An Application of Evolving Fuzzy Neural Network for Compressed Video Parsing

Irena Koprinska and Nikola Kasabov

Department of Information Science, University of Otago, Dunedin, New Zealand
e-mail: {ikoprinska, nkasabov}@infoscience.otago.ac.nz

Abstract

This paper reports an application of evolving fuzzy neural network (EFuNN) as a module in a system for MPEG compressed video parsing. EFuNN learns from pre-classified examples in the form of motion vector patterns in order to distinguish between six classes: static, panning, zooming, object motion, tracking and dissolve. The performance of EFuNN is compared with LVQ and the results are discussed. In addition, the impact of the number of membership functions and the contribution of the rule node aggregation are analyzed.

Key words: video parsing, MPEG, neural networks, evolving fuzzy neural networks, one pass training

1. Introduction

Video parsing is the first step towards automatic annotation of digital video sequences. Its goal is to divide the video stream into a set of meaningful and manageable segments that are used as basic elements for indexing. Typically video is segmented into shots, shot transitions and camera operations within shots. Each shot is then represented by key frames and indexed by extracting spatial and temporal features.

Since video is increasingly stored in compressed format (e.g. MPEG), several parsing approaches that operate directly on the compressed video were proposed [7,8,9,10,12]. Their main disadvantages are summarized in [5,6] and a hybrid rule-based/neural system was proposed as an alternative. It is based only on the information available in the MPEG-2 stream, namely the motion vectors (MV) and macroblock (MB) coding mode in B and P frames. The system follows a two-pass scheme. A rough scan over the P frames locates the potential shot boundaries and the solution is then refined by a precise scan over the B frames of the respective neighborhoods. The "simpler" boundaries are recognized by the rule-based module, while the decisions for the "complex" ones are refined by the neural part. To do this, the neural module learns from pre-classified examples in the form of MV patterns and is trained to distinguish between six classes: static, panning, zooming, object motion, tracking and dissolve.

The goal of this paper is to study the potential of *evolving fuzzy neural networks (EFuNNs)* as a neural module in a system for video parsing. EfuNNs are novel and promising neural model that offers one-pass training and good generalization.

2. Evolving Fuzzy Neural Networks

EFuNNs are fuzzy neural networks for prediction and classification introduced by Kasabov [2,3]. They have a five-layer structure, see Figure 1.

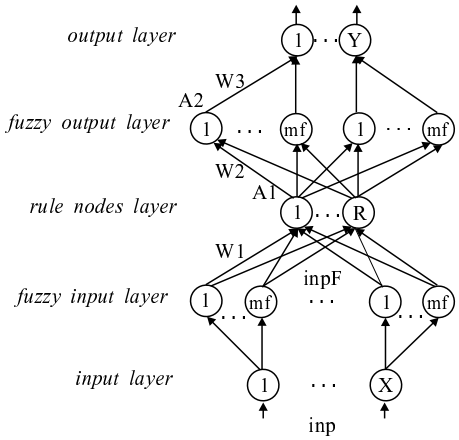

Figure 1. EFuNN's architecture

The *input* layer represents input variables. The second layer of nodes (*fuzzy input* neurons) represent fuzzy quantification of each input variable space using membership functions. The third layer contains *rule nodes* that evolve through learning according to the ECOS principles [2]. The rule nodes represents prototypes of data mapping between the fuzzy input and fuzzy output spaces. Each rule node is defined by two vectors of connection weights: W1 and W2. The former is adjusted via unsupervised learning based on the similarity between the fuzzy input vector and the prototypes already stored. W2 is updated applying LMS algorithm to minimize the output fuzzy error. Neurons in the fourth layer are called *fuzzy output* neurons and represent the fuzzy quantization for the output variables. Finally, the fifth layer contains *output* nodes that represent the real values for the output variables.

There are several options for growing of the EFuNN architecture [4]. We used the 1-of-n EFuNN algorithm that is summarized below.

2.1 Overview of the EFuNN algorithm

Data preprocessing:

1. Normalize input vectors (training examples) inp_i in [0,1], where *i=1..N* and *N* is the number of training examples.

2. Fuzzify training examples using *mf* triangular membership functions:

$inpF_i = fuzzify(inp_i)$, *i=1..N*

Hence, if *X* denotes the number of input vector features (i.e., the dimension of inp_i), the fuzzy input vector $inpF_i$ will contain $X \cdot mf$ elements.

