11

Engine Knock Detection Based on Computational Intelligence Methods

Adriana Florescu[1], Claudiu Oros[1] and Anamaria Radoi[2]
[1]*University Politehnica of Bucharest,*
[2]*Ecole Politechnique Federale de Laussane,*
[1]*Romania*
[2]*Switzerland*

1. Introduction

Artificial intelligence emerged from human thinking that has both logical and intuitive or subjective sides. The logical side has been developed and utilized, resulting advanced von Neumann type computers and expert systems, both constituting the *hard computing* domain. However, it is found that hard computing can't give the solution of very complicated problems by itself. In order to cope with this difficulty, the intuitive and subjective thinking of human mind was explored, resulting the *soft computing* domain (also called *computational intelligence*). It includes *neural networks*, *fuzzy logic* and *probabilistic reasoning*, the last gathering *evolutionary computation* (including *genetic algorithms* with related efforts in *genetic programming* and *classifier systems*, *evolution strategies* and *evolutionary programming*), *immune networks*, *chaos computing* and parts of *learning theory*. In different kind of applications, all pure artificial intelligence methods mentioned above proved to be rather complementary than competitive, so that combined methods appeared in order to gather the advantages and to cope with the disadvantages of each pure method. The scope of this chapter is to study and finaly compare some representative classes of pure and combined computational intelligence methods applied in engine knock detection.

The internal-combustion engine is one of the most used vehicle power generators in the world today. When looking at the characteristics of a vehicle - and therefore the ones of the engine that drives it - , some of the most important are the emissions, fuel economy and efficiency. All three of these variables are affected by a phenomenon that occurs in the engine called knock**.** *Engine knock* (also known as *knocking*, *self-combustion*, *detonation*, *spark knock* or *pinging*) in spark-ignition internal combustion engines occurs when combustion of the mixture of fuel and air in the cylinder starts off correctly because of the ignition by the spark plug, but one or more pockets of the mixture explode outside the normal combustion front. The importance of knock detection comes from the effects it generates; these can range from increased fuel consumption and pollution, the decrease of engine power and up to partial or complete destruction of the cylinders, pistons, rods, bearings and many other damages around the engine bay.

Internal combustion engines present an optimum working cycle that is right on the edge of self-combustion or knock. If engine knock occurs and is detected in a cycle then the ignition timing (spark angle) needs to be modified so that the next cycle does not suffer from the same phenomenon. This is why the detection needs to be done in under a cycle (Bourbai, 2000; Li&Karim, 2004; Hamilton&Cowart, 2008; Erjavec, 2009).

Engine knock can be detected using a series of devices placed in and around the engine bay like: pressure sensors mounted inside each cylinder, devices that measure the ionization current in the spark plug or accelerometers mounted on the engine to measure vibrations etc. The best and most accurate information on knock is given by the *pressure sensors* but the easiest and less expensive way to detect it is by using *vibration sensors* mounted on the engine (Erjavec, 2009; Gupta, 2006; Bosch, 2004; Thomas et al., 1997; Ettefag, 2008, Fleming, 2001). The knock detection methods used so far for extracting information from the engine sensors include *time*, *frequency (spectrum)* or a diversity *time-frequency analysis (Wavelet)* based solutions (Adeli&Karim, 2005; Park&Jang, 2004; Radoi et al., 2009; Midori et al., 1999; Lazarescu et al., 2004; Jonathan et al., 2006). The restriction of average detection rates and the complexity of information needed for the Wavelet analysis support further developments and hybridization with mixed techniques that proved useful in other fields of application than the one explored in this chapter: *wavelet-fuzzy* (Borg et al., 2005), *wavelet-neural* (Zhang&Benveniste, 1992; Billings&Wei, 2005; Wu&Liu, 2009; Banakar&Azeem, 2008) and *wavelet-neuro-fuzzy* (Ylmaz&Oysal, 2010).

Among the pure computational intelligence methods described in (Wang&Liu, 2006; Prokhorov, 2008; Mitchell, 2010;Wehenkel, 1997), different types of neural network applications have been employed with better detection rates than the previous non-neural methods but no clear comparative analysis results have been presented so far for engine knock detection. The methods taken into account and finally compared in this chapter start with the *Fuzzy Kwan-Cai Neural Network* (Kwan&Cai, 1994) - for the application of which other neuro-fuzzy or fuzzy logic models were studied (Zhang&Liu, 2006; Ibrahim,2004; Liu&Li, 2004; Hui, 2011; Chen, 2005) -, expand to the *Kohonen Self-Organizing Map (SOM)* (Kohonen, 2000, 2002; Hsu, 2006; Lopez-Rubio, 2010) and end with *Bayes Classifier* (Larose, 2006) to which results of this chapter conforming with other work (Auld et al., 2007) published so far have proved needing hybridization.

Work started using two sizes of training and testing sample groups, both belonging to the Bosch Group database in order to see how data size can affect the results. The applications were built to handle both pressure and vibration samples in order to see which of them can supply the most valuable information. In addition, due to the lack of chapters available on this subject, through the analysis of the results, we can get a better impression of the nature of these types of signals, the coherence of samples and evolution of detection rates with every new sample added. Also, to complete the analysis, a comparison of the responses from pressure and vibration families of samples is made for the three methods.

2. Mathematical background of used computational intelligence methods

2.1 Fuzzy Kwan-Cai neural network

The Fuzzy Kwan-Cai neural network shown in Fig.1 has four layers, each of them being a fuzzy block represented by a different type of fuzzy neurons with their own specific purpose and functions (Kwan&Cai, 1994).

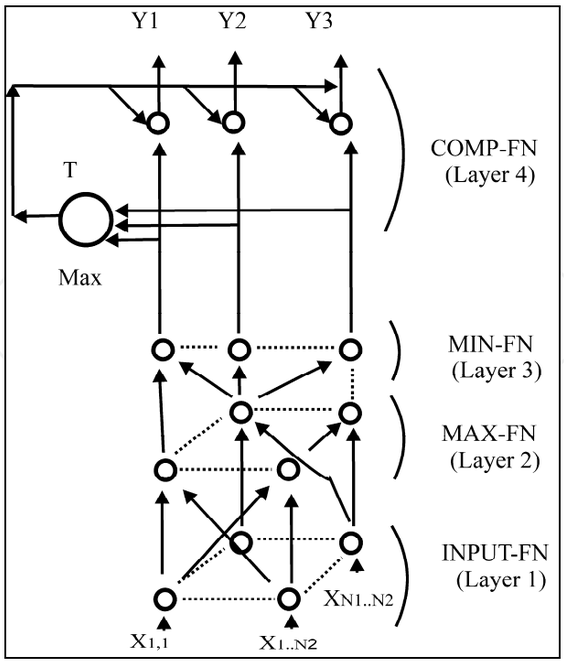


Fig. 1. The Fuzzy Kwan-Cai Neural Network structure

The first layer represents the input and is built with fuzzy input neurons, each one selecting a characteristic of the original sample vector. In the case of a two dimensional sample containing N1xN2 vector elements we will have a first layer that has N1xN2 neurons. For the neuron on the (i, j) position the equations are:

$$s_{ij}^{[1]} = z_{ij}^{[1]} = x_{ij} \,, \tag{1}$$

$$y_{ij}^{[1]} = s_{ij}^{[1]} \Big/ P_{v\max} \,, \tag{2}$$

for i=1, 2, …, N1; j=1, 2, …, N2, where $s_{ij}^{[1]}$ represents the state of the neuron on the (i, j) position for the first layer, $z_{ij}^{[1]}$ is it's input value, x_{ij} is the value of the element (i, j) in the input sample pattern, ($x_{ij} \geq 0$), $y_{ij}^{[1]}$ is its output value and $P_{v\max}$ is the maximum value of all the input elements. The notation will be kept for neurons belonging to all the following layers.

The second layer is built of N1xN2 neurons and its purpose is to perform the fuzzification of the input patterns by means of the weight function $w(m,n)$ - also called the fuzzification function - , defined as:

$$w(m,n) = e^{-\beta^2 (m^2+n^2)} \,, \tag{3}$$

where parameters m=-(N1-1), …, +(N1-1), n=-(N2-1), …, +(N2-1) and β determines how much of the sample vector each fuzzy neuron sees. Each neuron from the second layer has M outputs, one for each neuron in the third layer. The output for the second layer neuron on position (p, q) is:

$$y_{pqm}^{[2]} = q_{pqm}^{[2]}, \tag{4}$$

for p=1, …, N1; q=1, …, N2; m=1,…,M, where $y_{pqm}^{[2]}$ is the m^{th} output of the second layer neuron on position (p,q) to the m^{th} third level neuron. The output function q_{pqm} is determined within the training algorithm. For a more simplified approach, we can choose isosceles triangles with the base α and the height 1, mathematically defined as:

$$y_{pqm}^{[2]} = q_{pqm}\left(s_{pq}^{[2]}\right) = \begin{cases} 1 - \dfrac{2\left|s_{pq}^{[2]} - \theta_{pqm}\right| \le \dfrac{q}{2}}{\alpha}, \; for \; \left|s_{pq}^{[2]} - \theta_{pqm}\right| \le \dfrac{\alpha}{2}, \\ 0, \; other \end{cases} \tag{5}$$

where $\alpha \ge 0$, p=1, …, N1; q=1, …, N2; m=1, …, M. Parameter θ_{pqm} is the center of the isosceles triangle base. By means of the training algorithm, p, q and m values corresponding to α and θ_{pqm} are determined.

The third layer is made-up of M neurons each of them representing a learned pattern and so the value for M can only be determined at the end of the learning process. It can be seen as a fuzzy deduction (inference) layer. The output for the third layer neuron is:

$$y_m^{[3]} = s_m^{[3]} = \min_{p=1...N1}(\min_{q=1...N2}(y_{pqm}^{[2]})), \tag{6}$$

for m=1,…, M.

The fourth and final layer is the network's output layer and is made up of competitive neurons one for each pattern that is learned; it is the defuzzification layer. If an input pattern is more similar to the mth pattern that was learned, then the output of the mth comparative neuron will be attributed value 1 and the others value 0:

$$y_m^{[4]} = s_m^{[4]} = z_m^{[4]}, \tag{7}$$

$$y_{pqm}^{[4]} = q[s_m^{[4]} - T] = \begin{cases} 0, \; if \; s_m^{[4]} < T \\ 1, \; if \; s_m^{[4]} = T \end{cases}, \tag{8}$$

$$T = \max_{m=1...M}(\max_{j=1...N2}(y_m^{[3]})), \tag{9}$$

for m=1,…, M, where T is defined as the activation threshold for all the neurons in the forth layer.

