = \begin{cases} 1, & \text{if } \underline{Z} \text{ fullfils (24)} \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

Equation (23) using (25) indicates that the optimal vector $\underline{S}$ and the output of the retrained classifier $\underline{Z}$ can be computed as follows:

$$\max_{\underline{Z}, \underline{w}, \underline{S}} \Pr(\underline{Z}, \underline{w}, \underline{S}/\underline{Y}) = \max_{\underline{w}, \underline{S}} \{\Pr(\underline{S}/\underline{Y})\ \Pr(\underline{w}/\underline{S})\}. \tag{26}$$

Optimal network weights $\hat{\underline{w}}$ are calculated by the retraining algorithm of Section III given a vector $\underline{S}$, i.e., the training set S_c. Thus, maximization of the latter term in (26) is achieved through computation of weights $\underline{w}_a$ by the retraining algorithm. Consequently, maximization of (26) is equivalent to estimation of $\underline{S}$ (and consequently S_c), based on the output $\underline{Y}$ of the initially trained neural classifier.

Thus (26) results in

$$\hat{\underline{S}} = \arg\max_{\underline{S}} \log \Pr(\underline{S}/\underline{Y}). \tag{27}$$

Using the Bayes formula, (27) can be written as follows:

$$\begin{aligned}\hat{\underline{S}} &= \arg\max_{\underline{S}} \{\log \Pr(\underline{S}/\underline{Y})\}\\ &= \arg\max_{\underline{S}} \{\log \Pr(\underline{Y}, \underline{S}) - \log \Pr(\underline{Y})\}\\ &= \arg\max_{\underline{S}} \{\log \Pr(\underline{Y}/\underline{S}) + \log \Pr(\underline{S}) - \log \Pr(\underline{Y})\}\\ &= \arg\min_{\underline{S}} \{-\log \Pr(\underline{Y}/\underline{S}) - \log \Pr(\underline{S})\}\end{aligned} \tag{28}$$

where $\Pr(\underline{Y})$ has been omitted from the log likelihood function, since it is independent of $\underline{S}$. Equation (28) indicates that in order to find $\hat{\underline{S}}$, we should compute both the conditional probability density $\Pr(\underline{Y}/\underline{S})$ and the probability density $\Pr(\underline{S})$.

Let us first assume that for a given vector $\underline{S}$, there is only one classification vector $\underline{Y}$ which is associated to $\underline{S}$ by a linear relation through a fixed matrix $\boldsymbol{D}$, or equivalently that different training sets $\underline{S}$ are estimated for different vectors $\underline{Y}$. As a result, the conditional probability density $\Pr(\underline{Y}/\underline{S})$ is given by

$$\Pr(\underline{Y}/\underline{S}) = \begin{cases} 1, & \text{if } \underline{Y} = \boldsymbol{D}\underline{S} \\ 0, & \text{if } \underline{Y} \neq \boldsymbol{D}\underline{S}. \end{cases} \tag{29}$$

The next step is to model probability density $\Pr(\underline{S})$. The elements of $\underline{S}$ represent the probabilities that an image block belongs to each of the p available classes. It is well known that image blocks $\underline{x}_i$ are strongly correlated to their spatial neighbors, due to smoothness of the objects appearing in images. Thus, the elements of $\underline{S}$ are also locally correlated, i.e., for image blocks $\underline{x}_i$ and $\underline{x}_j$ we have [5], [20]

$$\begin{aligned} &\Pr(\underline{S}(\underline{x}_i)/\underline{S}(\underline{x}_j), j \neq i) \\ &\quad = \Pr(\underline{S}(\underline{x}_i)/\underline{S}(\underline{x}_j), j \in \partial_i), \qquad \forall i \end{aligned} \tag{30}$$

where ∂_i denotes a neighborhood of the image block corresponding to $\underline{x}_i$. Such a property characterizes an MRF, hence a Gibbs distribution can be used for modeling $\Pr(\underline{S})$ [5], [15], [35]

$$\Pr(\underline{S}) = \Gamma \exp\left(-\frac{\sum_{c \in C} V_c(\underline{S})}{\gamma}\right) \tag{31}$$

where

- Γ normalizing constant;
- γ "temperature" parameter of the density;
- $V_c(\cdot)$ any function of a local group of points c called clique and C is the set of all cliques.

Using the MAP estimation procedure and based on (28) and (29) the estimate of $\underline{S}$ can be written as

$$\hat{\underline{S}} = \arg\min_{\underline{S}} \left\{ \frac{1}{\gamma} \sum_{c \in C} V_c(\underline{S}) \right\}. \tag{32}$$

Based on the previous equation, it is observed that estimation of the optimal vector $\hat{\underline{S}}$ is not affected by the value of temperature γ since the latter has the same influence on all directions of the clique. For this reason, we will consider, for simplicity, in the following that $\gamma = 1$.

As already mentioned, images are locally smooth [38]. Consequently, it is not probable that an image block belongs to a specific class, if all its neighbors belong to another one. Since retraining should be performed using data that have already been classified with maximum confidence, function $V_c(\cdot)$ should award image blocks that satisfy the smoothness property and discourage the rest. It should also be mentioned that the knowledge included in the initially trained network classifier should be trusted as much as possible within the above selection procedure; it is, therefore, desired that the finally computed classification is similar to the one originally provided by the classifier, modified according to the smoothness principle. Following the previous assumption, a common choice for function $V_c(\cdot)$ in a two-class classification problem, where $\underline{s}(\underline{x}_i)$ is a scalar denoted by s_i, is

$$\sum_{c \in C} V_c(\underline{S}) = \sum_{i=0}^{L-1} \sum_{l \in \partial_i} \rho(s_i - s_l) \tag{33a}$$

where measure ρ depends both on the distance of s_i from the corresponding initial classification y_i as well as on its distance from its neighbors s_l. In (33a), the neighborhood ∂_i of s_i, i.e., the clique structure, has been selected to be a 3×3 pixel grid. An equal-weight operator has been used for all eight directions of the clique, since the same training set should be produced irrespectively from the image orientation. For this reason, function $\rho(\cdot)$ in (33a) is given by

$$\rho(s_i - s_l) = (s_i - y_i)^2 + \tau(s_i - s_l)^2 \tag{33b}$$

where parameter τ controls the contribution of each of the two terms in the right-hand side of (33b) to the minimization of the cost function. In particular, values of τ close to zero result in a vector $\hat{\underline{S}}$ that approximately equals output $\underline{Y}$ obtained by the initially trained classifier. However, several misclassified blocks are encountered in $\underline{Y}$ as was mentioned in Section II. Since training set S_c, which is used for network retraining, follows the estimation of the optimal vector $\hat{\underline{S}}$, erroneous training elements will be presented in S_c for small values of τ. As a result, the network performance deteriorates after retraining. On the other hand, unnecessarily large values of τ stress the importance of the smoothness (second) term of (33b), resulting in many ambiguous blocks, i.e., blocks which do not present high probability of belonging to a specific class. In this case the set S_c cannot appropriately describe the current condition since it contains few training data, i.e., only those which are not ambiguous.

As parameter τ increases from small to large values, more and more misclassified blocks of $\underline{Y}$ are considered as ambiguous and are therefore discarded from S_c. However, very large values of τ reduce the representativity of S_c, since high confident blocks are also discarded apart from the misclassified ones. Assuming that the output $\underline{Y}$ contains $a\%$ misclassified blocks, the parameter τ is selected so that the training set S_c contains a slightly smaller number than 1-$a\%$ of the total number of image blocks; a slight deviation of τ around the selected value does not significantly affect network performance after retraining.

Function $\rho(\cdot)$ in (33b) satisfies the convexity property since it is of squared form [25]. Therefore, (33a) also defines a convex function since the sum of convex functions also yields a convex function (see [25, p. 178, Proposition 1]). Convexity guarantees that a global minimum can be obtained by minimizing (32). Otherwise, local minima would be present and a computationally expensive technique, such as simulated annealing [24], should be used for function minimization. Direct minimization of (32) leads to the linear relation between $\underline{Y}$ and $\underline{S}$ that has been assumed in (29). In (33a) and (33b) the value of s_i depends on the values of the eight connected neighbors s_l (located on the clique structure), which are also involved in the minimization process. For this reason, the iterative conditional modes (ICM's) technique [3] has been proposed to estimate the optimal vector $\hat{\underline{S}}$.

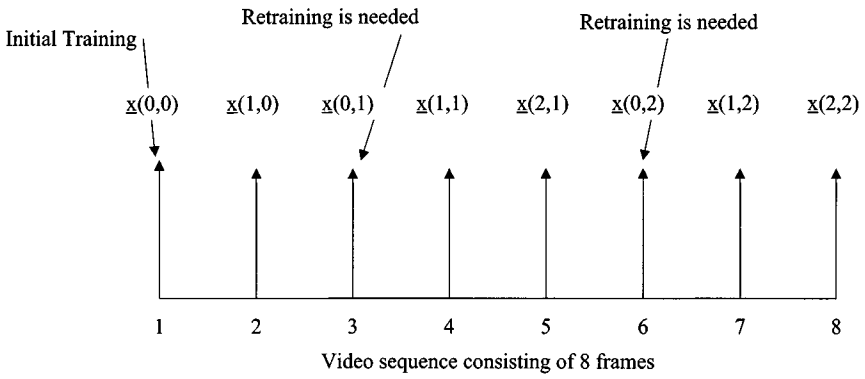

Fig. 3. A scenario of a video sequence consisting of eight frames in which retraining at frames three and six has been accomplished.