3. Set the defuzzifying weights W3 between the fuzzy outputs and real outputs as follows:

$$W3_y = \frac{y}{mf - 1},$$

where $W3_y$ is the *W3* vector for the class *y*, *y=1..Y*, *Y* is the number of classes.

Hence, the dimension of the fuzzy output vector will be $Y \cdot mf$.

Training of the network:

1. Create the first rule node r_1 to represent the first example*:*

$W1_1 = InpF_1,\ W2_1 = target_1$

2. While *(i<N)* (i.e. there are training examples):

i=i+1;

For the i^{th} fuzzy training example $(InpF_i, target_i)$:

a) calculate the *normalized fuzzy local distance D* between the fuzzy input vector $InpF_i$ and the already stored prototypes in the rule nodes r_j, *j=1..R,* where *R* is the current number of rule nodes:

$$D(InpF_i, r_j) = \frac{\sum_{j=1}^{R} \left| InpF_i - W1_{r_j} \right|}{\sum_{j=1}^{R} W1_{r_j}}$$

b) calculate the activation $A1_{r_j}$ of the rule nodes r_j , *j=1..R:*

$$A1_{r_j} = 1 - \frac{D(InpF_i, r_j)}{2}$$

c) find the rule node r_{j*} with highest activation $A1$

d) <u>if</u> $A1_{r_{j*}} < sThr$ <u>then</u>

create a new rule node:

$W1_i = InpF_i,\ W2_1 = target_1$;

j=j+1;

<u>else</u>

- propagate the activation of r_{j*} to the action neurons: $A2 = A1_{r_{j*}} \cdot W2_{r_{j*}}$
- calculate the fuzzy output error:

  $Err = \left| A2 - target_i \right|$
- find the action node *k** with highest activation *A*2
- <u>if</u> *((k* ≠ t) or (Err(k*) > errThr))* <u>then</u>

  create a new rule node:

  $W1_i = InpF_i,\ W2_1 = target_1$;

  j=j+1;

  <u>else</u> update the input and output connections of rule node *k*:*

  $W1_{k*}^{t+1} = W1_{k*}^{t} + lr1 \cdot D(inpF_i, r_j)$

  $W2_{k*}^{t+1} = W2_{k*}^{t} + lr2 \cdot Err_{k*} \cdot A1_{j*}$

e) <u>if</u> *(i = iAg)* <u>then</u> aggregate:

- for each rule node r_j , *j=1..R* find the subset of rule nodes r_a, *a=1..A, A<R* for which the *normalized Eucledian distances* $D_{euc}(W1_{r_j}, W1_{r_a})$ and $D_{euc}(W2_{r_j}, W2_{r_a})$ are below the thresholds *w1Thr* and *w2Thr*, respectively:

$$D_{euc}(W1_{r_j}, W1_{r_a}) = \frac{\sqrt{\sum_{j=1}^{m} (W1_{r_j}^{m} - W1_{r_a}^{m})^2}}{\sqrt{m}} < w1Thr$$

$$D_{euc}(W2_{r_j}, W2_{r_a}) = \frac{\sqrt{\sum_{j=1}^{l} (W2_{r_j}^{l} - W2_{r_a}^{l})^2}}{\sqrt{l}} < w2Thr$$

where *m* and *l* are the numbers of conditional nodes and action nodes, respectively.

- merge the nodes r_a, *a=1..A* and update $W1_{r_j}$ and $W2_{r_j}$ as follows:

$$W1_{r_j} = \frac{\sum_{a=1}^{A} (W1_{r_a})}{A}, W2_{r_j} = \frac{\sum_{a=1}^{A} (W2_{r_a})}{A}$$

- delete r_a , *a=1..A*

Classification:

To classify an example that has not been seen during the learning phase, it is first normalized and then propagated through EFuNN. The propagation from

rule nodes to output layer is restricted only for the winning rule node. The example is classified as an instance of the class *y*, where *y* is the index of the output neuron with the highest value.