The flowchart in Fig.2 summarizes the procedure of adapting and implementing the Fuzzy Kwan-Cai algorithm to the application proposed in the chapter. The differences from the standard theoretical algorithm are that the sample databases are first imported and validated for integrity and then separated into pressure and vibration, respectively training and testing classes. The standard classification steps follow and the algorithm ends with the calculation of the detection rate.

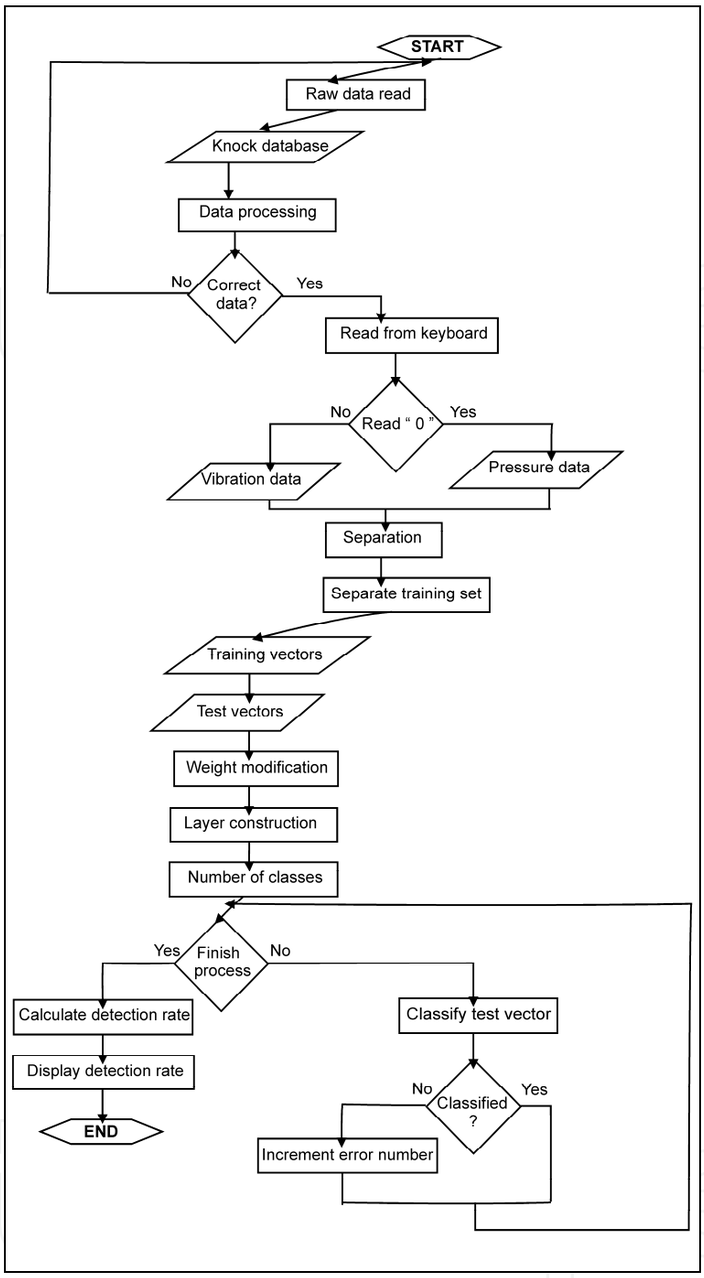


Fig. 2. Flowchart for implemented Kwan-Cai algorithm

2.2 Kohonen Self-Organizing Map (SOM)

The Kohonen Self-Organizing Map (SOM) with the structure presented in Fig.3 is a neural network characterized by the fact that neighboring neurons (cells) communicate among themselves by mutual-lateral interactions transforming into detectors of specific classes when given input patterns. The learning can be unsupervised or supervised (Kohonen, 2000, 2002; Hsu, 2006; Lopez-Rubio, 2010) In this chapter the supervised learning algorithm was used.

The network transforms similarities among vectors into neural vicinities (the similar input patterns will be found as neighbors).

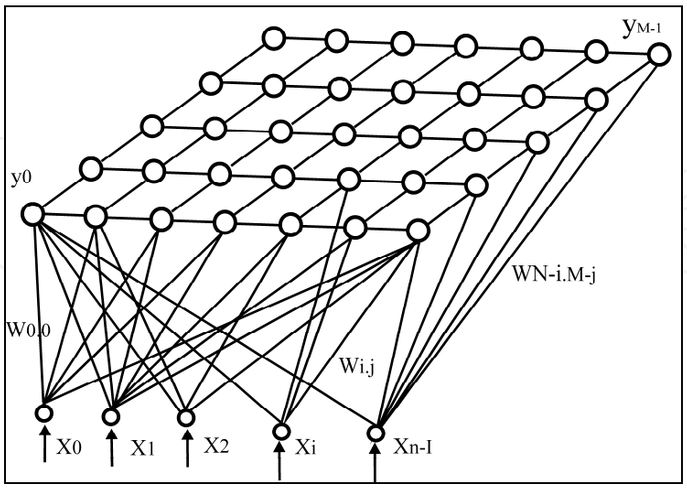


Fig. 3. The SOM neural network

From a structural point of view, the Kohonen neural network is composed of two layers out of which the first one is an input layer made of transparent neurons with no processing functions. Its purpose is to receive the input pattern and send it to the second layer. This first layer has the same size as the input pattern.

The second layer contains M output neurons, a number equal or higher than the number of classes desired in order to classify the entry patterns. They can be arranged planar, linear, circular, as a torus or sphere, the training and performances being dependent on the network shape. The planar network can also be rectangular or hexagonal depending on the placement of neurons.

An input vector $X_p \in R^n$ is applied in parallel to all the neurons of the network, each of them being characterized by a weight vector:

$$W_{j=}(w_{0j}, w_{1j}, ..., w_{n-11j})^T \in R^n, \tag{10}$$

for j=0, 1, …, M-1.

In order to choose the winning neuron j* with its associated weight vector W_j* for an input pattern we must calculate the Gaussian distance d_j between that pattern and each of the neuron's weight vectors. The winner will be chosen by the lowest distance d_j^* of all:

$$d_j = \left\| X_p - W_j \right\|, \tag{11}$$

$$d_j^* = \min\{d_j\}, \tag{12}$$

for j=0, 1, …, M-1.

After the winner determination process has finished the weights refining one is started and this must not have an effect on all the neurons but only in a certain vicinity V_j^* around the winner j*. Outside this perimeter the influence of this process is considered null. The radius of this vicinity starts out big and keeps on getting smaller and smaller with the refining process.

The learning rate can have many expressions. In this application, the chosen expression was:

$$\eta(t) = \eta_0 \exp[-\left\| r_k - r_j^* \right\| / \sigma^2], \tag{13}$$

where r_j* and r_k are position vectors in the network representing the characteristic of the neural center of the vicinity and the neuron with the index k for which the refining process is taking place. Function $\eta_0=\eta_0(t)$ decrease in time, representing the value of the learning rate in the center of the vicinity:

$$\eta_0(t) = a / t^p, \tag{14}$$

The parameter σ controls the speed of decreasing the learning rate, depending on the radius of the vicinity.

After the refining process for the current input vector is finished the next one is selected and so on until all the input vectors are used and the stop training condition is inspected. A useful stopping condition is the moment when the weights of the network cease being refined (are no longer being modified):

$$\left\| w_{ij}(t+1) - w_{ij}(t) \right\| < \varepsilon, , \tag{15}$$

where i=0, 1, …, n-1 and j=0, 1, …, M-1.

The flowchart in Fig.4 summarizes the procedure of adapting and implementing the Kohonen Self-Organizing Map algorithm to the application proposed in the chapter. The differences from the standard theoretical algorithm are the same as those described for the Fuzzy Kwan-Cai algorithm in Fig. 2.

2.3 Bayes classifier

For the Bayes Classifier working with Gaussian classes (Larose, 2006) considering first the case of two (R=2) 1-dimensional classes (n=1), the density of probability being of Gaussian nature can be defined:

$$g_r(x) = p(x \mid \omega_r) \cdot P(\omega_r) = \frac{1}{\sqrt{2\pi} \cdot \sigma_r} \cdot e^{-\frac{(x-m_r)^2}{2\sigma_r^2}} \cdot P(\omega_r), \tag{16}$$

where parameter $r \in \{1;\ 2\}$.