Starting from an initial, say zero, condition, the algorithm converges in a few steps to the minimum value of $\underline{S}$.

Following computation of optimal vector $\hat{\underline{S}}$, selection of the data set S_c for retraining the classifier can be performed as follows. Let us denote by I a set of indexes i, which correspond to elements of vector $\underline{S}$, say, s_i, which belong with high probability to one of the two classes

$$I = \{i\colon s_i > T_h \text{ or } s_i < T_l\} \tag{34}$$

where T_h and T_l are proper thresholds which define the regions of high probability. If we denote by i_k the kth element of set I, then the pairs $(\underline{x}_{i_k}, s_{i_k})$ are included in S_c.

Although in general the above described method converges to the optimal value after a small number of iterations, in case of real-time applications, where the amount of computational cost is very crucial, alternative techniques can be used for selection of the training set S_c. A fast technique, which yields a suboptimal but good selection of training data, is based on the binary form of the mask $\underline{Y}$ of the initially trained network. This form is derived by thresholding the probability values of (1) for all image blocks. Then, a block is selected as training one, if all its eight neighbors belong to the same class with it. In this way, ambiguous blocks are discarded from the training set and only the most confident ones are selected for training, through a very fast procedure.

V. The Decision Mechanism for Network Retraining

The purpose of this mechanism is to detect when the output of the neural-network classifier is not appropriate and consequently to activate the retraining algorithm at those time instances when a change of the environment occurs.

Let us index images or video frames in time, denoting by $\underline{x}(k, N)$ the kth image or image frame following the image at which the Nth network retraining occurred. Index k is therefore reset each time retraining takes place, with $\underline{x}(0, N)$ corresponding to the image where the Nth retraining of the network was accomplished. Fig. 3 indicates a scenario with two retraining phases (at frames three and six, respectively, of a video sequence composed of eight frames) and the corresponding values of indexes k and N. It can been seen that $\underline{x}(0, N+1) = \underline{x}(k_0, N)$ where k_0 indicates that after k_0 images from the Nth retraining phase, a new retraining phase, i.e., the $(N+1)$th takes place.

Retraining of the network classifier is accomplished at time instances where its performance deteriorates, i.e., the current network output deviates from the desired one. Let us recall that vector $\underline{c}$ in (11) expresses the difference between the desired and the actual network outputs based on weights $\underline{w}_b$ and applied to the current data set S_c. As a result, if the norm of vector $\underline{c}$ increases, network performance deviates from the desired one and retraining should be applied. On the contrary, if vector $\underline{c}$ takes small values, then no retraining is required. In the following we denote this vector as $\underline{c}(k, N)$ indicating its dependence upon image $\underline{x}(k, N)$. However, direct calculation of the norm of $\underline{c}(k, N)$ would require application of the MAP estimation procedure described in the previous section so as to extract S_c from each image $\underline{x}(k, N)$. In video processing applications, for example, each image frame arrives at a rate of 40 ms (25 frames/s in PAL system); it will therefore be very time consuming to activate the MAP estimation procedure for each $\underline{x}(k, N)$. In such applications, detection of the retraining time instances can be performed through the following methodology.

A. Detection of the Retraining Time Instances

Let us assume that the Nth retraining phase of the network classifier has been completed. If the classifier is then applied to all blocks of the image $\underline{x}(0, N)$, including the ones used for retraining, it is expected to provide classification results of good quality. The difference between the output of the retrained network and of that produced by the initially trained classifier at image $\underline{x}(0, N)$ constitutes an estimate of the level of improvement that can be achieved by the retraining procedure. Let us denote by $e(0, N)$ this difference, which is computed as follows:

$$e(0, N) = \frac{1}{L}\sum_{i=1}^{L} \left\{\underline{y}(\underline{x}_i(0, N)) - \underline{z}(\underline{x}_i(0, N))\right\}^T \cdot \left\{\underline{y}(\underline{x}_i(0, N)) - \underline{z}(\underline{x}_i(0, N))\right\} \tag{35}$$

where L is the number of blocks in the image; dependence of vector $\underline{z}$ on the current network weights $\underline{w}_a$ has been omitted for simplicity.

Let $e(k, N)$ denote the difference between the corresponding classification outputs, when the two networks are applied to the

kth image or image frame following the Nth network retraining phase, for $k = 1, 2, 8, \cdots$

$$e(k, N) = \frac{1}{L}\sum_{i=1}^{L}\left\{\underline{y}(\underline{x}_i(k, N)) - \underline{z}(\underline{x}_i(k, N))\right\}^T \cdot \left\{\underline{y}(\underline{x}_i(k, N)) - \underline{z}(\underline{x}_i(k, N))\right\}. \quad (36)$$

It is anticipated that the level of improvement expressed by $e(k, N)$ will be close to that of $e(0, N)$ as long as the classification results are good. This will occur when input images are similar, or belong to the same scene with the ones used during the retraining phase. An error $e(k, N)$, which is quite different from $e(0, N)$, is generally due to a change of the environment. Thus, the quantity $a(k, N) = |e(k, N) - e(0, N)|$ can be used for detecting the change of the environment or equivalently the time instances where retraining should occur. Thus

$$\text{if } a(k, N) < T \text{ no retraining is needed} \quad (37)$$

where T is a threshold which expresses the maximum tolerance, beyond which retraining is required for improving the network performance. In case of retraining, index k is reset to zero while index N is incremented by one.

Such an approach detects with high accuracy the retraining time instances both in cases of abrupt and gradual changes of the operational environment since the comparison is performed between the current error difference $e(k, N)$ and the one obtained right after retraining, i.e., $e(0, N)$. In an abrupt operational change, error $e(k, N)$ will not be close to $e(0, N)$; consequently, $a(k, N)$ exceeds threshold T and retraining is activated. In case of a gradual change, error $e(k, N)$ will gradually deviate from $e(0, N)$ so that the quantity $a(k, N)$ gradually increases and retraining is activated at the frame where $a(k, N) > T$.

The norm of vector $\underline{c}(k, N)$ in (11) is not directly involved in (37). Since $\underline{z}$ in (36) corresponds to $\underline{z}_b$ in (11) and d's are replaced by y's, it can be concluded that, (36) uses the output provided by the initially trained network as an approximation of the desired output in each new image, or frame. For detection purposes $a(k, N)$ and the norm of $\underline{c}(k, N)$ appear to have similar behavior and properties and thus $a(k, N)$ provides a quick way for determining the retraining time instances without requiring to activate the MAP estimation procedure at every image frame $\underline{x}(k, N)$.

Network retraining can be instantaneously executed each time the system is put in operation by the user by utilizing zero initial values for the output mask $\underline{Z}$. Thus, the quantity $a(0, 0)$ initially exceeds threshold T and retraining is forced to take place.

In some, rather rare, cases, the aforementioned mechanism could detect no need for retraining although it should. This is, for example, the case when a significant amount of image blocks lie near class boundaries, and consequently there is low probability that they belong to a specific class. Thus, despite a possible large change in the classification of the image blocks, due to moving boundaries, there is only a small variation of the block probabilities. In this case, the decision mechanism detects no need for retraining since only a small perturbation of the block probabilities has occurred. To avoid such situations, the binary forms of masks $\underline{Y}$ and $\underline{Z}$ can be used in (35) and (36) as an additional or alternative mechanism for detecting the need for retraining. This is due to the fact that masks $\underline{Y}$ and $\underline{Z}$, in binary form, suffer large variation for any change of the environment; thus, they cause a large variation of the difference $a(k, N)$ and as a consequence, retraining is activated. Other measures that can be also used for detecting the need of retraining are the comparison of the spatial distribution of classified blocks to that of each class as well as of their number with respect to total number of blocks in the image.

B. The Retraining Phase

When $a(k, N)$ exceeds threshold T for some k, say k_0, the $(N + 1)$th retraining phase starts. In this case, the MAP estimation procedure is activated so as to create the training set S_c which represents the current condition; vector $\underline{c}(k_0, N) \equiv \underline{c}(0, N + 1)$ is also calculated. The weights of the network before retraining are used as previous network weights $\underline{w}_b$ in the retraining algorithm.