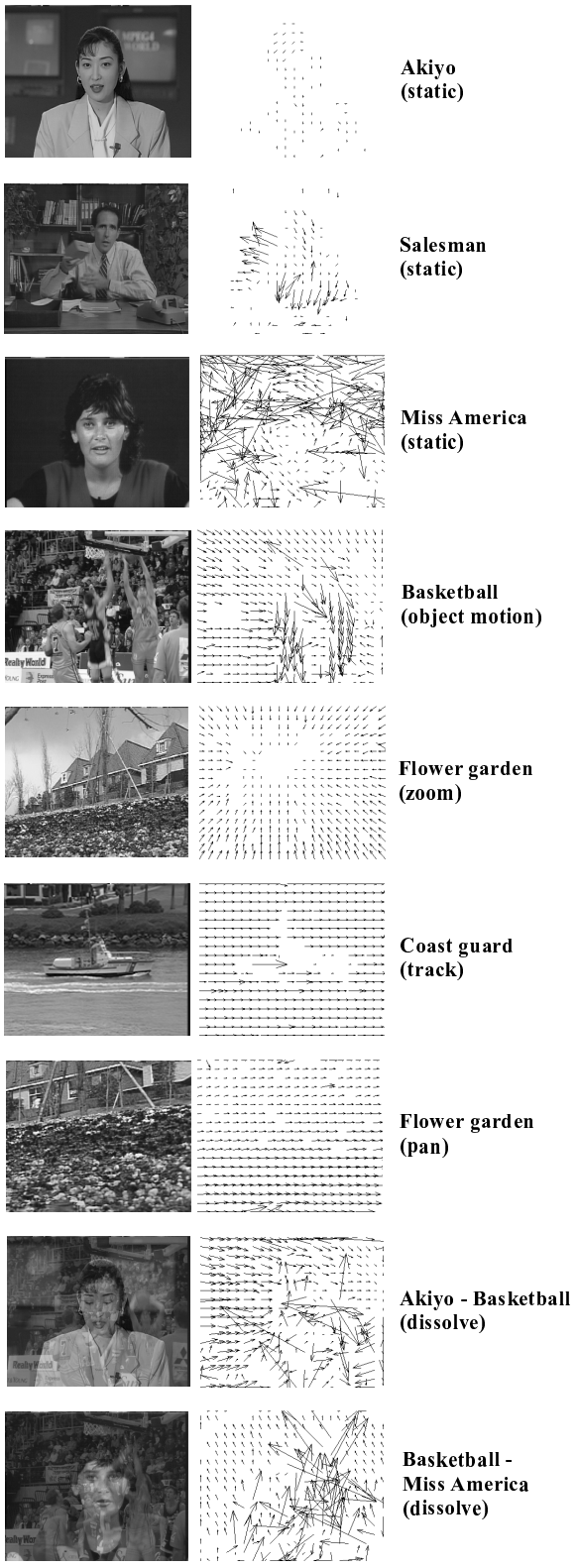

Figure 2. Motion vector patterns corresponding to the different classes

3 Data Description

3.1 Video frame classification

Following [5], six classes are defined:

- *static* - stationary camera and little scene motion*;*
- *panning* - camera rotation around its horizontal axis;
- *zooming* - focal length change of a stationary camera;
- *object motion* - stationary camera and large scene motion;
- *tracking* - moving object being tracked by a camera
- *dissolve* - gradual transition between two sequences showing one image superimposed on the other as the frames of the first shot get dimmer and these of the second one get brighter.

Each of these six classes is characterized by a specific pattern in the field of MVs of P and B frames in a MPEG encoded sequence, as shown in Figure 2. Hence, the goal is to build an EFuNN classifier for successful recognition of the samples.

The well-known benchmark sequences *Akiyo*, *Salesman*, *Miss America*, *Basketball*, *Football*, *Tennis*, *Flower Garden*, and *Coastguard* have been used in our experiments. As the image format is CIF (288x352 pixels), there are 396 (16x16 pixels) MBs in each frame. While the first three video sequences are static, the next three involve a lot of object motion, and the last two are mainly examples for panning and tracking, respectively. Since there were only few frames with zoom in the sequences above, additional zooming was generated artificially for each of them.

3.2 Data pre-processing

Pre-processing of the data and feature extraction were done as in [6]. After their compression by the *MPEG Software Simulation Group Encoder*[1], the MVs associated with each MB in P and B frames have been extracted and smoothed by the application of 3x3 cross shaped vector median filter. Based on them, a 22-dimentional feature vector has been generated for each frame. The first component (V_1) is the fraction of MBs with no motion, calculated as: $(Z_f+Z_b+Z_{df}+Z_{db})/(f+b+2d)$, where Z_f, Z_b, Z_{df} and Z_{db} are the number of zero MVs in forward and backward predicted MBs, forward and backward MV components of bi-directional MBs of the current frame, respectively. Then, the forward MV area is sub-divided in 7 vertical strips, for which 3 parameters are computed: the average of the MV direction ($V_2 \div V_8$), the standard deviation of the MV

[1] http://www.mpeg.org/MPEG/MSSG/VMPEG

direction ($V_9 \div V_{15}$) and the average of MV modulus ($V_{16} \div V_{22}$).