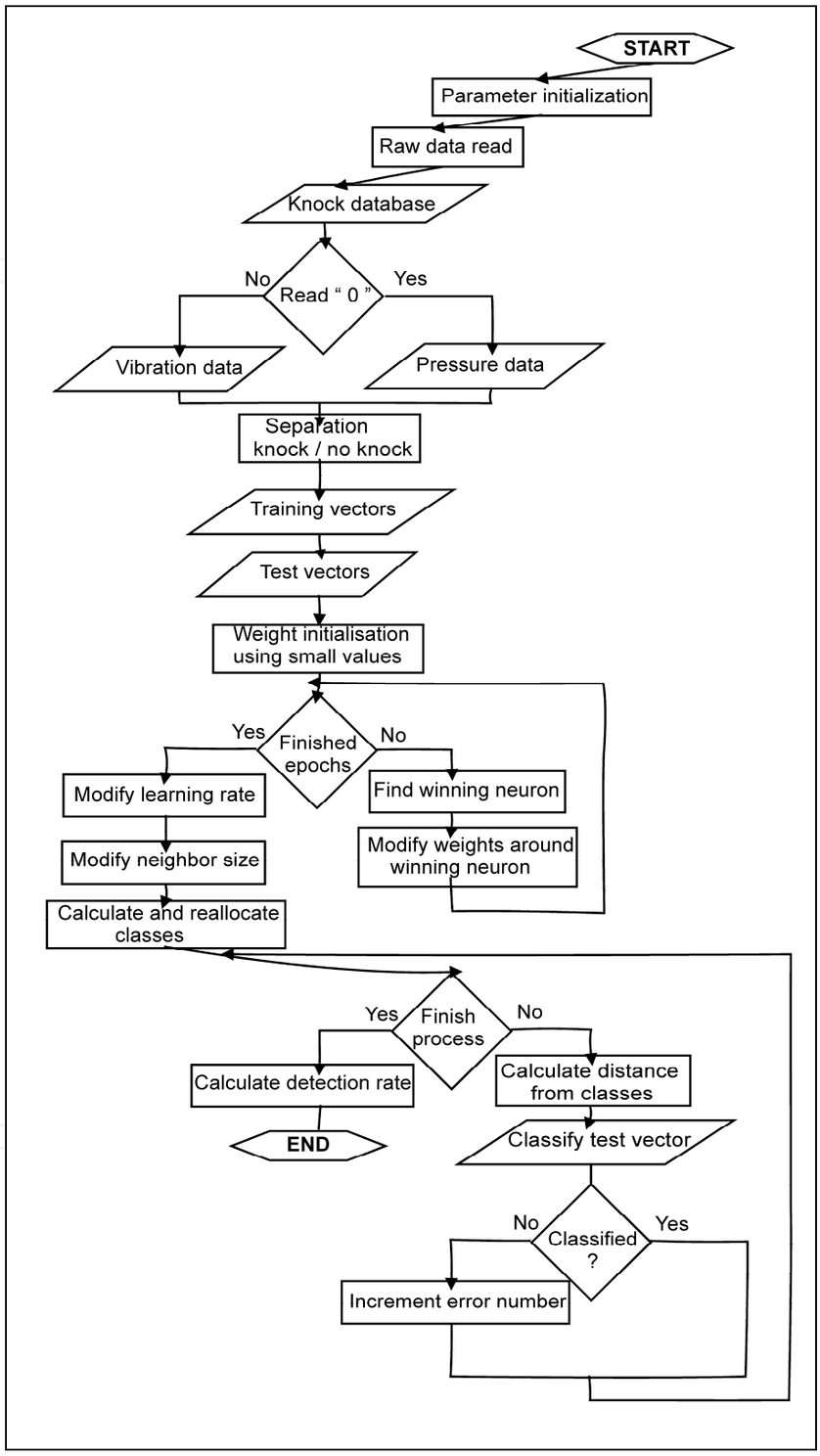


Fig. 4. Flowchart for implemented Kohonen Self- Organizing Map algorithm

Making an expansion to the n-dimensional case, the formula (16) for the Gaussian dispersion becomes:

$$p(x \mid \omega_r) = \frac{1}{(2\pi)^{n/2}|C_r|^{1/2}} e^{-\frac{1}{2}(x-m_r)^T C_j^{-1}(x-m_r)}, \quad (17)$$

where $m_r = E_r\{x\}$ represents the means of vectors in class r, $C_r = E_r\{(x-m_r)(x-m_r)^T\}$ is the matrix of covariance for the vectors in class r and $E_r\{\bullet\}$ is an operator that determines the mean value and that is used to make estimations concerning m_r and C_r based on a finite number N_r of patterns from ω_r. Their formulae are:

$$m_r = \frac{1}{N_r} \sum_{x \in \omega_r} x, \quad (18)$$

$$C_r = \frac{1}{N_r} \sum_{x \in \omega_r} xx^T - m_r m_r^T, \quad (19)$$

C_r being a positive semi-defined symmetrical matrix. The discriminant function based on the Gaussian density of probability will be:

$$g_r(x) = \ln P(\omega_r) - \frac{1}{2}\ln|C_j| - \frac{1}{2}[(x-m_r)^T C_r^{-1}(x-m_r)], \quad (20)$$

The flowchart in Fig.5 summarizes the procedure of adapting and implementing the Bayes Classifier algorithm to the application proposed in the chapter. The differences from the standard theoretical algorithm are the same as those described for the Fuzzy Kwan-Cai algorithm in Fig.2 and the Kohonen Self-Organizing Map in Fig.4.

3. Experimental results for each method

3.1 Methodology description, results and analysis

The algorithms treated in this chapter were tested on a Bosch Group database using two sizes of vector sample groups: one of 100 vectors and one of 1000, both of them containing pressure and vibration samples. In each case two thirds of the group was used for training and one third for testing.

The vectors that make up the database represent samples taken from petrol engines, some corresponding to knock situations and some not. The signals related to these samples were taken from pressure and vibration sensors mounted in and around the engine bay. The change in pressure caused by knock activity is seen as an immediate rise in pressure due to causes outside the normal engine piston cycle. On the other hand, the vibration sensors will detect vibrations – knocking noises – representing abnormal combustion fields being generated inside the pistons.

The applications have to declare knock or no knock for every sample vector received and, after testing the database, reach a verdict on the error of the process or in this case the identification rate. When knock is encountered actions can be taken to return the engine to a non-knock state.

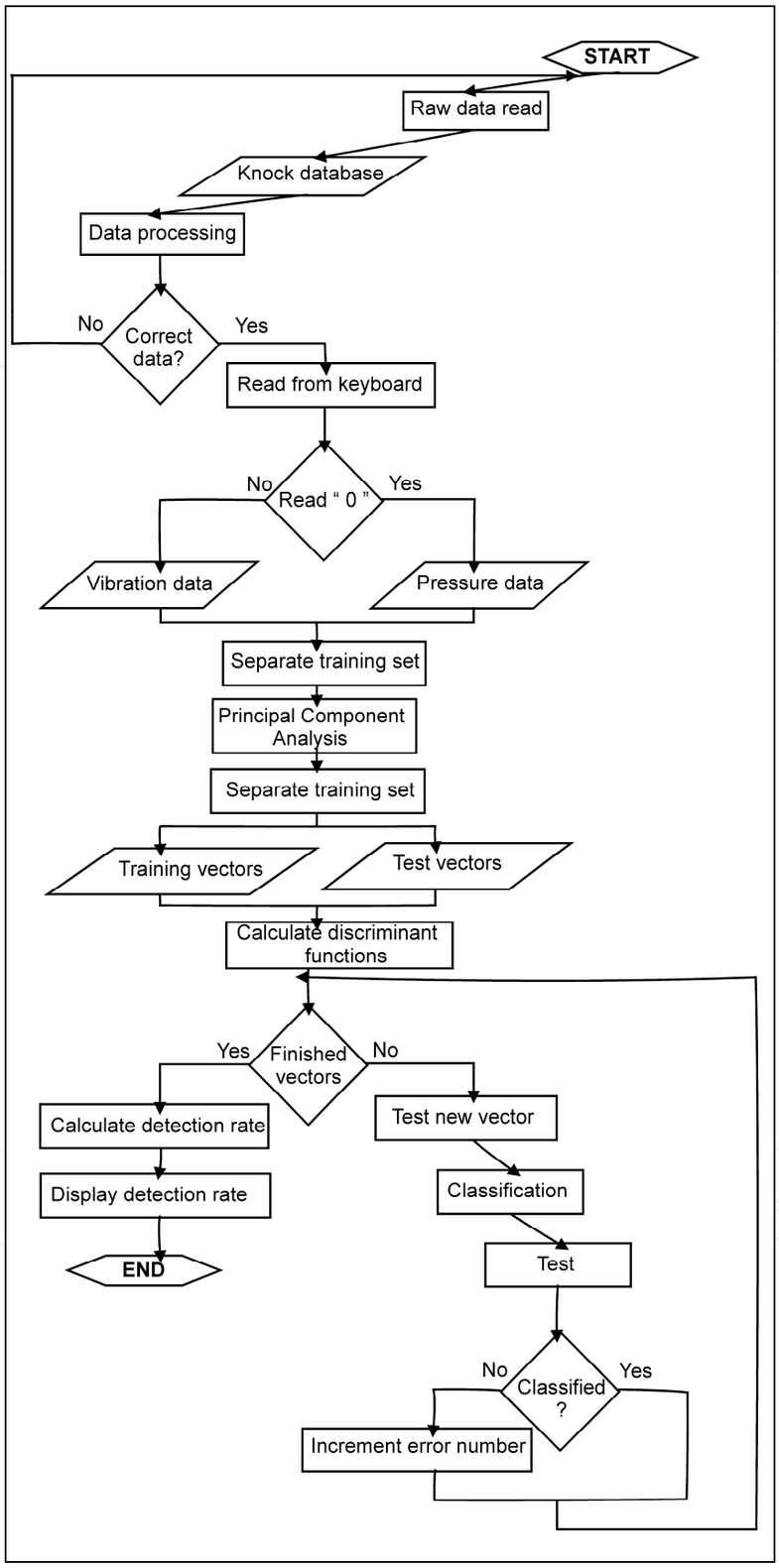


Fig. 5. Flowchart for implemented Bayes Classifier

The testing method for both algorithms (Fuzzy Kwan-Cai and Kohonen Self-Organizing Map) is the following: one parameter varies between its theoretical limits whereas the others remain constant. It is obvious that the difference between the bigger training group and the smaller one should be the higher detection rate.

The following tables contain only the significant part of the experimental results in order to outline the highest detection rates obtained.

3.2 Fuzzy Kwan-Cai neural network results

This type of neural network does not need training cycles because it learns as it studies the vectors it receives and builds its own classes in the testing process. In order not to get the wrong idea from the start we have to mention that the high number of classes observed in Table I and Table II for this neural network is due to the second nature of the application which acts like a "focusing lens", examining the internal structure of the two main classes. Therefore it must be stated that the number of classes we are interested in, for this experiment, is two. The significance and proper function limits of this application for parameters given in Table I and Table II are: α which is the base of isosceles triangles ($\alpha \in [1.5; 3.5]$), β that determines how much of the sample vector each fuzzy neuron sees ($\beta \in [0.1; 1.6]$) and Tf that represents the neural network's sensitivity to errors ($Tf \in [0.1; 0.35]$).

The first vector generates a class of its own, the next ones either are found relatives of one of the vectors that have come before and therefore are put in the same class or start a new class. The maximum detection results in the tables mentioned above are outlined by being bolded. Unsatisfactory results with high detection rates are presented in italic.

Tables Ia and Ib present the pressure sample detection rate results for the Fuzzy Kwan-Cai neural network using the small sample database and the large sample database. According to Table Ia, the highest detection rate value of 68% was obtained for combination (3.5; 0.15; 1) where parameters Tf and β are kept constant whereas α varies.