The training algorithm provides the new network weights through minimization of (14) subject to constraints (10) and (12). However, existence of a feasible solution is assured only if the hyper plane given by (10) crosses through the boundary constraints imposed by (12). For this reason the minimal distance from the origin to the surface $\underline{c}(0, N + 1) - \boldsymbol{A} \cdot \Delta\underline{w} = \underline{0}$, is first calculated, by solving the following minimization problem with respect to $\Delta\underline{w}$

minimize

$$\|\Delta\underline{w}\|_2 \text{ or equivalently } (\Delta\underline{w})^T \cdot \Delta\underline{w}$$

subject to

$$\underline{c}(0, N + 1) - \boldsymbol{A} \cdot \Delta\underline{w} = \underline{0}. \quad (38)$$

Using Lagrange multipliers the above minimization problem leads to the following solution:

$$\Delta\underline{w}^p = \boldsymbol{A}^T \cdot (\boldsymbol{A} \cdot \boldsymbol{A}^T)^{-1} \cdot \underline{c}(0, N+1) = \boldsymbol{Q} \cdot \underline{c}(0, N+1) h, v, j$$

with

$$\boldsymbol{Q} \equiv \boldsymbol{A}^T \cdot (\boldsymbol{A} \cdot \boldsymbol{A}^T)^{-1} h, v, j \quad (39)$$

where $\Delta\underline{w}^p$ is the minimal distance from the origin to $\underline{c}(0, N + 1) - \boldsymbol{A} \cdot \Delta\underline{w} = \underline{0}$. For this value of $\Delta\underline{w}^p$, if the inequality constraints defined by (12) are satisfied for all data in S_c, the new network weights can be calculated by minimizing (14) subject to (10) and (12). Otherwise, (2) cannot be expressed in the simple form of (10) and (14) and thus direct minimization of (2) is required for estimating the network weights. This is due to the fact that the former knowledge is quite different from the current one and a small weight perturbation is inadequate for sufficiently adapting the network weights.

Another situation where the above can be used is the following. Let us assume that several neural-network classifiers along with their respective training sets and weights have been given to a user by different suppliers or that the user has created different retraining sets himself using the procedure presented in the paper; the user is assumed to apply one of them. In cases that the decision mechanism detects that retraining is needed, but the training algorithm cannot be applied to the users' classification problem since the boundary constraints are not satisfied, it seeks, among the available networks, the most appro-

TABLE I

Initial Phase
Start from a feasible point $\Delta\underline{w}(0)$ and set $n=0$.
1. Find the active constraints for the point $\Delta\underline{w}(n)$ and form matrix $\mathbf{A}_q$.
2. Calculate the projection matrix $\mathbf{P}$ based on (16b) and then the projected gradient $\underline{h}(n)$ (16a).
3. If $\underline{h}(n)\neq 0$ compute $\Delta\underline{w}(n+1)$ using (15), set $n:=n+1$ and go to step 1.
4. If $\underline{h}(n)=0$ compute vector $\underline{\lambda}$ based on (17).
4a. If all elements of vector $\underline{\lambda}$ corresponding to the active inequalities are nonnegative, the process is terminated.
4b. Otherwise, drop the active inequality with the most negative component of $\underline{\lambda}$ from the working set, delete the respective row of matrix $\mathbf{A}_q$ and go to step 2.

TABLE II

Set all elements of vector $\underline{S}$ equal to zero.
1. Find the optimal $\hat{\underline{S}}$ by minimizing the convex function (32) through a gradient-based algorithm.
2. Form the set of indices I using (34).
Create the training set S_c.

TABLE III

1. For the kth image after the Nth retraining, $\underline{x}(k,N)$ do
1. Compute $a(k,N)=\lvert e(k,N)-e(0,N)\rvert$ using (35) and (36)
2. If (37) is satisfied then
set $k:=k+1$ and go to step 2.
4. Else do
4.1. Set index $k:=0$ and $N:=N+1$.
4.1. Using the steps of Table 2, compute the training set S_c for the current image $\underline{x}(k,N)$.
4.2. Using (39) calculate $\Delta\underline{w}^P$.
4.3. If the inequalities in (12) are satisfied at point $\Delta\underline{w}^P$ then
4.3.1. Using the algorithm described in Table 1 compute the weight increments $\Delta\underline{w}$.
4.3.2. Compute the weights after retraining, as $\mathbf{W}_a^0=\mathbf{W}_b^0+\Delta\mathbf{W}^0$ and $\underline{w}_a^1=\underline{w}_b^1+\Delta\underline{w}^1$.
4.3.3. Set $k:=1$ and go to step 2.
4.4. Otherwise, estimate the new network weights by direct minimization of (2).

priate one for the current environment. Then, the selected network is considered to represent the best previous knowledge and the training algorithm is used to further improve the network performance. The inner steps of the method, i.e., the retraining algorithm, the selection of retraining data and the decision mechanism are given in algorithmic form in Tables I–III of this paper.

VI. Experimental Study

In the following, the performance of the proposed scheme for on line neural network retraining was examined for detection and extraction of humans, and particularly of the upper part of human body including the head, shoulders and arms areas, in images or video sequences. Such an extraction plays an important role in many image analysis problems. Examples include retrieval of images or video sequences containing humans from image data bases [9], [11], [13] low bit rate coding of image sequences for videophone and videoconferencing applications [8], [10], [36], video surveillance of specific areas, such as city centers, for identifying suspects for crimes [14], as well as analysis by synthesis methods, where 2-D or 3-D modeling follows the extraction of human bodies from scenes. Furthermore, human extraction from background has recently attracted a great

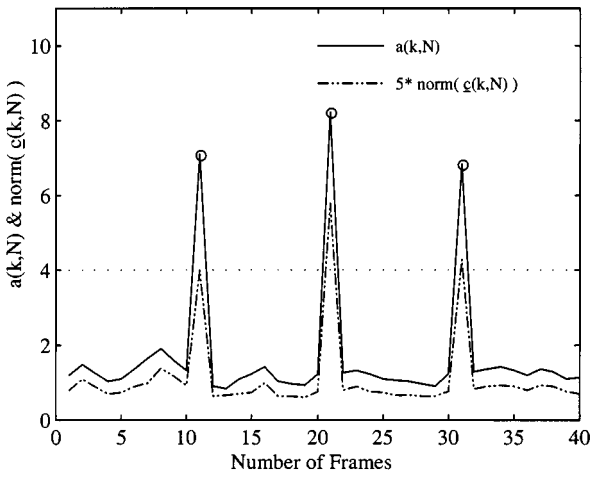

Fig. 4. The error-fuction $a(k, N)$ computed by the decision mechanism (solid line) along with the threshold T beyond of which retraining is required (dashed line). For comparison the $\|\underline{c}(k, N)\|_2$ is also shown in the figure.

research interest especially in the framework of MPEG-4 and MPEG-7 standards for content-based video coding/representation and content-based visual query in image and video.

An image scene was formed consisting of 40 frames from three different color videophone sequences, Claire, Trevor, and Miss America. In particular the sequence consisted of ten frames of Claire, followed by ten frames of Claire with their luminosity changed, followed by ten frames of Trevor and by ten frames of Miss America. Thus, the image scene included three changes of the environment. Several characteristic data of video conference applications has been used to train the initial neural-network classifier.

The format of the three-color sequences was QCIF, i.e., each image frame consisted of 144 × 176 pixels per color image component. The latter were separated in image blocks consisting of 8 × 8 pixels, resulting in L = 396 blocks per component. The target was to classify each image block into one of two classes, namely foreground objects, i.e., upper part of human body, and background. The dc coefficient and the first 8 ac coefficients of the zig-zag scanned DCT transform of each color component of the ith block, i.e., 27 elements in total were used as input $\underline{x}_i$ to the network, which consisted of q = 15 units in the hidden layer and one output unit as described in Section III-A. This results in 450 network weights, apart from the biases. The S_b set contains approximately 1000 image blocks as training elements.

Fig. 4 shows, $a(k, N)$, computed by the proposed decision mechanism using (35), (36), and the norm of $\underline{c}(k, N)$ estimated by applying the MAP procedure to each image frame. For clarity of presentation we have multiplied the values of $\|\underline{c}(k, N)\|_2$ by the factor of five. Similar behavior of the value of $\|\underline{c}(k, N)\|_2$ and $a(k, N)$ is observed. It can be seen that using a constant threshold of 4% the proposed mechanism was able to correctly detect whether retraining was required or not. After network initialization, where retraining has been activated on the first frame of Claire, the $a(k, N)$ was below the value of 4% in the following frames and therefore no retraining was needed. On the other hand, at the first frame of the Claire sequence with a luminosity change, the value of $a(k, N)$ exceeded the threshold and the decision mechanism activated the retraining algorithm for adapting the network weights to the current condition. The same happened at the first frames of Trevor and Miss America.

The retraining algorithm, described in Section III, was used to adapt the network weights in all the above changes of the scene. Fig. 5 shows the values of the 27 network weights connecting 1) the first hidden and 2) the tenth hidden neuron to the network input computed in the case of Claire before and after the luminosity change. The procedure described in Section IV was used for optimally selecting the retraining data sets and providing them to the training algorithm. In all the above cases the inequality constraints imposed by (12) were satisfied. The relative tolerance error of the network output was chosen to be less than 10%. Since, the desired output for all training data in S_c, corresponding to foreground (background), was close to 1 (−1), the bounds in (12) were almost equal, for all training inputs $\underline{x}_i$. For the selected relative error, ϕ was equal to 0.5, while ϑ was calculated based on (A7) and (A12) and was equal to 0.38 for the first frame of Claire sequence with luminosity change. Fig. 6 illustrates the inner product of (A13) for the weights connecting the first and the tenth hidden neurons to the input layer, for all input vectors $\underline{x}_i$ in S_c.