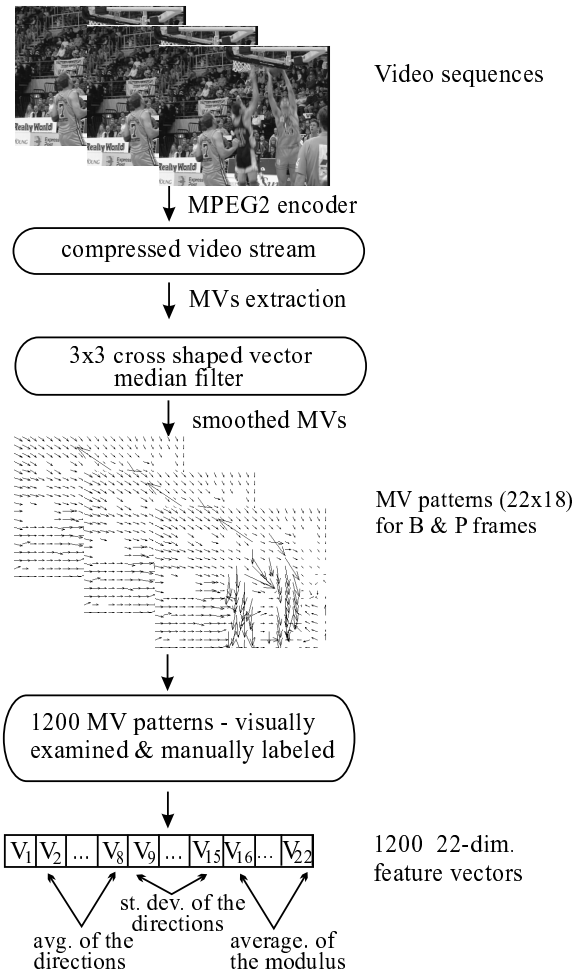

Figure 3. Video data pre-processing

In order to build the EFuNN classifier, the MV patterns of 1200 P and B frames, have been visually examined and manually labeled. The frames were selected so that the number of examples for each type is equal (i.e. 200).

Several aspects of data complicate learning. First, as it could be seen from Figure 2, the three static examples have rather different MV fields. While the frames of *Akiyo* can be viewed as examples of ideal static images, there are occasionally sharp movements in the second video sequence *(Salesman)*. The MV field of the last static sequence *(Miss America)* is completely different because the homogeneous black background results in random orientation of the respective MVs.

4. Experiments

The goal of the experiments was fourfold: 1) to test the overall classification performance of EFuNN for video frames classification; 2) to analyze the recognition of the individual classes; 3) to find how the different number of fuzzy membership functions influence the EFuNN performance; 4) to assess the contribution of the hidden nodes aggregation.

4.1 Evaluation methodology

For the evaluation of the results of the EFuNN classification we used *10-fold cross validation* [11]. The original data file was randomly split into 10 non-overlapping subsets of equal size (120 examples). Each time the examples of 9 of the subsets were used for training and the resulting classifier was tested on the remaining partition. The experiments were repeated 10 times and the results were averaged. This methodology is known to provide a very good error estimate.

4.2 EFuNN Parameters

Apart from the various values for *mf* and *w1Thr/w2Thr* as discussed below, we use the following EFuNN parameters: *sThr*=0.92, *errThr*=0.08, *lr1*=0.05, *lr2*=0.01, *nAgg*=60. No pruning was applied.

4.3 Results and discussion

Table 1 shows the classification accuracy of EFuNN with different number of membership functions when applied for video frames classification. The respective number of nodes are presented in Table 2. For the needs of comparison, Table 3 summarizes the results achieved by *learning vector quantization (LVQ)* [1] using the public domain package LVQPack[2].

mf	accuracy [%] on the training set	accuracy [%] on the testing set
2	85.84 ± 1.47	84.50 ± 2.39
3	91.43 ± 1.09	86.75 ± 4.5
4	95.26 ± 0.62	91.58 ± 2.9
5	95.50 ± 0.38	89.25 ± 4.3
6	95.23 ± 0.87	88.58 ± 4.45

Table 1. EFuNN classification accuracy [%] on the training and testing set (*w1Thr*=*w2Thr*=0.2)

	mf				
nodes	2	3	4	5	6
input	22	22	22	22	22
fuzzy input	44	66	88	110	132
rule (hidden)	30 ± 2.0	101.3 ± 5.5	183.1 ± 5.5	204.9 ± 9.6	229.5 ± 7.9
fuzzy output	12	18	24	30	36
output	6	6	6	6	6
total	114	213	323	362	425