The same method has been used for Table Ib showing the combinations used by changing the parameter Tf while keeping constant the other two. Combinations are from (3.5; 0.35; 1) down to (3.5; 0.15; 1). A maximum correct detection rate of 93.40% was obtained for the (3.5; 0.22; 1) group.

The detection rate results in Tables Ia and Ib show that from this point of view the Fuzzy Kwan-Cai neural network is very stabile, small variations of its parameters not affecting the experimental outcome. It is clear from the results presented that an increase in the sample database leads to an increase in the detection rates, the network not being affected by sample vectors that are not cohesive in nature with the rest of their class.

Tables IIa and IIb contain the vibration sample detection results. Table IIa represents the small sample database and Table IIb the large one. Table IIa uses the same method of parameter variation as Tables Ia and Ib but valid variations are not achieved because for a result to be considered satisfactory it should at least be higher than 50%.

The first part of Table IIb contains results obtained by using combinations in the same way as Tables Ia, Ib and IIa, the parameter that varies being Tf whereas the others are kept constant. Used combinations start at (3.5; 0.35; 1) and end at (3.5; 0.15; 1). In this first set the

α	Tf	β	Rate [%]	No. classes
3.5	0.15	1	68%	2
3.4	0.15	1	64%	4
3.3	0.15	1	64%	4
3.2	0.15	1	64%	4
3.1	0.15	1	64%	4
3	0.15	1	48%	5
2.9	0.15	1	48%	5
2.8	0.15	1	48%	7
2.7	0.15	1	48%	7
2.6	0.15	1	48%	7
2.5	0.15	1	72%	9
2.4	0.15	1	60%	9
2.3	0.15	1	58%	11
2.2	0.15	1	58%	11
2.1	0.15	1	58%	11
2	0.15	1	68%	12

Table Ia. Pressure detection rates- small database

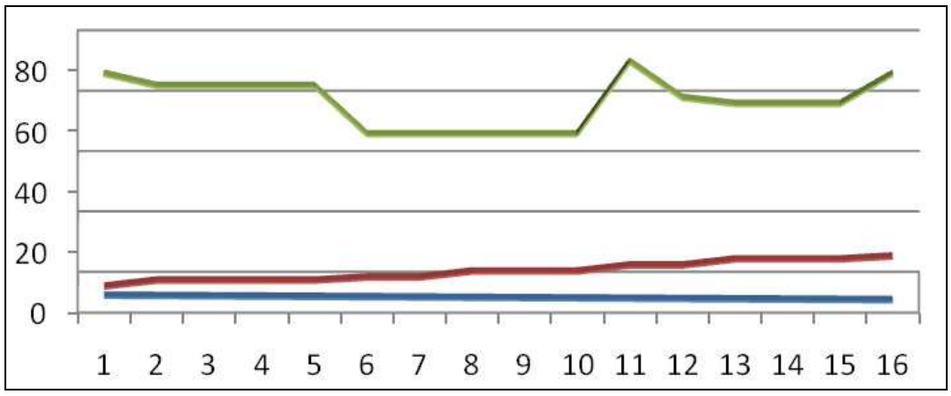


Fig. 6. Plot of α (blue) ,Rate[%](green) and No. classes (red) (Table Ia)

α	Tf	β	Rate [%]	No. classes
3.5	0.35÷0.23	1	93.40%	1
3.5	0.22	1	93.40%	2
3.5	0.21	1	93.40%	3
3.5	0.2	1	93.40%	3
3.5	0.19	1	93.40%	4
3.5	0.18	1	93.40%	5
3.5	0.17	1	93.40%	5
3.5	0.16	1	93.40%	8
3.5	0.15	1	93.40%	10
3.5	0.35÷0.23	1	93.40%	1

Table Ib. Pressure detection rates- large database

α	Tf	β	Rate [%]	No.classes
3.5	0.35÷0.29	1	48%	2
3.5	0.28	1	48%	3
3.5	0.27	1	62%	6
3.5	0.26	1	62%	6
3.5	*0.25*	*1*	*68%*	*9*
3.5	*0.24*	*1*	*68%*	*9*
3.5	0.23	1	62%	9
3.3	0.3	1	48%	3
3.3	0.3	0.9	48%	3
3.3	0.3	0.8	48%	3
3.3	0.3	0.7÷0.3	48%	2
3.3	0.3	0.2	48%	1

Table IIa. Vibration detection rates- small database

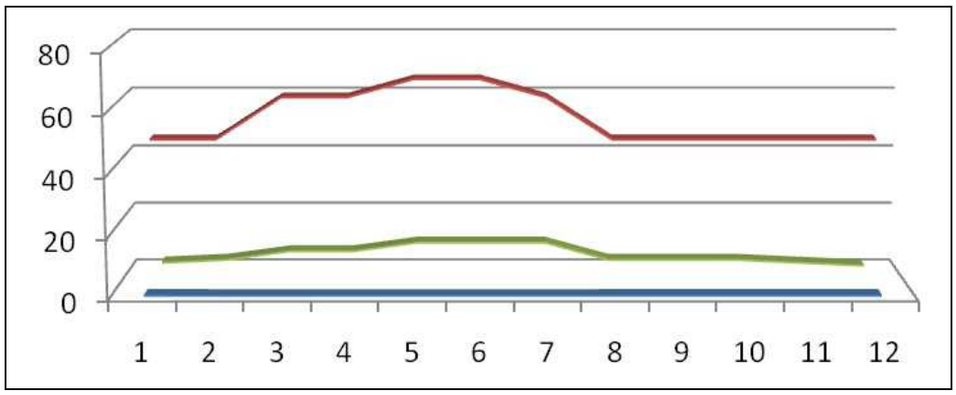


Fig. 7. Plot of Tf (blue),Rate[%] (red) and No.classes (green) (Table IIa)

α	Tf	β	Rate [%]	No. classes
3.5	0.35÷0.24	1	93.40%	1
3.5	**0.24**	**1**	**93.40%**	**2**
3.5	0.23	1	93.40%	3
3.5	0.22	1	93.40%	5
3.5	0.21	1	93.40%	9
3.5	0.2	1	82.05%	10
3.5	0.19	1	93.13%	32
3.5	0.18	1	92.34%	57
3.5	0.17	1	90.23%	107
3.5	0.16	1	85.75%	143
3.5	0.15	1	85.75%	200
3.3	0.3	1	85.10%	3
3.3	0.3	0.9	85.10%	3
3.3	0.3	0.8	85.10%	3
3.3	**0.3**	**0.7÷0.3**	**85.10%**	**2**
3.3	0.3	0.2	85.10%	1

Table IIb. Vibration detection rates- large database

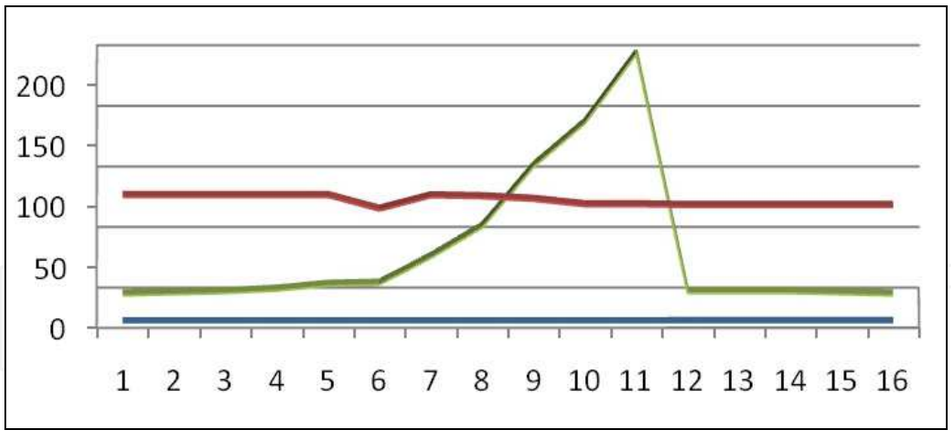


Fig. 8. Plot of Tf (blue) ,Rate[%](red) and No.classes (green) (Table IIb)

maximum correct detection rate of 93.40% is achieved for (3.5; 0.24; 1) – values bolded. Set two contains combinations from (3.3; 0.3; 1) down to (3.3; 0.3; 0.2), parameter β varying between 1 and 0.2 and the other two staying constant. A detection value not as high but equally as important as the maximum one obtained in the previous set is showed in combinations from (3.3; 0.3; 0.7) to (3.3; 0.3; 0.3). The value is 85.10% and presents interest because it is a much higher value than the ones constantly obtained and also represents a correct class detection of two classes.

The vibration situation presented in Tables IIa and IIb leads us to the same results revealed by Tables Ia and Ib, that an increase in the database size will lead to a substantial increase in the detection rate.

In the case of the large sample group shown in Tables Ia and Ib, respectively in Tables IIa and IIb, the neural network does not show any difference in maximum detection rates, differences being observed only for the small sample group. Both tables also present the same maximum detection rate, showing that the network can learn to identify both types of samples with the same accuracy.

Table III presents the time situation. It contains the average detection time situation for both pressure and vibration samples and also from a small and large database point of view. It is clear that the large database obtains better results with almost equally small detection times – 0.0022s for pressure and 0.0046s for vibration – and that pressure vectors have the tendency of being faster detected than vibration ones because the pressure group is more coherent and homogenous than vibration group.

Average detection time [s]	Pressure	Vibration
Large database	**0.0022**	**0.0046**
Small database	0.0052	0.0056

Table III. Average detection times representing pressure and vibration for both small and large databases

What can be observed from the start is that the bigger sample group has almost equal detection times in both pressure and vibration cases to the smaller group, a significant increase being shown in the detection rates. The average detection times in Table III show that via optimization the network can be used in real–time knock applications with very good detection rates and with no prior in-factory learning processes.

One can observe for the Fuzzy Kwan-Cai algorithm that different combinations of parameters can produce the same detection rates, so that a linear variation in any of the parameters will not always lead to a linear variation in the detection rate.

3.3 Kohonen Self–Organizing Map neural network results

The Kohonen–Self Organizing Map has a separate learning stage taking place before the detection process begins and being composed of epochs. After the learning stage has ended it does not need to be repeated and the processing of the test batch begins.