The next figures refer to the quality of classification achieved by the proposed system. Fig. 7(a) shows a characteristic frame of the Claire sequence, while Fig. 7(b) the first frame of Claire with a luminosity change, where retraining was performed. Fig. 7(c) shows the classification output provided by the network after retraining (final classification). The output is shown in the form of a map, in which all blocks classified to belong to background are, for clarity of presentation, marked with black color. It is observed that, after retraining, the network correctly classifies the image blocks into the two categories. Fig. 7(d) illustrates the network output before retraining. It is clear that several blocks, mainly in Claire's body, have been misclassified due to the luminosity change. The output of the MAP estimation procedure, which was used for selecting the retraining data set is presented in Fig. 7(e) in continuous form. In this case the parameter τ has been selected so that the S_c set contains about 70% of the total number of image blocks; its value was $\tau = 3.9$. This is due to the fact that it is assumed that the initially trained network provides an approximation slightly greater than 70% for the final classification (30% misclassification). Dark blocks indicate areas of background, white blocks indicate foreground areas while gray ones indicate ambiguous areas. This mask has been generated by minimizing (33a) and (33b) with respect to vector $\underline{S}$. A 3 × 3 pixel grid with equal weights in all eight directions has been selected to form the clique structure, as was mentioned in Section IV. Fig. 7(f) also shows the selected training blocks, in discrete form; black color denotes all blocks that were not selected as training data. Since there is a large number of similar training blocks, especially in the background, a distance measure was used to reduce their number before using them to define the constraints in (10). The number of selected blocks was reduced, in this way, from 198 to ten.

Fig. 8 refers to network retraining when fed with the first frame of the Trevor sequence, shown in Fig. 8(a). Fig. 8(b) shows the output of the initial network, which provided a coarse approximation of the desired classification. The effect of parameter τ on selecting the training set S_c is illustrated in Fig. 9. In

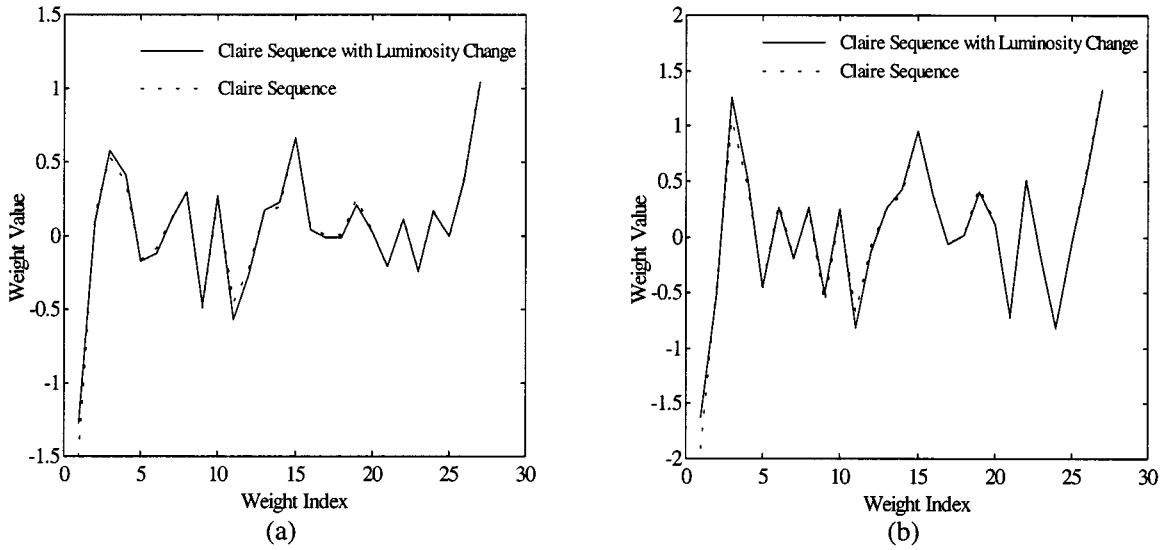

Fig. 5. Weight adaptation by the training algorithm in case of the Claire sequence with luminosity change. The weights between the input layer and (a) the first hidden neuron, i.e., $\underline{w}_1^0$ and (b) the tenth hidden neuron i.e., $\underline{w}_{10}^0$.

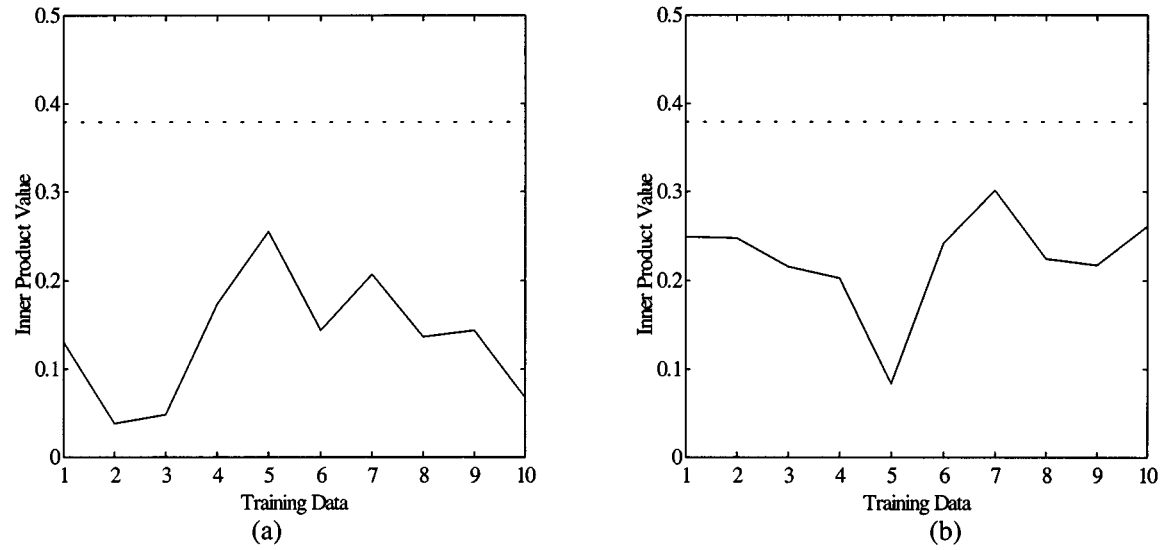

Fig. 6. The inner product of network weight perturbation, corresponding to the weights connecting the input layer to (a) the first and (b) the tenth hidden neuron, respectively, for all input vectors of the training set S_c in case of Claire sequence with luminosity change (solid line). The bound derived from (A7) and (A12) is illustrated in dashed line.

this figure the selected foreground blocks are depicted for two extreme values of τ; τ = 0.5, and τ = 22. As is observed, when τ is close to zero, erroneous training data are estimated. Instead, large values of τ result in a small training set, which is generally inadequate to represent with high accuracy the current image. Fig. 10 presents the percentage of the selected training data in S_c to the total number of image blocks versus parameter τ. It can been seen that the number of selected blocks to that of the total image decreases as the value of τ increases. Having assumed that the initially trained neural network approximates the desired classification by about 70%, the parameter τ is selected so that the percentage of the selected data is 70% of the total blocks. Fig. 8(c) shows the output of the MAP estimation technique in a form similar to that of Fig. 7(e) using the value of τ selected from Fig. 10 (τ = 4.92). Fig. 8(d) shows the training set selected by the MAP estimation procedure in this case. Moreover, the convergence of the MAP estimation procedure is shown in Fig. 11, in terms of the number $\Delta\underline{S}$ of blocks moved from one class to the other in consecutive iterations. It can be easily seen that, starting from a zero initial value of $\underline{S}$, the algorithm converges within five iterations to the global minimum of the cost function (33a) and (33b). Fig. 8(e) presents the network output after retraining, which shows that correct classification has been achieved.

In the following, a video conference application showing a scene of a conference room with three people talking is examined. In this case, since the scene can be considered as superposition of three video frames, each of which presents a single person, the initially trained neural-network classifier will continue to provide satisfactory results. After network retraining, extraction of foreground objects from background is very satisfactorily accomplished. The first frame of the respective se-