Table 2. Number of nodes for the various EFuNN architectures (*mf*=2÷6)

[2] http://nucleus.hut.fi/nnrc/

accuracy [%] on the training set	accuracy [%] on the testing set	nodes (input & codebook)	training epochs
85.42 ± 2.5	85.83 ± 2.2	60 (22 inp. & 38 cod.)	1520

Table 3. LVQ performance on video frame classification

As it can be seen from Table 1, EFuNN achieves best classification accuracy when 4 membership functions are used. Further increase in the number of membership functions almost does not affect the accuracy on the training set but results in worse accuracy on unseen examples due to overtraining.

On the other hand, Table 2 indicates that increasing the number of the membership functions implies considerable growth in the number of rule nodes and, hence, the computational complexity of the EFuNN's training algorithm. As a result, learning speed slows down significantly. However, depending on the specific application a suitable trade-off between the learning time and the classification accuracy can be found.

The performance of EFuNN compares favourably with LVQ in terms of classification accuracy (see Table 1,2 and Table 3). Another advantage of EFuNN is that it requires only 1 epoch for training in contrast to LVQ's multi pass learning algorithms that needs 1520 epochs in our case study. It should be noted, however, that the LVQ network is much smaller than the EFuNN architectures.

mf	zoom	pan	object mov.
2	100 ± 0.0	97.19 ± 5.42	74.58 ± 12.01
3	100 ± 0.0	92.75 ± 14.11	78.05 ± 12.69
4	100 ± 0.0	97.36 ± 2.80	88.08 ± 10.70
5	100 ± 0.0	99.01 ± 2.11	84.96 ± 9.20
6	100 ± 0.0	94.28 ± 5.22	88.38 ± 9.66

mf	static	tracking	dissolve
2	95.00 ± 6.59	75.85 ± 15.58	62.08 ± 14.79
3	97.75 ± 4.41	72.89 ± 20.55	77.29 ± 14.65
4	98.11 ± 2.45	83.02 ± 11.46	83.95 ± 10.83
5	97.73 ± 3.11	69.70 ± 21.51	85.15 ± 9.54
6	97.68 ± 3.26	63.54 ± 18.84	87.57 ± 8.03

Table 4. Classification accuracy [%] of the individual video classes using EfuNN with different number of fuzzy membership functions

Table 4 summarizes the EFuNN classification of the individual classes. It was found that while some classes are easily identified (e.g. zoom, pan), the recognition of object movement, tracking and dissolve is more difficult. A more detailed analysis indicates that the algorithm actually has difficulties to discriminate well between these three classes which explains also the large standard deviations. Despite the fact that the MV fields of *Miss America* were not typical for static sequences and complicated learning, they are correctly classified by the EFuNN system.

Figure 4 and Figure 5 show the impact of the aggregation on the classification accuracy and the number of rule nodes, respectively. Again, 10-fold cross validation was applied and each number represents the mean value for the ten runs[3].

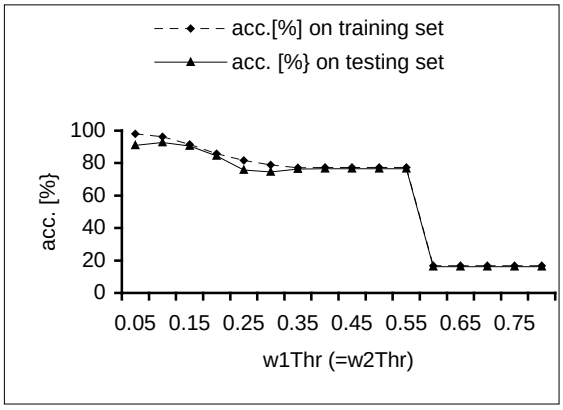

Figure 4. Impact of the aggregation parameters *w1Thr & w2Thr* on the classification accuracy (*mf*=2)