For this neural network three sizes of neural maps were used – nine, one-hundred and four-hundred neurons –, as shown in Tables IV, V, VI. They were tested on both pressure and vibration samples.

Table IVa contains only the pressure sample detection rate results for the small vector database using the one hundred–neuron configuration. By keeping the number of epochs constant at 100 and the learning rate at 0.2 and by means of a variation of the neighborhood size from 90 down to 10, we obtained the following spike values: a detection rate of 80% marked bold-italic for the (100; 0.2; 83) group and the maximum value of the detection rate for the small database 82.85% marked bold for the (100; 0.2; 82) combination.

Table IVb contains the pressure sample detection rates using the large database. From the start, using the nine-neuron map, an important fact appears: the nine-neuron map can not cope with the large database due to the small number of neurons that have to remember a large amount of samples, leading to confusion and very low detection rates. The variation methods are the same ones as in the complete version of Table IVa but, even by varying each of the parameters and keeping the other two constant, we can not obtain a spike value higher than 29.78% marked italic, value resulting from the combination (100; 0.4; 5). Performing the same variation techniques as in Table IVa, the maximum value for the detection rate in Table IVb results of 90.57% from the (400; 0.2; 400) and (500; 0.2; 400) combinations – both marked bold - , with lower but not less important spikes of 89.66% for (100; 0.2; 400) and (100; 0.3; 400) – marked bold-italic.

Table Va contains the vibration sample detection rates for the small database. The same variation methods as those in Tables IVa and IVb were used for the exact same values. The one-hundred-neuron network encounters its top value of 80% for the (100; 0.2; 95) combination and also a smaller spike of 74.28% for (100; 0.2; 60). The four-hundred-neuron network tops out at the 82.85% detection rate for the (300; 0.2; 400) combination of parameters. The same marking methods as in the previous tables were also used here and in the following ones.

The large database results for the vibration sample vectors are found in Table Vb. These values have come from the same methods of testing and values used in Tables IVa, IVb and Va. As in the case of the complete Table IVa (from which only the one-hundred neuron

No. neurons	Epochs	Learning rate	Neighborhood	Rate [%]
100	100	0.2	90	68.57
100	***100***	***0.2***	***83***	***80***
100	**100**	**0.2**	**82**	**82.85**
100	100	0.2	80	77.14
100	100	0.2	70	74.28
100	100	0.2	60	68.57
100	100	0.2	50	71.42
100	100	0.2	40	71.42
100	100	0.2	30	71.42
100	100	0.2	20	77.14
100	100	0.2	10	65.71

Table IVa. Pressure detection rates- small database

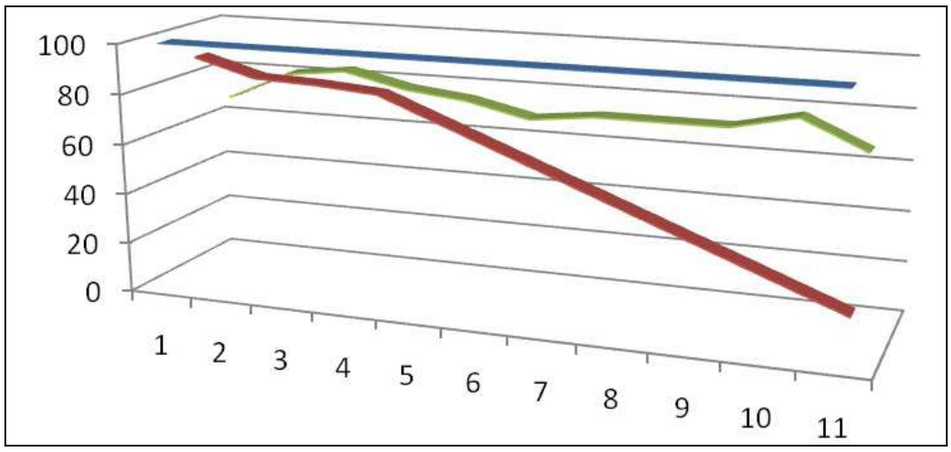


Fig. 9. Plot of No. Neurons (blue), Neighborhood (red) and Rate[%] (green)(Table IVa)

No. neurons	Epochs	Learning rate	Neighborhood	Rate [%]
9	100	0.2	5	23.03
9	100	0.3	5	27.35
9	*100*	*0.4*	*5*	*29.78*
9	*100*	*0.5*	*5*	*28.57*
9	100	0.6	5	19.75
9	100	0.7	5	20.06
400	***100***	***0.2***	***400***	***89.66***
400	***100***	***0.3***	***400***	***89.96***
400	100	0.4	400	87.84
400	100	0.5	400	88.75
400	100	0.6	400	89.96
400	100	0.7	400	88.75
400	***100***	***0.2***	***400***	***89.66***
400	200	0.2	400	89.36
400	300	0.2	400	88.75
400	**400**	**0.2**	**400**	**90.57**
400	**500**	**0.2**	**400**	**90.57**

Table IVb. Pressure detection rates- large database

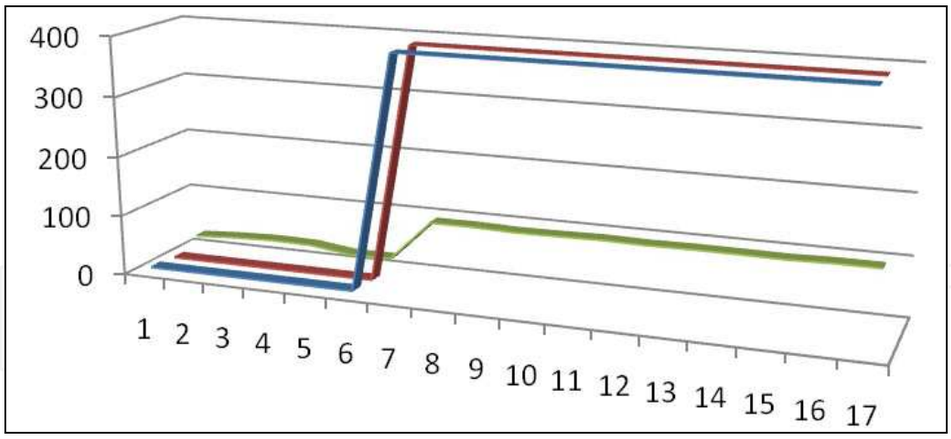


Fig. 10. Plot of No. Neurons (blue), Neighborhood (red) and Rate[%] (green)(Table IVb)

No. neurons	Epochs	Learning rate	Neighborhood	Rate [%]
100	**100**	**0.2**	**95**	**80**
100	100	0.2	90	65.71
100	100	0.2	80	65.71
100	100	0.2	70	57.14
100	**100**	**0.2**	**60**	**74.28**
100	100	0.2	50	65.71
100	100	0.2	40	65.71
100	100	0.2	30	68.57
100	100	0.2	20	60
400	100	0.2	400	65.71
400	200	0.2	400	71.42
400	**300**	**0.2**	**400**	**82.85**
400	400	0.2	400	65.71
400	500	0.2	400	62.85
400	600	0.2	400	71.42

Table Va. Vibration detection rates- small database

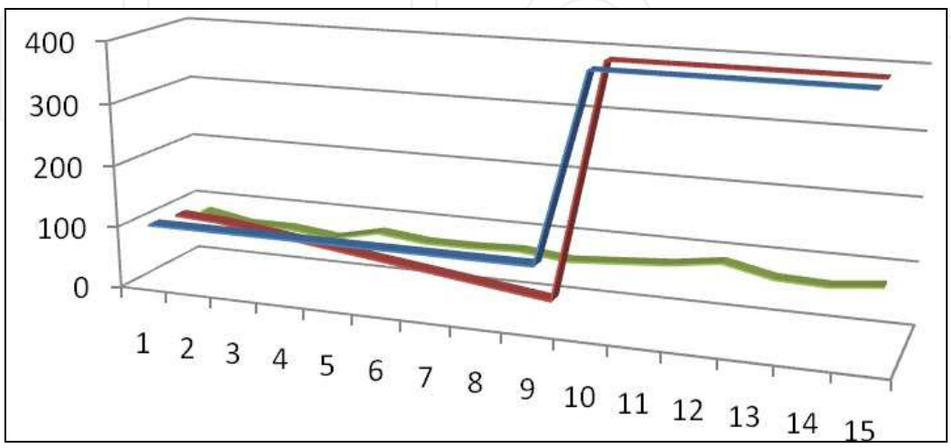


Fig. 11. Plot No. Neurons (blue) Neighborhood(red) and Rate[%] green)(Table Va)

No. neurons	Epochs	Learning rate	Neighborhood	Rate [%]
100	100	0.2	95	79.63
100	100	0.2	90	79.93
100	100	0.2	80	79.02
100	***100***	***0.2***	***70***	***81.15***
100	100	0.2	60	78.11
100	**100**	**0.2**	**50**	**81.76**
100	100	0.2	40	75.98
100	100	0.2	30	79.93
100	100	0.2	20	76.59
400	100	0.2	400	87.53
400	***100***	***0.2***	***375***	***89.05***
400	100	0.2	350	88.75
400	***100***	***0.2***	***325***	***89.36***
400	100	0.2	300	86.83
400	100	0.2	275	86.62
400	**100**	**0.2**	**250**	**89.66**
400	100	0.2	225	88.75
400	100	0.2	200	88.44
400	100	0.2	175	88.44

Table Vb. Vibration detection rates- large database

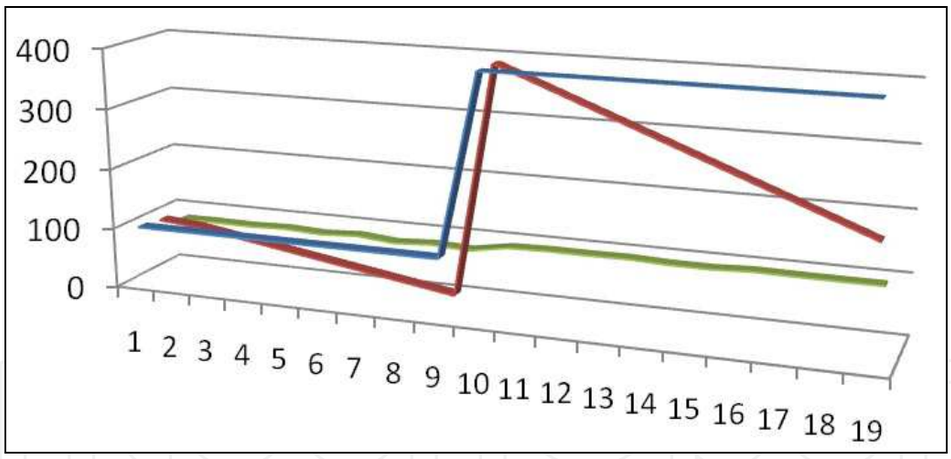


Fig. 12. Plot of No. Neurons (blue), Neighborhood (red) and Rate[%] (green) (Table Vb)

section has been presented in this chapter), the nine-neuron network in the complete Table Vb is not suited for working with such a large database, the network becoming confused. This shows in constant results under 50% which can not be taken into account as valid experimental results. These values can only be used as examples of exceptional cases. The one-hundred-neuron network section presented in Table Vb obtains a maximum detection rate of 81.76% for combinations (100; 0.2; 50), another important value over 80% being of 81.15 % for (100; 0.2; 70) . The four-hundred-neuron network tops out at 89.66% for combinations (100; 0.2; 250) and present other important values of 89.36% for (100; 0.2, 325) and of 89,05% for (100; 0.2; 375).