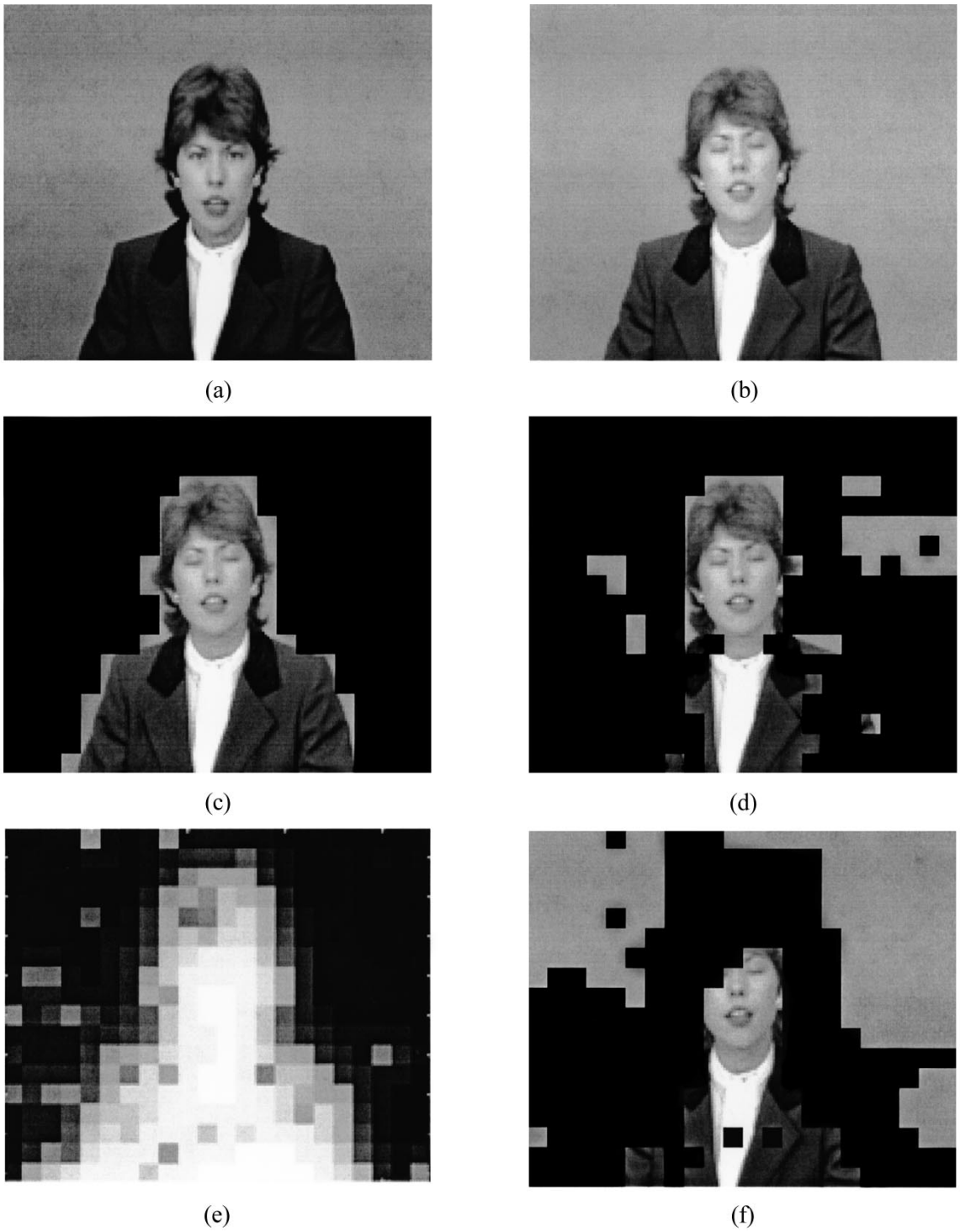

Fig. 7. (a) A characteristic frame of Claire, (b) the first frame of Claire with luminosity change, (c) the classification output, in discrete form, after network retraining, (d) the network output before retraining, for the same frame, in discrete form, (e) the training set S_c selected for retraining, in continuous form, i.e., the output of the MAP estimation procedure, set S_c, and (f) in discrete form.

quence is depicted in Fig. 12(a). Fig. 12(b) shows the respective training set in binary form, while Fig. 12(c) the final classification results.

As far as the problem of foreground–background separation is concerned, the proposed neural-network architecture provides very accurate results regardless of the luminosity conditions, foreground–background location/orientation and the respective color characteristics as Figs. 7 and 8 indicate. Segmentation techniques based on spatial and/or texture homogeneity criteria have been also proposed in the literature for this purpose [6], [27], [34], [37]. However, such approaches cannot provide accurate foreground–background separation since, in general, a person in a scene, contains regions with different color and texture characteristics (e.g., head, hair, clothes' color) which are

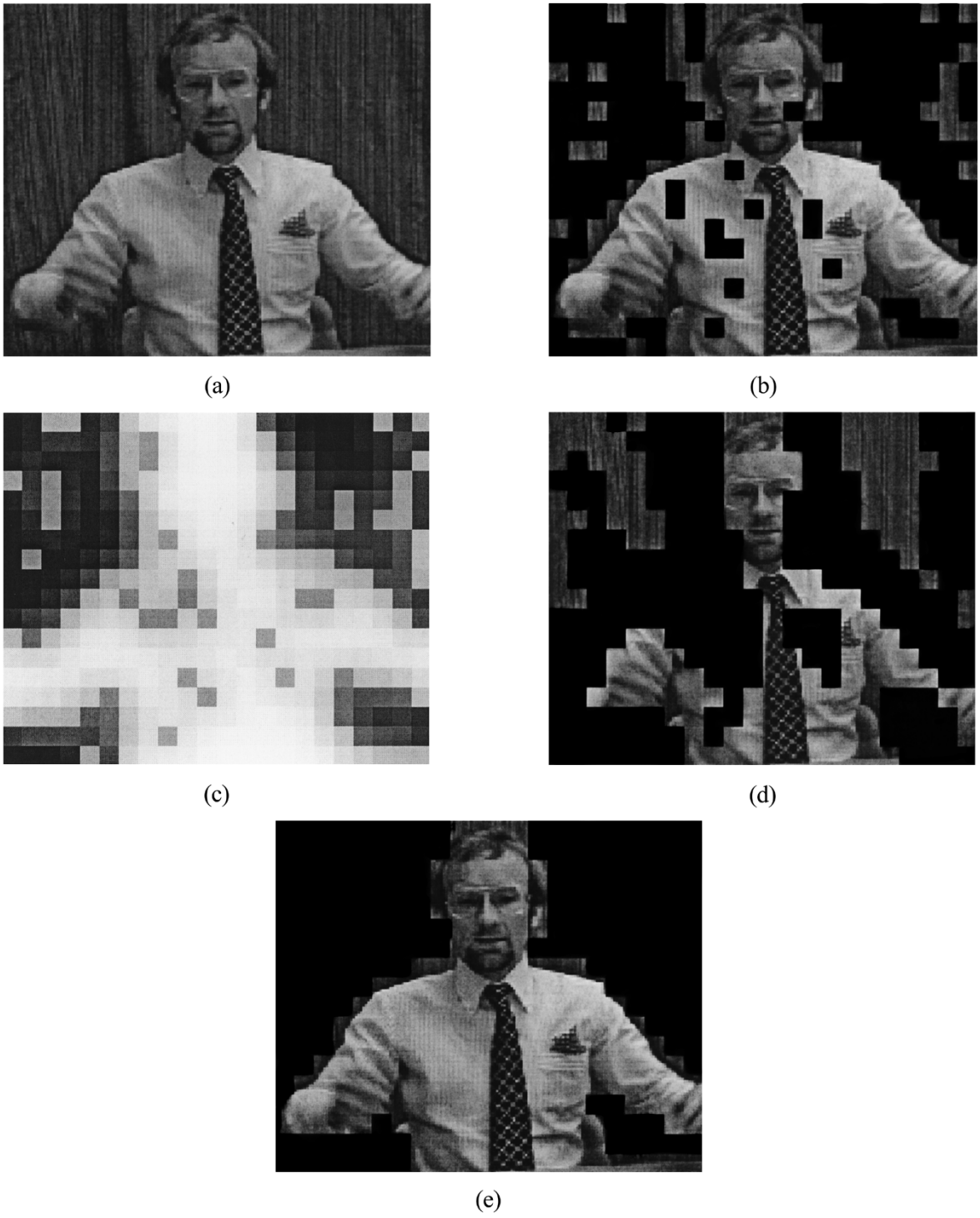

Fig. 8. Network performance at the first frame of Trevor sequence: (a) the original image, (b) the output of initially trained network, (c) the training set S_c selected for retraining in continuous form, (d) the training set in discrete form, and (e) the final classification output in discrete form.

classified to different segments (objects) according to such homogeneity criteria. For example, division of a square image into four equal-sized square blocks (quadtree decomposition), according to the texture homogeneity of the block, results in many misclassified blocks both in background and foreground areas [10]. Recently, other approaches have been proposed in the literature, which combine (fuse) several image properties according to predefined rules in order to provide more semantic video object extraction [2], [26]. However, these methods are restricted to the specific applications and thus they cannot be applied to different operational conditions.

VII. Conclusions

This paper has discussed on-line retraining of neural networks, focusing on image and video analysis applications. A training al-

Fig. 9. The selected foreground blocks for different values of parameter τ: (a) τ = 0.5 and (b) τ = 22.0.

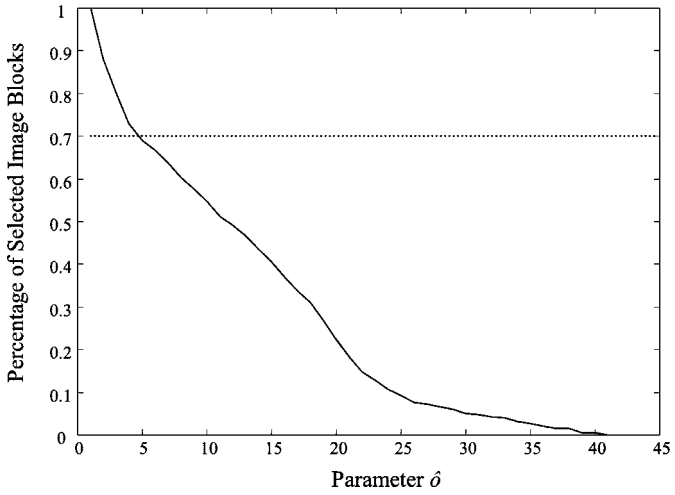

Fig. 10. Variation of the percentage of selected image blocks versus parameter τ.