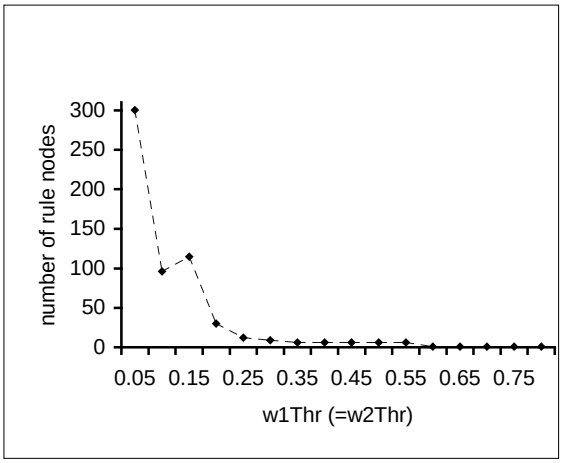

Figure 5. Impact of the aggregation parameters *w1Thr* and *w2Thr* on the number of rule nodes (*mf*=2)

As expected, the results demonstrate that the aggregation is an important factor for good generalization and keeping the network's architecture at reasonable size. The best performance in terms of a good trade-off between the recognition accuracy and the network size was achieved for *w1Thr*=*w2Thr*=0.15, 0.2. When the aggregation coefficients are between 0.25 and 0.55, the classification accuracy on the testing set drops with about 10% as the number of rule nodes becomes insufficient. Further increase in the values of the

[3] For the sake of visualization the standard deviations are omitted.

aggregation coefficients results in network with one rule node that obviously can not be expected to generalize well.

5. Conclusions

An application of EFuNN for compressed video parsing was presented. EFuNN learns from examples in the form of motion vector patterns, extracted from the MPEG-2 stream. The success of the neural system can be summarized as high classification accuracy and fast training.

Future work will focus on the possibility to further improve the performance combining image and motion information with audio features and text captions. The approach will be also extended to incrementally accommodate new classes, e.g. other common types of gradual transitions and camera operations.

Acknowledgements

The work was supported by the *New Zealand Foundation for Research, Science and Technology*, contract UOO-808. The implementation of the EFuNN algorithm is a part of the *New Zealand Repository for Intelligent Information Systems (RICBIS)* and is available from the following site: http://divcom.otago.ac.nz/infosci/kel/CBIIS/CBIIS.html, link Software/EfuNNs/.

References

[1] T. Kohonen (1990). The Self-Organizing Map, *in Proc. of the IEEE*, v.78(9), pp. 1464-1480.

[2] N. Kasabov (1998). ECOS: A Framework for Evolving Connectionist Systems and the ECO Learning Paradigm, *in Proc. of Int. Conference of Neural Information Processing (ICONIP'98)*, pp.1222-1235.

[3] N. Kasabov (1999). Evolving connectionist and fuzzy connectionist systems – theory and applications for adaptive, on-line intelligent systems, *In: Neuro-Fuzzy Techniques for Intelligent Information Processing*, N. Kasabov and R. Kozma (eds.), Heidelberg Physica Verlag.

[4] N. Kasabov and Q. Song (1999). Dynamic Evolving Fuzzy Neural Networks with 'm-out-of-n' Activation Nodes for On-line Adaptive Systems, *IEEE Transactions on Fuzzy Systems*, submitted.

[5] I. Koprinska and S. Carrato (1998). Detecting and Classifying Video Shot Boundaries in MPEG Compressed Sequences, *in Proc. of the European Conference on Signal Processing (EUSIPCO'98)*, pp. 1729-1732.

[6] I. Koprinska and S. Carrato (1998). Segmentation of Compressed Video by Learning Vector Quantizer, *in Proc. of the Int. Conference on Engineering Applications of Neural Networks (EANN'98)*, Gibraltar, pp. 9-16.

[7] J. Meng, Y. Juan and S. Chang (1995). Scene Change Detection in a MPEG Compressed Video Sequence, *in Proc. IS&T/SPIE Intern. Symposium Electronic Imaging*, vol. 2417, pp. 14-25.

[8] N.V. Patel and I.K. Sethi (1997). Video Shot Detection and Characterization for Video Databases, *Pattern Recogition*, v. 30, pp. 583-592.

[9] K. Shen and E.J. Delp (1996). A Fast Algorithm for Video Parsing Using MPEG Compressed Sequences, *in Proc. of the Intern. Conference on Image Processing (ICIP'96)*.

[10] B. Yeo and B. Liu (1995). Rapid Scene Analysis on Compressed Video, *IEEE Transactions on Circuits & Systems for Video Technology*, vol. 5(6), pp. 533-544.

[11] S.W. Weiss and C.A. Kulikowski (1991). *Computer Systems That Learn*, Morgan Kaufmann, 1991.

[12] H. Zhang, C.Y. Low and S.W. Smoliar (1995). Video Parsing and Browsing Using Compressed Data, *Multimedia Tools & Applications*, vol.1, pp. 89-111.