Table VI represent the average detection times using both pressure and vibration vectors for both small and large databases. With values of 0.0023s (small database) and 0.0024s (large database) the pressure samples obtain smaller detection times than the vibration samples with 0.0027s (small database) and 0.0028s (large database). This situation is representative for the four-hundred-neuron network, this also being the slowest solution but with the highest detection rates. The nine-neuron network, even though it has the best detection times, can not be taken into account as a real application because it is not able to cope with large database. The one-hundred-neuron network is the best compromise between detection speed and detection rates as shown in this table.

As with the previous described algorithms, the SOM results shown in Tables IV and V that an increase in the sample group size (training set case) will lead to an increase in detection rates. In this case, the two separate groups are not separated by big detection rate gaps.

Average detection time [s]	Small database		Large database	
	Pressure samples	Vibration samples	Pressure samples	Vibration samples
SOM - 400 neurons	0.0023	0.0027	0.0024	0.0028
SOM - 100 neurons	**0.000193**	**0.000478**	**0.000538**	**0.000498**
SOM - 9 neurons	0.0000576	0.0000579	0.0000535	0.0000872

Table VI. Pressure and vibration average detection times for both small and large sample databases

As in theory, the experimental results in Tables IV, V and VI show that with the increase in neurons there is an increase in detection rates but a decrease in detection times because more neurons translate to more detail that can be remembered, so the distinction between knock and non-knock situations can be more precisely done - therefore a compromise must be made. Being interested not only in obtaining high detection rates but also detection times that would be coherent to the task at hand (samples must be processed in under an engine cycle so the modifications can be brought to the next one), the one-hundred-neuron map seems to be the best option from the three methods tested. The nine-neuron map, even if it produces very high detection times, has a very poor detection rate in both pressure and vibration groups making it useless for any further applications.

The four-hundred-neuron map presented the highest detection rates for this neural network, values that are a little bit smaller than the Fuzzy Kwan-Cai but with detection times very similar to it, the only difference being that the SOM needs separate training. In this case, looking at the detection times in Table VI, the SOM does not seem to make any difference between pressure and vibration signals, the medium detection times showing very small variations. There is a small difference in detection rates between pressure and vibration samples; the SOM seems to handle both models very well.

A very important factor in the good working of the Kohonen Self-Organizing Map is getting the number of epochs and the learning rate well calibrated. A greater than necessary number of epochs would lead to the situation where the network learns in the necessary time period but it is left with more epochs that are not used for learning. This situation, in combination with a high learning rate, would lead to the network learning everything very fast in the first epochs and then forgetting or distorting the knowledge in the following ones.

3.4 Bayes classifier results

The Bayes Classifier, as described by its name, is not a neural network but has been included in this chapter as a basic reference point for the evaluation of the two neural networks. It uses a method of calculating the minimum distance from a sample to one of the knock or non-knock class centers - classes that are considered Gaussian by nature. That is why it presents the worst detection times, as shown in Table VIII.

Table VIIa represents the combined pressure and vibration detection rates status for the small database. The way the testing has been done for this algorithm is by progressively growing from a small comparison group (the batch of samples chosen to represent the known classes for testing) versus large test group situation, to a large comparison group versus small test group situation.

The process starts out with a balance of 11 training vectors and 90 testing ones, which leads to a detection rate starting from 65.50% for pressure and 55.55% for vibration and grows (for training vectors) versus shrinks (for testing vectors) in a progressive way to 85 training vectors and 16 testing vectors, leading to a detection rate ending at 43.75% for pressure and 81.25% for vibration. An interesting detail can be observed in this table: the pressure vectors seem to present a constant state even though more and more are added to the learning group every time the detection rates stay approximately between 50% and 72.50%, the last value being the highest pressure detection rate.

The change of state occurs at the end of the table where we can observe a decrease in the learning rate for the combinations of (80 training vectors; 21 testing vectors) with a detection rate of 42.85% and (85 training vectors; 16 testing vectors) with a detection rate of 43.75%.

This decrease is due to the inclusion in the learning group of vectors that are radically different from their stated class; therefore, the knock or non-knock distinction can not be made. In the case of the vibration sample vectors the progression is of almost uniform growth from 55.55% to 81.25%, the last being also the maximum detection rate for the small database experiment.

Table VIIb follows the same type of progression, only that the large database is used for both pressure and vibration samples. The progression goes from a combination of (371 training vectors; 629 testing vectors) with a detection rate of 93.64% for pressure and 90.30% for vibration samples to a combination of (671 training vectors; 329 testing vectors) with the maximum detection rate achieved in this table of 95.44% for pressure samples and 92.40% for vibration samples. Within this progression it can be seen more clearly that the pressure samples are very cohesive in nature and that, given enough samples, the algorithm goes past the problems it has with radically different sample vectors, maintaining a detection rate over 90% in every case.

Table VIII represents the average detection times for both the small and large databases using both pressure and vibration samples.

Being a simple comparative algorithm, we can see in Table VIII that an increase in the database size leads to a slowing down of the process because the comparison must be made with more vectors. In the case of the small database, pressure vectors are detected faster

Training vectors	Test vectors	Press. rate [%]	Vib. rate [%]
11	90	65.50	55.55
12	89	56.17	58.42
13	**88**	**72.50**	**60.22**
15	86	51.16	53.48
21	80	63.75	66.25
28	73	54.79	64.38
35	66	59.09	60.60
41	60	68.33	70
47	54	59.25	68.51
55	46	56.52	73.91
61	40	67.50	70
65	36	52.77	77.77
67	34	68.57	74.28
75	26	50	76.92
80	21	42.85	76.19

Table VIIa. Pressure - vibration detection rates- small database

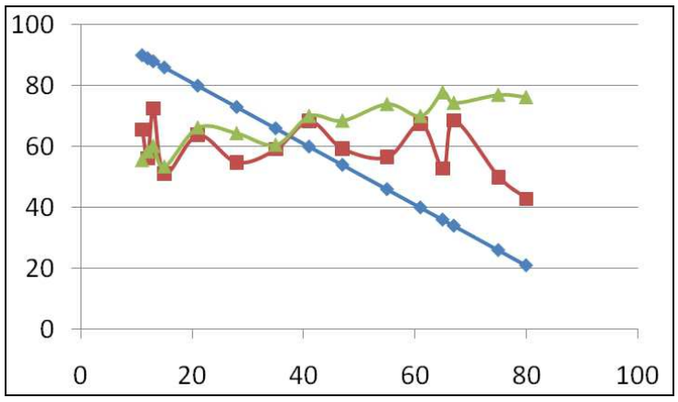


Fig. 13. Test vectors (blue), pressure (red) and vibration rates(green) (Table VIIa)

Training vectors	Test vectors	Press. rate [%]	Vib. rate [%]
371	**629**	**93.64**	**90.30**
391	609	93.43	90.14
411	589	93.20	90.32
431	569	92.97	89.98
451	549	92.71	89.79
471	529	92.43	89.60
491	509	92.14	89.58
511	489	91.82	89.77
531	469	91.42	89.55
551	**449**	**91.09**	**89.08**
571	**429**	**90.67**	**89.04**
591	**409**	**91.44**	**89.48**
611	389	92.28	90.23
631	369	93.22	91.32
651	349	94.26	92.26

Table VIIb. Pressure - vibration detection rates- large database

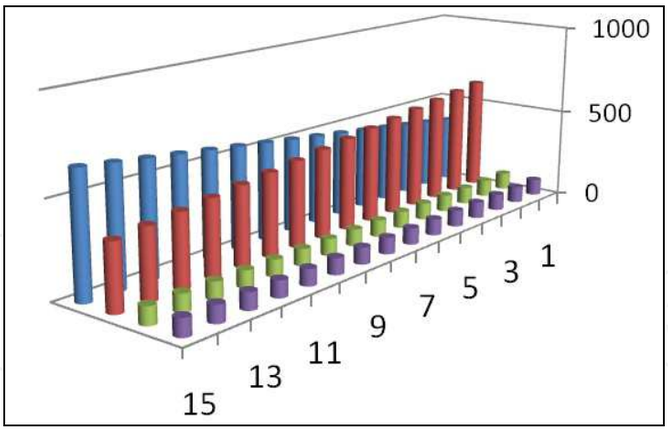


Fig. 14. Test vectors (red), training vectors (blue), pressure rates (green) and vibration rates (violet)

Average detection time [s]	Pressure	Vibration
Small sample database	**0.0287**	0.0297
Large sample database	0.0948	**0.094**

Table VIII. Pressure and vibration average detection times for both small and large sample databases

(0.0287s) than vibration samples (0.0297s). The large database experiments lead to almost equal average detection times between pressure (0.0948s) and vibration (0.094s) samples, with a tendency to better recognize vibration samples.

There is little relevance in the detection rates for the small sample group, even though a small variation between pressure and vibration can be seen. The increase in detection rates due to a bigger knowledge database can also be seen from Tables VIIa and VIIb.