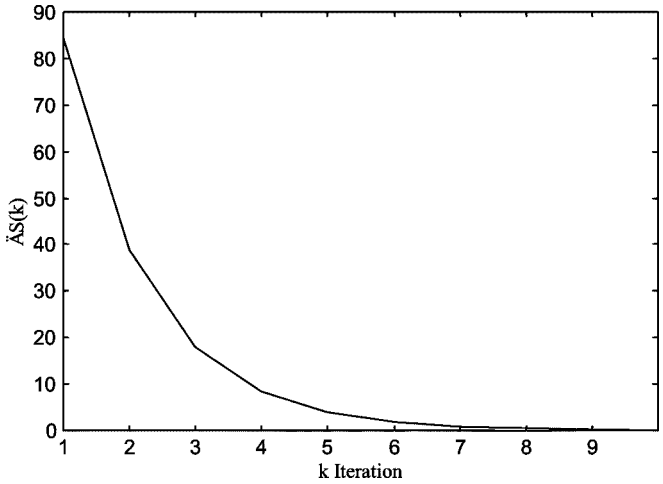

Fig. 11. Convergence of the MAP estimation procedure at the first frame of Trevor.

gorithm has been presented, which can efficiently handle cases of small variations of the operational environment, while a maximum *a posteriori* estimation technique has been developed which optimally selects the retraining data set from the image presented to the network. This is accomplished by modeling the image as an MRF. A decision mechanism has also been introduced which automatically activates network retraining whenever the network performance is not considered satisfactory.

The presented results refer to extraction of foreground areas from background ones in image/video classification problems. These results indicate the ability of the method to be successfully applied to a variety of image analysis applications, where neural-network-based classification constitutes a possible solution, especially when considering the forthcoming MPEG standards. This constitutes a topic, which is currently further investigated. It should be mentioned that the proposed neural-network architecture can be used to a variety of other applications. Examples include prediction of video traffic over high-speed networks where the traffic characteristics vary from time to time according to the scene complexity or nonlinear system identification where the system to be identified is changing with time. As a result, the proposed scheme can be viewed as a method for improving the performance of neural networks in dynamically changed environments.

Of particular interest is the interweaving of MAP estimation algorithms with neural-network learning optimization algorithms; in the resulting scheme optimal selection of the network training set is performed by the former algorithms and optimal output estimation is provided by the neural network. Investigation of convergence and performance of a general block component estimation method, which iteratively uses these two procedures, is another topic, which is under investigation.

APPENDIX A

As was described in Section III, network weights $\underline{w}_b$, $\underline{w}_a$, before and after retraining, respectively, are related through the following equation:

$$\underline{w}_a = \underline{w}_b + \Delta\underline{w} \tag{A1}$$

where $\Delta\underline{w}$ denotes a small perturbation.

Using (6), the output of the ith neuron of the hidden layer, after retraining, is given by

$$u_{i,a} = f\left(\left(\underline{w}^0_{i,a}\right)^T \cdot \underline{x}\right) \tag{A2}$$

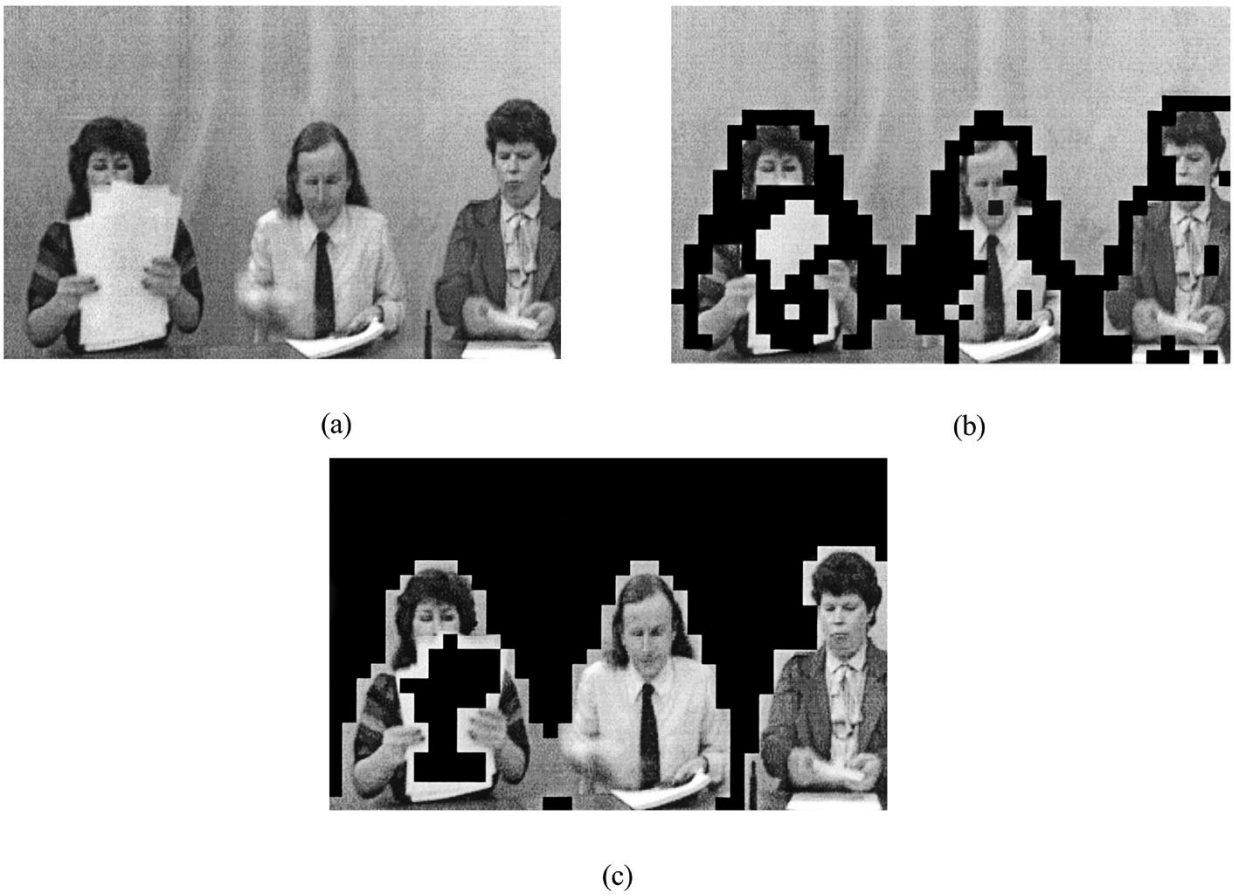

Fig. 12. Network performance for a video scene indicating a conference room with three people talking: (a) the original image, (b) the training set selected in discrete form, and (c) the final classification mask.

omitting, for simplicity, subscript j from $\underline{x}_j$ as well as the dependence of $u_{i,a}$ on $\underline{x}_j$.

From (A2) it can be seen that $u_{i,a}$ depends only on the inner product $(\underline{w}_{i,a}^0)^T \cdot \underline{x}$; setting $\beta_{i,a} = (w_{i,a}^0)^T \cdot \underline{x}$, (A2) takes the form

$$u_{i,a} = f(\beta_{i,a}). \tag{A3}$$

Combining (A1) and (A3) and using the formula of first-order Taylor series expansion we have

$$\begin{aligned} u_{i,a} &= f(\beta_{i,b} + \Delta\beta_i) \\ &= f(\beta_{i,b}) + f'(\beta_{i,b})\Delta\beta_i + R_i \\ &= u_{i,b} + \Delta u_i + R_i \end{aligned} \tag{A4}$$

where $u_{i,b}$ is the output of the ith hidden neuron before the retraining phase, i.e., $u_{i,b} = f(\beta_{i,b})$ with $\beta_{i,b} = (w_{i,b}^0)^T \cdot \underline{x}$, and Δu_i is a small perturbation, $\Delta u_i = f'(\beta_{i,b})\Delta\beta_i$, with $\Delta\beta_i = (\Delta w_i^0, b)^T \cdot \underline{x}$. Function $f'(\cdot)$ in (A4) denotes the first derivative of $f(\cdot)$, while R_i represents the first-order Taylor residual for the ith hidden neuron. Using the Lagrange formula, R_i can be expressed as

$$R_i = \frac{f''(\xi_i)\Delta\beta_i^2}{2!} \tag{A5}$$

where

$f''(\cdot)$ second derivative of $f(\cdot)$;

ξ_i scalar taking values between $\beta_{i,b}$ and $\beta_{i,b} + \Delta\beta_i$.

In general the absolute value of R_i in (A5) should be bounded, so that the error of the first-order Taylor series be small, i.e.,

$$|R_i| \le e_i \tag{A6}$$

where e_i is the maximum error of the residual R_i, that is allowed.

Based on (A6) and using the Lagrange formula (A5), the inner product $\Delta\beta_i$ can be bounded as follows:

$$|\Delta\beta_i| \le \left(\frac{2e_i}{|f''(\xi_i)|}\right)^{1/2} \equiv \vartheta_i. \tag{A7}$$

The value of $f''(\xi_i)$ in (A7) is difficult to be calculated, since scalar ξ_i is unknown. Estimation of $f''(\xi_i)$ can be performed by considering that the second derivative of $f(\cdot)$ is bounded and by letting $f''(\xi_i)$ be equal to its maximum value, $f''(\xi_i) \to \mu_2$, where $\mu_2 = \max f''(\cdot)$. From (A7) we get two linear inequalities for $\Delta\underline{w}_i^0$ so that the first-order Taylor series expansion be valid with an error less than ε_i, namely

$$-\vartheta_i \le (\Delta\underline{w}_i^0)^T \cdot \underline{x} \le \vartheta_i, \qquad i = 1, 2, \cdots q.$$

Using (4) and ignoring the residual errors R_i in (A4), we can express the network output z_a, after retraining, as follows:

$$\begin{aligned} z_a &= f\left(\left(\underline{w}_b^1\right)^T \cdot \underline{u}_b + \left(\underline{w}_b^1\right)^T \right. \\ &\qquad \left. \cdot\ \Delta\underline{u} + \Delta\underline{w}^1 \cdot \underline{u}_b + \Delta\underline{w}^1 \cdot \Delta\underline{u}\right) \\ &\approx f\left(\left(\underline{w}_b^1\right)^T \cdot \underline{u}_b + \left(\underline{w}_b^1\right)^T \cdot \Delta\underline{u} + \Delta\underline{w}^1 \cdot \underline{u}_b\right) \end{aligned} \tag{A8}$$

where the term $\Delta\underline{w}^1 \cdot \Delta\underline{u}$ has been ignored since it is very small compared to the other terms. Let us recall that $\underline{u}_b$, $\underline{u}_a$ are vectors defined in (6), while $\Delta\underline{u} = [\Delta u_1 \Delta u_2 \cdots \Delta u_q]^T$ is defined as $\underline{u}_a = \underline{u}_b + \Delta\underline{u}$. Expanding (A8) as a first-order Taylor series we get

$$\begin{aligned} z_a &= f(\delta_a) \\ &= f(\delta_b) + f'(\delta_b)\Delta\delta + R \\ &= z_b + \Delta z + R \end{aligned} \tag{A9}$$

where we have set $\delta_{\{a,b\}} = (\underline{w}_{\{a,b\}}^1)^T \cdot \underline{u}_{\{a,b\}}$ and $\Delta\delta = (\underline{w}_b^1)^T \cdot \Delta\underline{u} + \Delta\underline{w}^1 \cdot \underline{u}_b$. In (A9), R represents the residual error of the first-order Taylor series approximation which can be expressed similarly to (A5). Requiring that the residual R is smaller than a relative error provided by the user, say, $\varepsilon_1|z_a|$, the following inequality should be satisfied:

$$|\Delta\delta| \le \left(\frac{2\varepsilon_1 |z_a|}{f''(\xi)}\right)^{1/2} \equiv \phi. \tag{A10}$$

It should be mentioned that the value of $|z_a|$ is known, being estimated by the MAP estimation procedure.

Computation of $f''(\xi)$ can be performed similarly to (A7). As is proved in Appendix B, $\Delta\delta$ can be written as $\Delta\delta = (\Delta\underline{w})^T \cdot \underline{a}$, where $\underline{a}$ is a vector depending on the weights before retraining and on the input vector $\underline{x}$. Consequently (A10) imposes two linear inequalities as far as the total weight perturbation $\Delta\underline{w}$ is concerned

$$-\phi \le (\Delta\underline{w})^T \cdot \underline{a} \le \phi. \tag{A11}$$

Satisfaction of (A10) indicates that the approximate network output, using the first-order Taylor series expansion, differs from the actual one at most by $\varepsilon_1|z_a|$, under the assumption that $\underline{u}_a = \underline{u}_b + \Delta\underline{u}$. However, taking into consideration the residuals R_i in (A4) the total error of the network output is greater by an amount depending on R_i. Let us denote by $\underline{r}$ a vector containing all residuals R_i of the first-layer neurons, i.e., $\underline{r} = [R_1 \ R_2 \cdots R_q]^T$. Then, using the mean value theorem, the error of the network output should be increased by the quantity $\mu_1(\underline{w}_a^1)^T \cdot \underline{r}$ where $\mu_1 = \max f'(\cdot)$. Since $\underline{w}_a^1 = \underline{w}_b^1 + \Delta\underline{w}$ and $\underline{r}$ is a very small quantity, the previous term can be considered to be approximately equal to $\mu_1(\underline{w}_b^1)^T \cdot \underline{r}$. Using the Cauchy–Schwartz formula and requiring that $\|\underline{r}\|_2 \le \varepsilon_2|z_a|/\mu_1\|\underline{w}_b^1\|_2$, the output error due to residuals R_i is smaller than $\varepsilon_2|z_a|$. Assuming for simplicity that all bounds of R_i in (A6) are equal to each other, i.e., $e_i = \varepsilon$ for all i, we can estimate the residual bounds as

$$\varepsilon \equiv e_i = \frac{\varepsilon_2|z_a|}{\mu_1 \left\|\underline{w}_b^1\right\|_2 \sqrt{q}} \tag{A12}$$

where q is the number of neurons in the first hidden layer. In this case all bounds ϑ_i of the inner product $\Delta\beta_i$ are the same, i.e.,

$$-\vartheta \le \left(\Delta\underline{w}_i^0\right)^T \cdot \underline{x} \le \vartheta, \qquad i = 1, 2, \cdots q. \tag{A13}$$

APPENDIX B

For a given input vector $\underline{x}_i$ in S_c and ignoring residual R in (A9), we can approximate the network output as follows:

$$z_a(\underline{x}_i) = z_b(\underline{x}_i) + f'(\delta_b(\underline{x}_i))\Delta\delta(\underline{x}_i) \tag{B1}$$

where we have introduced the dependence of the output on input vector $\underline{x}_i$. Using (9), (B1) is written as

$$c(\underline{x}_i) \equiv d_i - z_b(\underline{x}_i) = f'(\delta_b(\underline{x}_i))\Delta\delta(\underline{x}_i). \tag{B2}$$

Since $\Delta\delta(\underline{x}_i) = (\underline{w}_b^1)^T \cdot \Delta\underline{u}(\underline{x}_i) + \Delta\underline{w}^1 \cdot \underline{u}_b(\underline{x}_i)$ [see (A8)] indicating a linear relationship with the total weight perturbation $\Delta\underline{w}$, the term $f'(\delta_b(\underline{x}_i))\Delta\delta(\underline{x}_i)$ in (B2) can be expressed as

$$c(\underline{x}_i) = (\Delta\underline{w})^T \cdot \underline{a}(\underline{x}_i) \tag{B3}$$

where $\underline{a}(\underline{x}_i)$ is a vector provided by the solution of the following equation:

$$\begin{aligned} &(\Delta\underline{w})^T \cdot \underline{a}(\underline{x}_i) \\ &\quad = f'(\delta_b)\left[\left(\underline{w}_b^1\right)^T \cdot \Delta\underline{u}(\underline{x}_i) + \Delta\underline{w}^1 \cdot \underline{u}_b(\underline{x}_i)\right]. \end{aligned} \tag{B4}$$

Since $\Delta\underline{u}(\underline{x}_i)$ depends on the network weights, we have

$$\Delta\underline{u}(\underline{x}_i) = \nabla_{\underline{\beta}}\underline{f}\left(\underline{\beta}(\underline{x}_i)\right) \cdot \Delta\boldsymbol{W}^0 \cdot \underline{x}_i \tag{B5}$$

with

$$\underline{\beta}(\underline{x}_i) = [\beta_{1,b}(\underline{x}_i)\ \beta_{2,b}(\underline{x}_i) \cdots \beta_{q,b}(\underline{x}_i)]^T$$

and $\nabla\underline{f}(\cdot)$ the gradient matrix of the vector valued function $\underline{f}(\cdot)$. The $\underline{a}(\underline{x}_i)$ is thus calculated by

$$\begin{aligned} &(\Delta\underline{w})^T \cdot \underline{a}(\underline{x}_i) \\ &\quad = f'(\delta_b(\underline{x}_i))\left[\left(\underline{w}_b^1\right)^T \cdot \nabla_{\underline{\beta}}\underline{f}\left(\underline{\beta}(\underline{x}_i)\right) \cdot \Delta\boldsymbol{W}^0 \right. \\ &\qquad \left. \cdot\ \underline{x}_i + \Delta\underline{w}^1 \cdot \underline{u}_b(\underline{x}_i)\right]. \end{aligned} \tag{B6}$$

Taking into account all input vectors $\underline{x}_i$ in S_c, (B3) is expressed as

$$\underline{c} = \boldsymbol{A} \cdot \Delta\underline{w} \tag{B7}$$

where

$$\underline{c} = [c(\underline{x}_1)\ c(\underline{x}_2) \cdots c(\underline{x}_{m_c})]^T$$

and

$$\boldsymbol{A}^T = [\underline{a}(\underline{x}_1)\ \underline{a}(\underline{x}_2) \cdots \underline{a}(\underline{x}_{m_c})].$$

REFERENCES

[1] A. K. Aizawa and T. Huang, "Model-based image coding: Advanced video coding techniques for very low bitrate applications," *Proc. IEEE*, vol. 83, pp. 259–271, 1995.

[2] A. Alatan, L. Onural, M. Wollborn, R. Mech, E. Tuncel, and T. Sikora, "Image sequence analysis for emerging interactive multimedia services—The European cost 211 framework," *IEEE Trans. Circuits Systems Video Technol.*, vol. 8, no. 7, pp. 802–813, Nov. 1998.

[3] J. E. Besag, "On the statistical analysis of dirty pictures," *J. Roy. Statist. Soc.*, ser. B, vol. 48, pp. 259–302, 1986.

[4] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth, "Learnability and the Vapnik–Chervornenkis dimension," *J. Assoc. Computing Machinery*, vol. 36, pp. 929–965, 1989.

[5] C. Bouman and K. Sauer, "A generalized Gaussian image model for edge-preserving MAP estimation," *IEEE Trans. Image Processing*, vol. 2, pp. 296–310, 1993.
[6] C. Bouman and M. Shapiro, "A multiscale random field model for Bayesian image segmentation," *IEEE Trans. Image Processing*, vol. 3, no. 2, pp. 162–177, Mar. 1994.
[7] B. Cheng and D. M. Titterington, "Neural networks: A review from a statistical perspective," *Statist. Sci.*, vol. 9, pp. 2–54, 1994.
[8] L. Chiariglione, "MPEG and multimedia communications," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, pp. 5–18, 1997.
[9] A. D. Doulamis, Y. S. Avrithis, N. D. Doulamis, and S. Kollias, "Indexing and retrieval of the most characteristic frames/scenes in video databases," in *Proc. Wkshp. Image Anal. Multimedia Interactive Services (WIAMIS)*, Louvain-la Neuve, Belgium, June 1997, pp. 105–110.
[10] N. Doulamis, A. Doulamis, D. Kalogeras, and S. Kollias, "Low bit rate coding of image sequences using adaptive regions of interest," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 8, pp. 928–934, Dec. 1998.
[11] N. Doulamis, A. Doulamis, Y. Avrithis, and S. Kollias, "Video content representation using optimal extraction of frames and scenes," in *Proc. IEEE Int. Conf. Image Processing*, Chicago, IL, Oct. 1998.
[12] A. Eleftheriadis and A. Jacquin, "Automatic face location for model-assisted rate control in H.261 compatible coding of video," *Signal Processing Image Commun.*, vol. 7, pp. 435–455, 1995.
[13] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *IEEE Comput. Mag.*, pp. 23–32, Sept. 1995.
[14] J. Freer, B. Beggs, and H. Fernandez-Canqus, "Moving object surveillance and analysis of camera-based security systems," in *Proc. IEEE 29th Annu. Int. Carnghan Conf. Security Technol.*, U.K., 1995, pp. 67–71.
[15] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-6, pp. 721–741, 1984.
[16] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New York: Macmillan, 1994.
[17] "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbps,", ISO/CD 11172-2, Mar. 1991.
[18] "Generic coding of moving pictures and associated audio,", ISO/IEC 13 818-2, Recommendation H.262, Committee Draft, May 1994.
[19] "MPEG-7: Context and objectives (v.3),", ISO/IEC JTC1/SC29/WG11 N1678, Apr. 1997.
[20] R. Kinderman and J. L. Snell, *Markov Random Fields and their Applications*. Providence, RI: Amer. Math. Soc., 1980.
[21] S. Kollias, "An hierarchical multiresolution approach to invariant image recornition," *Neurocomputing*, vol. 12, pp. 35–57, 1996.
[22] S. Kung, "Neural networks for intelligent multimedia processing," *IEEE Signal Processing Mag.*, vol. 14, no. 4, July 1997.
[23] T.-Y. Kwok and D.-Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Trans. Neural Networks*, vol. 8, pp. 630–645, 1997.
[24] S. Lakshmanan and H. Derin, "Simultaneous parameter estimation and segmentation of Gibbs random fields using simulated annealing," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 799–813, 1989.
[25] D. J. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1984.
[26] R. Mech and M. Wollborn, "A noise robust method for 2D shape estimation of moving objects in video sequences considering a moving camera," in *Proc. Wkshp. Image Anal. Multimedia Interactive Services (WIAMIS)*, Louvain-la-Neuve, Belgium, 1997.
[27] F. Meyer and S. Beucher, "Morphological segmentation," *J. Visual Commun. Image Representation*, vol. 1, no. 1, pp. 21–46, Sept. 1990.
[28] MPEG Video Group, "MPEG-4 video verification model-version 2.1,", ISO/IEC/JTCI//SC29/WG11, May 1996.
[29] P. Mulroy, "Video content extraction: Review of current automatic algorithms," in *Proc. Wkshp. Image Anal. Multimedia Interactive Services (WIAMIS)*, Louvain-la Neuve, Belgium, June 1997, pp. 45–50.
[30] N. G. Panagiotidis, D. Kalogeras, S. D. Kollias, and A. Stafylopatis, "Neural network-assisted effective lossy compression of medical images," *Proc. IEEE*, vol. 84, pp. 1474–1487, 1996.
[31] D. C. Park, M. A. EL-Sharkawi, and R. J. Marks II, "An adaptively trained neural network," *IEEE Trans. Neural Networks*, vol. 2, pp. 334–345, 1991.
[32] R. Reed, "Pruning algorithms—A survey," *IEEE Trans. Neural Networks*, vol. 4, pp. 740–747, 1993.
[33] V. Ruiz de Angulo and C. Torras, "On-line learning with minimal degradation in feedforward networks," *IEEE Trans. Neural Networks*, vol. 6, pp. 657–668, 1995.
[34] P. Salembier and M. Pardas, "Hierarchical morphological segmentation for image sequence coding," *IEEE Trans. Image Processing*, vol. 3, no. 5, pp. 639–651, Sept. 1994.
[35] R. R. Shultz and R. L. Stevenson, "A Bayesian approach to image expansion for improved definition," *IEEE Trans. Image Processing*, vol. 3, pp. 233–242, 1994.
[36] T. Sikora, "The MPEG-4 video standard verification model," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 1, pp. 19–31, Feb. 1997.
[37] C. Stiller, "Object-based estimation of dense motion fields," *IEEE Trans. Image Processing*, vol. 6, no. 2, pp. 234–250, Feb. 1997.
[38] A. M. Tekalp, *Digital Video Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1995.
[39] N. Tsapatsoulis, N. Doulamis, A. Doulamis, and S. Kollias, "Face extraction from nonuniform background and recognition in compressed domain," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP)*, Seattle, WA, May 1998.
[40] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.
[41] G. G. Wilknson, "Open questions in neurocomputing for earth observation," in *Proc. 1st "COMPARES" Wkshp*, York, U.K., July 17–19, 1996.

Anastasios D. Doulamis (M'97) was born in Athens, Greece, in 1972. He received the Diploma degree in electrical and computer engineering from the National Technical University of Athens (NTUA) in 1995 with the highest distinction. He joined the Image, Video, and Multimedia Lab of NTUA in 1996 and is working toward the Ph.D. degree.

His research interests include neural networks to image/signal processing, multimedia systems, and video coding based on neural-network systems.

Mr. Doulamis was awarded the Best New Engineer at national level and he received the Best Diploma Thesis Award by the Technical Chamber of Greece. He was also given the NTUA Medal, while he has received several awards and prizes by the National Scholarship Foundation.

Nikolaos D. Doulamis (M'97) was born in Athens, Greece, in 1972. He received the Diploma degree in electrical and computer engineering from the National Technical University of Athens (NTUA) in 1995 with the highest honor. He is currently a Ph.D. candidate in the Electrical and Computer Engineering Department of the NTUA and is supported by the Bodosakis Foundation Scholarship.

His research interests include image/video analysis and processing and multimedia systems.

Mr. Doulamis has received several awards and prizes by the Technical Chamber of Greece, the NTUA and the National Scholarship Foundation, including the Best New Engineer Award at national level, the Best Diploma Thesis Award and the NTUA Medal.

Stefanos D. Kollias (S'81–M'84) was born in Athens, Greece, in 1956. He received the Diploma degree in electrical engineering from the National Technical University of Athens (NTUA) in 1979, the M.Sc. degree in communication engineering from the University of Manchester (UMIST), U.K., in 1980 and the Ph.D. degree in signal processing from the Computer Science Division of NTUA in 1984.

In 1982, he received a ComSoc Scholarship from the IEEE Communications Society. Since 1986, he has been with the NTUA where he is currently a Professor. From 1987 to 1988, he was a visiting Research Scientist in the Department of Electrical Engineering and the Center for Telecommunications Research, Columbia University, NY. His current research interests include image processing and analysis, neural networks, image and video coding, multimedia systems. He is the author of more than 140 articles.