The greatest importance of the Bayes Classifier in this chapter comes from its great sensitivity to change. When the knowledge group includes vectors that are incoherent with the others or that are more different, the detection rate goes down immediately. In this case, the algorithm can not classify properly because one or both classes contain vectors that are very far away from their centers and vectors from one class may get tangled up with the other one. By doing this the Bayes Classifier acts as a monitor for change in the constitution of the sample classes or a "magnifying glass" reacting to the internal composition of the data groups.

Given a big enough knowledge database that is also very coherent in the nature of its classes, the detection rates go up and can be comparable to the neural networks but at a great cost in speed.

4. Comparison among the three tested methods

The first discussion will be based on the database size point of view. As we can see from Fig.15 and Fig.16 that summarize results in Tables I, II, IV, V and VII, the size of the learning, training or comparison database is very important in the good functioning of all three tested algorithms.

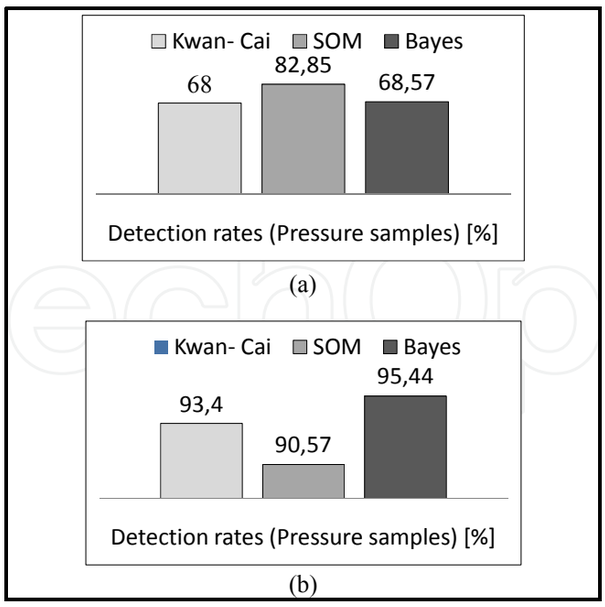


Fig. 15. Pressure sample detection rates using the small database (a) and the large database (b) for the Kwan- Cai, SOM neural networks and the Bayes Classifier

An increase in the database size from one hundred to one thousand sample vectors will lead to a minimum increase of ten percent in the detection rates. For the small database, the Fuzzy Kwan-Cai neural network obtains maximum detection rates for the pressure samples at 68% that are higher than the ones for vibration samples at 48%, but after using the large data set the maximum pressure and vibration detection rates become equal at 93.40%. The difference in detection rates for the pressure and vibration samples using the small database shows that the pressure samples are more coherent and therefore easier to classify. The same evolution as shown by the Fuzzy Kwan- Cai is also true for the Kohonen Self-Organizing Map (SOM). Even more so, the increase in learning database size will lead to a theoretical increase in the detection rate of the Bayes Classifier.

The second discussion will be based on the detection rate point of view. As shown in Fig.15 and Fig.16, the Bayes Classifier seems to show the best detection rates. Its fault is that it needs large amounts of comparison data in order to create classes that are comprehensive enough. Out of the three algorithms tested in this chapter, it is also the less stabile due to the fact that it calculates distances to the center of the comparison classes. If these classes are not well defined and separated, the detection rates fall dramatically. This can be seen in Table VIIb. The Fuzzy Kwan-Cai obtains the highest detection rates of all three algorithms - these being valid detection rates that are not influenced by the nature of learned vectors leading to the great stability of this method. The learning method used employs the automatic generation of learning classes as it goes through the sample set. The fuzzy logic creates a more organic representation of the knowledge classes than the boolean one. The Kohonen Self-Organizing Map (SOM) presents the second highest detection rates and a more controlled and stabile learning and training environment then the other two algorithms. Because the learning is done prior to the start of the testing process and in repetitive epochs, the neural network has the chance to go through the data set again and again until a

complete image is formed. The two neural networks show no considerable preference between pressure and vibration samples and present high stability to drastic variations in training samples which in a non-neural method could cause a decrease in detection rates. The nature of these types of signals and their differences are outlined by the Bayes Classifiers sensitivity to unclear classes and the way in which the Fuzzy Kwan-Cai neural network works by showing the internal structure of the classes.

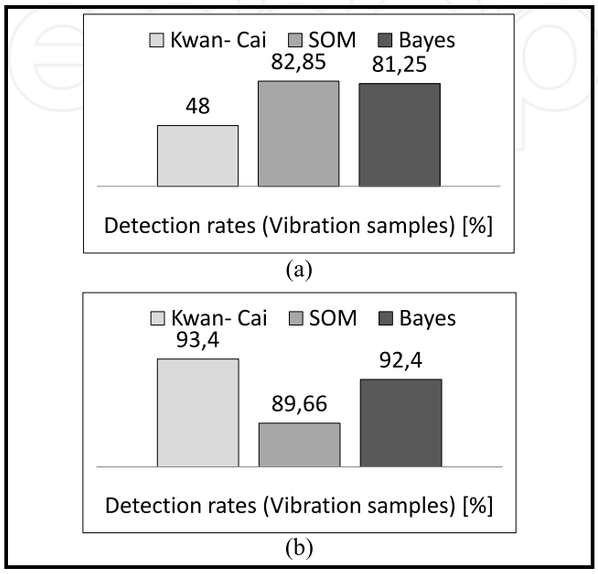


Fig. 16. Vibration sample detection rates using the small database (a) and the large database (b) for the Kwan-Cai, SOM neural networks and the Bayes Classifier

The third discussion will be based on the detection time point of view. As present in Fig.17 and Fig.18 that summarize results in Tables III, VI and VIII, it is clear at first glance that the neural networks are far superior to the normal non-neural classification algorithm. The Bayes Classifier obtains the longest detection times due to the process of comparing each new vector to the knowledge classes. The best, valid, detection times are shown by the Kohonen Self-Organizing Map with the one-hundred-neurons configuration. This configuration, given optimization of the code, can lead to detection times coherent to the engine combustion cycles in which the knock detection needs to take place. Any number of neurons under one hundred will make it hard for the network to give satisfactory detection rates even though the detection times will decrease dramatically. In this chapter we are interested in maximizing the balance between high detection rates and low detection times and not achieving the two extremes and having to compromise one outcome. The second best detection times that are also very close to one another belong to the Fuzzy Kwan-Cai and SOM with the configuration of four-hundred-neurons.

These two algorithms also show the highest detection rates from the methods tested in this chapter. In a real-time application there should not be any problem with the SOMs separate training stage because it would be performed only once inside the factory. The Fuzzy Kwan-Cai neural network presents a different advantage in that it can learn as it goes along, not needing a separate training stage and continuously receiving information and gaining knowledge.

It is clear from the information presented in this chapter that the best detection rates correlated to very good detection times belong to the Kohonen Self-Organizing Map with a configuration of one-hundred-neurons.

The SOM with a configuration of four-hundred-neurons obtains results almost similar to the Fuzzy Kwan-Cai. The difference between these two networks is that the SOM requires a separate training stage where the separated and well defined learning classes are given to it and the Fuzzy Kwan-Cai learns as it receives sample vectors and builds its own classes.

The Bayes Classifier is very useful for showing the nature of the knock and non-knock classes how well they are defined and separated due to its sensitivity to drastic variations in sample vectors. Its detection rate depends on the size of the knowledge database and its coherence making it useless in real-world applications.

From a real-world application point of view, in order to further maximize detection rates, it is clear that a parallel process composed of a pressure-vibration analysis and detection becomes necessary, based on the experimental results. Due to the developments in digital signal processing (DSP) technology, the parallel process would not lead to an increasing detection times.

5. Concluding remarks

In order to avoid overcrowding, this final chapter contains general concluding remarks due to the fact that details and accurate conclusions have already been widely presented in chapters III and IV above.

Three methods of knock detection were studied and compared in this chapter. Testing was performed on a Bosch Group database. Two of the three algorithms used are of neural nature: Fuzzy Kwan-Cai neural network – presenting the unsupervised learning approach and fuzzy inference core - and Kohonen Self-Organizing Map (SOM) – with a separate supervised learning stage - and the third is non-neural: Bayes Classifier.

The three algorithms were either trained or had comparison classes and were tested on two different database sizes, one small of one hundred samples vectors and one large representing one thousand samples in order to show how the database size would affect the detection outcome.

Experiments were made on both pressure and vibration sample vectors in order to see which of these are more coherent in nature, leading to results that show an overall greater coherence with slightly more increased detection rates and how this coherence might affect the algorithms being tested. The experiments performed have led to results that prove the superiority of the neural methods in contrast to the normal classification – the situation being looked at from a rate-time point of view as seen in Fig.15, Fig.16, Fig.17, Fig.18.The difference between the neural and non neural methods is represented by an average scale factor of 0,001s in favour of the neural. This superiority should be seen also from a stability to errors point of view as seen in Table VIIb where a stray vector can distort the judgement of the non neural Bayes Classifier so that detection rates fall.

Comparisons were made between the algorithms leading to experimental results enabling us to draw conclusions on which methods are superior to others, in what way and also on the properties and nature of the database used in the experiments.

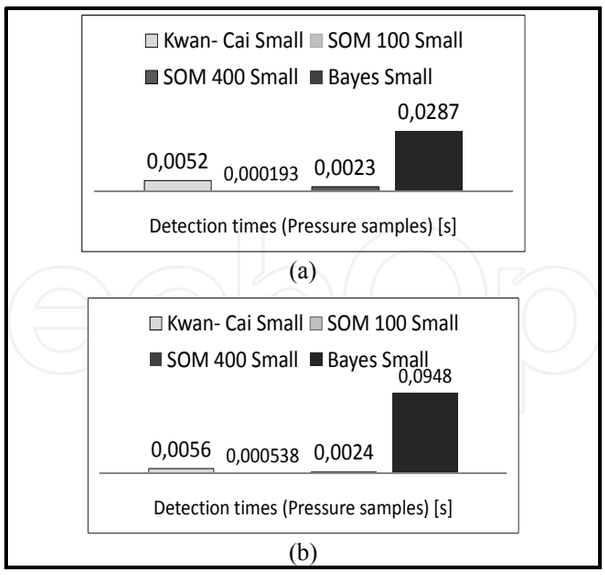


Fig. 17. Pressure sample detection times using the small database (a) and the large database (b) for the Kwan- Cai, SOM neural networks and the Bayes Classifier

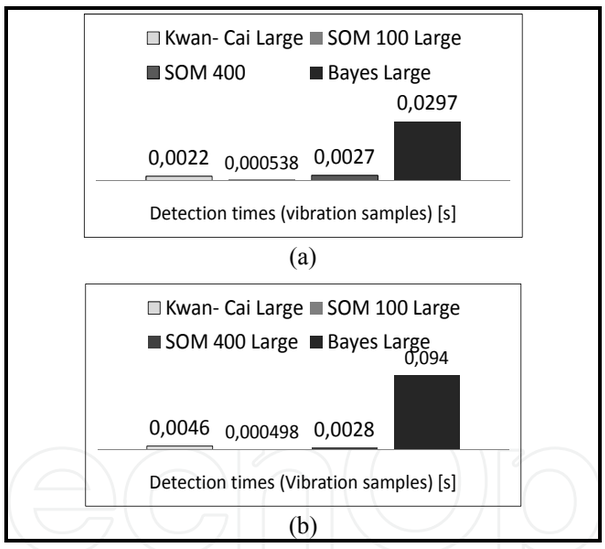


Fig. 18. Vibration sample detection times using the small database (a) and the large database (b) for the Kwan- Cai, SOM neural networks and the Bayes Classifier

Suggestions for real-world applications were made in the prior chapter leading to further optimizations around the strengths and weaknesses of each algorithm.

The three algorithms and most of all the two neural networks have long been used for varied applications showing great robustness and stability. The versions of these applications used in this paper are presented and have been used and tested in their standard form as presented in (Kohonen, 2000, 2002) and (Kwan&Cai, 1994) using as method of verification direct comparison of the outcome of detection and the optimal

known value for each vector at a time and incremented into an error counter. The databases were verified to be consistent of their description.

6. Acknowledgement

This work was supported by CNCSIS – UEFISCSU, project number PNII – IDEI code 1693/2008.

7. References

Adeli, H. & Karim , A. (2005). Wavelets in Intelligent Transportation Systems (1st edition) , Ed. Wiley, ISBN-13: 978-0470867426, England

Auld, T.; Moore, A.W. & Gull, S.F. (2007). Bayesian Neural Networks for Internet Traffic Classification, vol. 18, issue. 1, pp. 223–239, ISSN: 1045-9227

Banakar A. & Azeem M. F. (2008). Artificial wavelet neural network and its application in neuro-fuzzy models, Appl. Soft Comput., vol. 8, no. 4, pp. 1463–1485, ISSN: 1568-4946

Billings, S.A. & Wei H.L. (2005). A new class of wavelet networks for nonlinear system identification, vol. 16, issue. 4, pp. 862 – 874, ISSN: 1045-9227

Borg, J.M., Cheok K.C, Saikalis G. & Oho, S (2005). Wavelet-based knock detection with fuzzy logic, in IEEE International Conference on Computational Intelligence for Measurement Systems and Applications – CIMSA 2005, , pp.26-31, ISBN: 978-1-4244-2306-4, Sicily Italy 14-16 July 2005

Bosch, R. (2004). Bosch-Gasoline-Engine Management, Ed. Robert Bosch GmbH, ISBN-13: 978-0837611006

Boubai, O. (2000). Knock detection in automobile engines, vol.3, issue 3, pp. 24-28, ISSN: 1094-6969

Chen, P.C. (2005). Neuro-fuzzy-based fault detection of the air flow sensor of an idling gasoline engine, vol.219, no. 4, pp.511-524, ISSN 0954-4070

Erjavec, J. (2009). Automotive Technology: A System Approach (5th edition), Ed. Delmar Cengage Learning, ISBN-13: 978-1428311497, Clifton Park NY USA

Ettefagh, M M., Sadeghi, H., Pirouzpanah, V. H. & Arjmandi T. (2008). Knock detection in spark ignition engines by vibration analysis of cylinder block: A parametric modeling approach, vol. 22, Issue 6, pp. 1495-1514, august 2008, ISSN: 0888-3270

Fleming, W.J. (2001). Overview of Automotive Sensors, vol.1, issue 4, pp.296-308, ISSN: 1530-437X

Gupta, H.N. (2006). Fundamentals of Internal Combustion Engines, Ed. Prentice-Hall of India Private Limited, ISBN-13: 978-8120328549, New Delhi India

Hamilton, L J. & Cowart, J S. (2008). The first wide-open throttle engine cycle: transition into knock experiments with fast in-cylinder sampling, vol. 9, no. 2, pp. 97-109, ISSN 1468-0874

Hsu, C.C. (2006). Generalizing self-organizing map for categorical data, vol. 17, issue.2, pp. 294 - 304, ISSN: 1045-9227

Hui, C.L. P. (2011). Artificial Neural Networks - Application, Publisher: InTech, ISBN 978-953-307-188-6, Croatia

Ibrahim, A. M. (2004). Fuzzy logic for embedded systems applications, Ed. Elsevier Science, ISBN-13: 978-0750676052, MA USA

Jonathan, M.B., Saikalis, G., Oho, S.T. & Cheok, K.C. (2006). Knock Signal Analysis Using the Discrete Wavelet Transform, No. 2006-01-0226, DOI: 10.4271/2006-01-0226

Kohonen, T. (2000).Self-organizing Maps (3rd edition), Ed. Springer, ISBN-13: 978-3540679219, Berlin

Kohonen, T. (2002). The self-organizing map, vol. 78, no. 9., pp. 1464-1480, ISSN: 0018-9219

Kwan, H.K. & Cai, T (1994). A fuzzy neural network and its applications pattern recognition, IEEE Transactions on Fuzzy Systems, vol.2, issue.3, pp. 185-193, ISSN: 1063-6706

Larose, D.T. (2006).Data Mining Methods and Models, Wiley-IEEE Press, ISBN-13: 978-0471666561, USA

Lazarescu, D., Radoi, C. & Lazarescu, V. (2004). A Real-Time Knock Detection Algorithm Based on Fast Wavelet Transform, in International Conference Communications 2004, Bucharest, pp. 65-68., ISBN: 0-7803-8533-0, Bucharest 20-24 June 2004

Li, H. & Karim , G. A. (July 2004). Knock in spark ignition hydrogen engines, vol. 29, issue 8, pp. 859-865, ISSN: 0360-3199

Liu, P. & Li, H.X. (2004). Fuzzy neural network theory and application, Publisher: World Scientific Publishing Company, ISBN-13: 978-9812387868, Singapore

Lopez-Rubio E. (2010). Probabilistic Self-Organizing Maps for Continuous Data, vol.21, issue.10, pp. 1543 - 1554, ISSN: 1045-9227

Midori, Y., Nobuo, K. & Atsunori K. (1999). Engine Knock Detection Using Wavelet Transform, Dynamics & Design Conference, Issue B, , pp. 299-302, Tokio, 1999

Mitchell, T.M. (2010). Machine Learning (3rd edition), Ed. New York: McGraw Hill Higher Education, ISBN 0070428077, Oregon USA

Park, S.T. & Jang J. (2004). Engine knock detection based on Wavelet transform, Proceeding of the 8th Russian-Korean International Symposiom on Science and Technology – KORUS 2004, vol.3, pp. 80-83, ISBN: 0-7803-8383-4, 26 June-3 July 2004

Prokhorov, D.(2008). Computational Intelligence in Automotive Applications (1st Edition), Ed.Springer, ISBN 978-3-540-79256-7

Radoi, A., Lazarescu V., & Florescu A. (2009). Wavelet Analysis To Detect The Knock On Internal Combustion Engines, tome 54, no.3, pp. 301-310, ISSN: 0035-4066

Thomas, J.H., Dubuisson, B. & M.A. Dillies-Peltier (1997). Engine Knock Detection from Vibration Signals Using Pattern Recognition, Mecanica, vol.32, no 5, pp. 431-439

Wang, F.Y.& Liu, D. (2006). Advances in Computational Intelligence: Theory And Applications (1st edition), Ed. World Scientific Publishing Company,. ISBN-13: 978-9812567345, Singapore

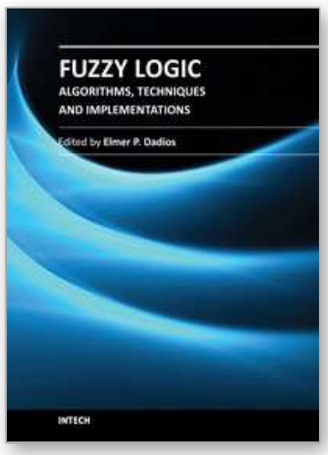
Wehenkel, L.A. (1997) Automatic Learning Technique in Power Systems, Kluwer Academic Publishers, ISBN-13: 978-0792380689 , USA

Wu J.D. & Liu, C.H. (2009). An expert system for fault diagnosis in internal combustion engines using wavelet packet transform and neural network, vol. 36, issue 3, pp. 4278-4286, ISSN: 0957-4174

Yilmaz, S. & Oysal, Y. (2010). Fuzzy Wavelet Neural Network Models for Prediction and Identification of Dynamical Systems, vol. 21 , issue 10, pp. 1599 – 1609, ISSN: 1045-9227

Zhang, Q. & Benveniste A. (1992). Wavelet networks, vol. 3, issue. 6, pp. 889–898, ISSN: 1045-9227

Zhang, H. & Liu, D. (2006). Fuzzy Modeling and Fuzzy Control (Control Engineering) (1st edition) , Ed. Birkhauser Boston, ISBN-13: 978-0817644918, MD USA



Fuzzy Logic - Algorithms, Techniques and Implementations
Edited by Prof. Elmer Dadios

ISBN 978-953-51-0393-6
Hard cover, 294 pages
Publisher InTech
Published online 28, March, 2012
Published in print edition March, 2012

Fuzzy Logic is becoming an essential method of solving problems in all domains. It gives tremendous impact on the design of autonomous intelligent systems. The purpose of this book is to introduce Hybrid Algorithms, Techniques, and Implementations of Fuzzy Logic. The book consists of thirteen chapters highlighting models and principles of fuzzy logic and issues on its techniques and implementations. The intended readers of this book are engineers, researchers, and graduate students interested in fuzzy logic systems.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen