#. A Methodology and a System for Adaptive Speech Recognition in a Noisy Environment Based on Adaptive Noise Cancellation and Evolving Fuzzy Neural Networks

Nikola Kasabov and Georgi Iliev

Department of Information Science, University of Otago, PO Box 56, Dunedin, New Zealand

nkasabov, giliev@infoscience.otago.ac.nz

Abstract

Speech and signal processing technologies need new methods that deal with the problems of noise and adaptation in order for these technologies to become common tools for communication and information processing. This chapter is concerned with a method and a system for adaptive speech recognition in a noisy environment (ASN). A system based on the described method can store words and phrases spoken by the user and subsequently recognise them when they are pronounced as connected words in a noisy environment. The method guarantees system robustness in respect to noise, regardless of its origin and level. New words, pronunciations, and languages can be introduced to the system in an incremental, adaptive mode. The method and system are based on novel techniques recently created by the authors, namely: adaptive noise suppression, and evolving connectionist systems. Potential applications are numerous, e.g. voice dialling in a noisy environment, voice command control, improved wireless communications, data entry into databases, helping disabled people, multimedia systems, improved human computer interaction. The method and system are illustrated on the recognition of English spoken digits in different noisy environments.

1. Introduction

Speech recognition is one of the most challenging applications of signal processing (Owens, 93). Yet there is no noise-robust, adaptive, speaker-independent speech recognition system capable to maintain a medium, or a large vocabulary, available on the world market. The general problem addressed in this chapter is developing adaptive systems working in a noisy environment typical for many applications (e.g. automatic speech recognition (ASR) in offices, vehicles, aeroplanes etc).

The chapter offers solutions to the following specific research problems:

- The noise suppression problem, that is to design novel methods and systems for effective noise suppression in different environments. Examples are vehicle noise (cars, trucks etc.), aircraft noise, industrial noise, office noise.
- The adaptive speech recognition problem, that is the development of methods and systems for speaker-independent recognition with high accuracy, capable to adapt fast to new words, new accents, new speakers for a small-, medium-, to large vocabulary of words, phrases and sentences.
- The problem of integrated speech systems design capable to work reliably in severe noise conditions.

The noise cancellation problem is comparatively well explored (Pelton, H., 93). There are methods available that are well situated for different types of noise (Widrow, Stearns, 85). The research on the adaptive speech recognition problem is still in its infancy. It has been solved only for speaker dependent systems where the user adjusts the system to their voice (Pelton, G., 93).

Here the goal is to develop a method and a system that combine the advantages of several approaches in order to achieve an adaptive system efficient in a range of noisy environments.

The method and system developed and investigated in the chapter are based on novel techniques that have been recently created by the authors, namely: adaptive noise suppression (Iliev, Kasabov, 99); evolving connectionist systems for adaptive learning (Kasabov, 98, 99).

The chapter presents first the framework of a noise-robust, adaptive speech recognition system. Then the methods for adaptive noise suppression and adaptive learning and recognition are explained. The chapter also gives some experimental results achieved on a case study problem - the recogni-

tion of English spoken digits in different noisy environments. A comparative analysis is also presented when the suggested method and system are compared with other systems on the case study problem. Conclusions and suggestions of how the method and the system can be used for practical applications are given at the end.

2. The Framework of the Methodology and the System for Adaptive Speech Recognition in a Noisy Environment

The framework of the proposed ASN methodology and a system is schematically shown in Fig.1. It consists of the following modules and procedures: adaptive noise suppression (ANS), endpoint detection (EPD), acoustic feature extraction (AFE), feature normalisation with the use of the Discrete Cosine Transform (DCT) and speech recognition module that uses evolving fuzzy neural networks (EFuNNs). Other modules not shown in the figure are a temporal buffer for storing a sequence of recognised words and a phrase and sentence recognition module that follow the EFuNN module.

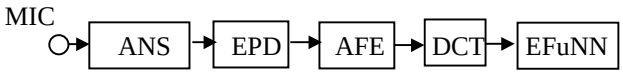


Fig. 1. Block diagram of the framework of the proposed ANS.

The methodology we propose here is based on the following principles:

- Speech recording and noise cancellation with the use of an original method suggested by the authors. The noise is effectively removed from the signal and a 'noise-free' signal is achieved. The method tolerates very low signal-to-noise ratio (close to 0).
- Transforming the 'noise-free' signal with the use of standard transformations tailored for the purpose. That includes: end point detection to determine the boundaries of each word; acoustic feature extraction to extract representative features from the signal; feature normalisation to produce a fixed length vector that represents each spoken word and preserves the difference between the words. Here specific to the method, but controllable features and parameters are used.

- The vector that represents the pronounced word or phrase than is fed into a word-storage module created with the use of the general purpose adaptive learning method realised as an EFuNN. The EFuNN allows for adaptive learning. New words and phrases can be added or deleted from the system at any time of its operation, e.g. “go”, “one”, “connect to the Internet”, “start”, “end”, “find a parking place”. New speakers can be introduced to the system, new accents, new languages. In the recognition mode, when speech is entered to the system, the recognised words and phrases at consecutive time moments are stored in the temporal buffer.
- The temporal buffer is fed into an EFuNN-sentence module where multiple word sequences (or a whole sentence) are recognised.
- The recognised word or a sequence of words can be passed to an action module for execution depending on the application of the proposed method and system.

The signal processing part at the beginning is organised as follows: Short-time energy and zero-crossing rate are combined to detect the speech utterance boundaries. Acoustic features of the input speech are extracted over 20 ms frames. Hamming windows having an overlap of 10 ms are used to calculate Mel Frequency Scale Cepstral Coefficients (MFSCC) and log-energy. A discrete cosine transform (DCT) is applied to the whole segment, retaining as many parameters as it is necessary.

3. Adaptive Noise Suppression

3.1. Adaptive Filtering and Adaptive Noise Cancellation

Fig. 2 shows the classical scheme for adaptive noise cancellation using digital filter with finite impulse response (FIR). The primary input consists of speech s(n) and noise $n_2(n)$ while the reference input consists of noise $n_1(n)$ alone. The two noises $n_1(n)$ and $n_2(n)$ are correlated and $h_i(n)$ is the impulse response of the noise path. The system tries to reduce the impact of the noise in the primary input exploring the correlation between the two noise signals. This is equivalent to the minimisation of the mean-square error $E[e^2(n)]$ where

$$e(n) = s(n) + n_2(n) - n_3(n) \quad (1)$$

Having in mind that by assumption, s(n) is correlated neither with $n_2(n)$ nor with $n_1(n)$ we have

$$E[e^2(n)] = E[s^2(n)] + E[n_2(n) - n_3(n)]^2. \quad (2)$$

In other words the minimisation of $E[e^2(n)]$ is equivalent to the minimisation of the difference between $n_2(n)$ and $n_3(n)$. Obviously $E[e^2(n)]$ will be minimal when $n_3(n) \approx n_2(n)$ i.e. when the impulse response of the adaptive filter closely mimics the impulse response of the noise path.

The minimisation of $E[e^2(n)]$ can be achieved by updating the filter taps $w_i(n)$. Most often NLMS and RLS algorithms are used.

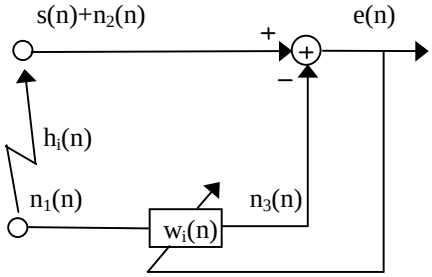


Fig. 2. Adaptive noise cancellation.

3.2. Adaptive Blind Noise Suppression

The basic idea is that in many applications, for instance, hands-free cellular phones in car environment (Kuo, Chuang, Mallela, 93), howling control in hands-free phones, noise reduction in an office environment, the noise reveals specific features that can be exploited. In most instances although the noise might be quite wide-band, there are always, as a rule, no more than two or three regions of its frequency spectrum that carry most of the noise energy (White, Walker, 82) and the removal of these dominant frequencies results in a considerable improvement of signal-to-noise ratio (SNR). This brings the idea to use notch adaptive filters capable of tracking the noise characteristics. They have to meet the following requirements:

- to adapt as fast as possible to the changes in the noise which might be quite rapid, for example car engine noise;

- the cancelled portions of the spectrum should be as narrow as possible in order to prevent speech signal distortions.

Both requirements could be met much easier using infinite impulse response (IIR) adaptive filters instead of finite impulse response (FIR) adaptive filters. IIR filters are usually avoided because they create a lot of stability problems. To overcome this problem we use a realisation based on second order Gray-Markel lattice circuit (Regalia, Mitra, Vaidyanathan, 88) - Fig.3. Using this circuit it becomes possible to implement a second order notch/bandpass section (Regalia, 91) - Fig. 4.

What are the advantages of such a realisation? First, it has extremely low pass-band sensitivity that means resistance to quantisation effects. Second, it is very convenient for realisation of adaptive notch filters because it is possible to control independently the notch frequency and the bandwidth.

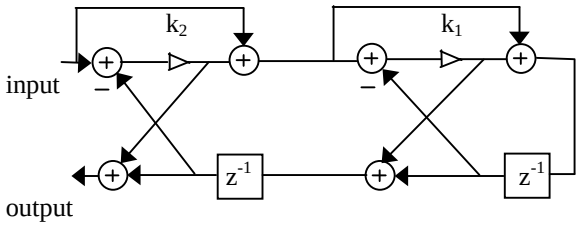


Fig.3. Second order lattice Gray-Markel circuit realising all-pass function A(z).

Thus if the all-pass function A(z) is

$$A(z) = \frac{k_2 + k_1(1+k_2)z^{-1} + z^{-2}}{1 + k_1(1+k_2)z^{-1} + k_2 z^{-2}} \tag{3}$$

then k_1 controls the notch frequency ω_0 while k_2 is related to the bandwidth (BW) via

$$k_1 = -\cos \omega_0 \tag{4}$$

$$k_2 = \frac{1 - \tan(BW/2)}{1 + \tan(BW/2)}. \tag{5}$$

But, on the other hand, BW is directly connected to the distance from the pole to the unity-circle. So if we use the structure of Fig. 3 as an adaptive filter we may fix BW and thus fixing k_2 we make the distance from the pole

to the unity-circle constant which means that with this constraint we obtain an adaptive IIR filter free of stability problems. Through adapting k_1 we shift the notch frequency around the unity-circle.

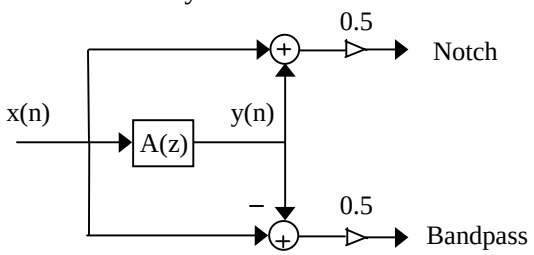


Fig. 4. Second order notch/bandpass section.

Using the basic structure of Fig. 4 and the constraint mentioned above, the final arrangement of our system is shown in Fig. 5. The system will work in the following manner: each section will remove one of the dominant frequencies using an appropriate adaptive algorithm. As shown in Fig. 4 we propose to update only the coefficients k_{11}, k_{12},…, k_{1M}, while k_2 is a priori determined from equation (5). Thus we can reduce considerably the number of computations and can guarantee the stability of the adaptive structure. The number of sections is determined by the application. Here we introduce the normalised least mean square (NLMS) algorithm for adjusting the filter coefficients as follows:

$$e_i(n) = 0.5[e_{i-1}(n) + y_i(n)] \tag{6}$$

for i = 1 to M and $e_0(n) = x(n)$

$$k_{1i}(n+1) = k_{1i}(n) - \mu \frac{e_i(n)\, y'_i(n)}{[y'_i(n)]^2} \tag{7}$$

$$y'_i(n) = \frac{d\, y_i(n)}{d\, k_{1i}(n)}$$

for $1 \le i \le M$

where M is the number of sections, $e_i(n)$ is the error signal, μ is the step size and $y'_i(n)$ is the derivative of $y_i(n)$ with respect to the coefficient subject of adaptation.

We also call this method adaptive blind noise suppression (ABNS). The method is illustrated by computer simulations. Fig. 6 shows the original

speech. The speech is corrupted with noise from computer cooling fan that is the most often encountered noise in an office environment and the resultant signal is depicted in Fig. 7 (a). The process of noise suppression is shown in Fig .7 (b). Here the system is composed of 3 sections each of them adapting its coefficient to one of the dominant frequencies in the noise spectrum.

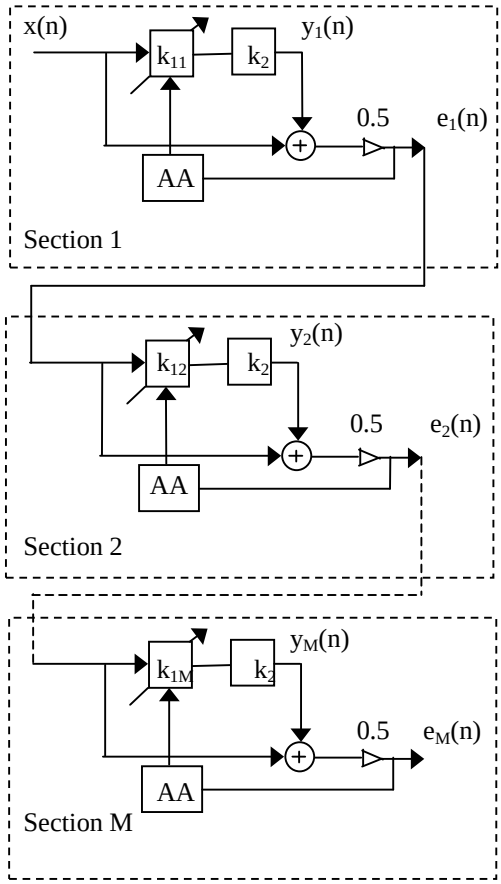


Fig. 5. Adaptive system for noise suppression.

The main advantages of this system for noise suppression are:

- the adaptive system has a short time of adaptation - about 100 iterations;

- The system is very simple and flexible. For comparison, here we adjust only 3 coefficients against 250-450 in the conventional adaptive noise cancellation systems (Widrow, Stearns, 85).
- the second-order lattice structures are stable during the adaptation that defines the high stability of the whole system.

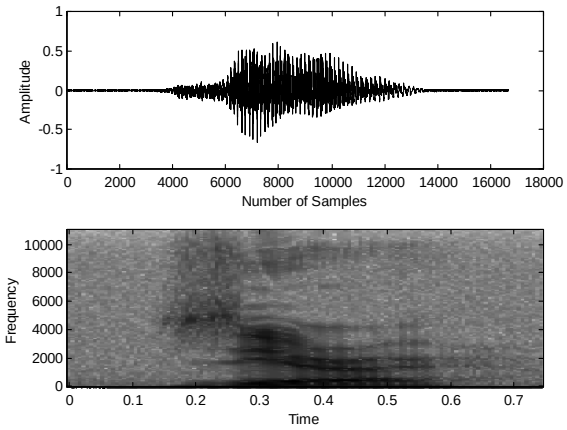


Fig. 6. Original speech - the word “zero”.

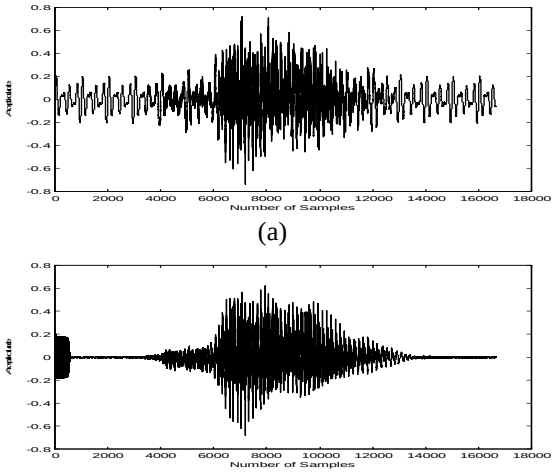


Fig.7. (a) Speech and noise. (b) Speech after noise suppression.

3.3. A Novel Algorithm for Adaptive Noise Suppression Based on Adaptive Filtering with Averaging

For application where the fast convergence rate is vital, NLMS algorithm is not very efficient. The more complex recursive least square (RLS) algorithm maintains a good rate of adaptation but the prize to be paid is an order-of-magnitude increase in complexity. Moreover, the RLS algorithm is known for having stability problems (Proakis, 95) due to the recursive covariance update formula. In this section we introduce a new adaptive algorithm applied for noise cancellation based on adaptive filtering with averaging.

We start with the definition of the problem in the following manner. To recursively adjust the filter coefficients, so that the mean-square error is minimised, a standard algorithm for approximating the vector of filter coefficients can be written as follows:

$$W(n+1) = W(n) - a(n)N_1(n)e(n), \quad (8)$$

where

$W(n) = [w_0(n), w_1(n),\ldots, w_N(n)]^T$ is the coefficients vector,

$N_1(n) = [n_1(n), n_1(n-1),\ldots, n_1(n-N)]^T$ is the input vector, and $a(n)$ is a sequence of positive scalars as $a(n)\to 0$ for $n\to\infty$.

In (8) the estimation error can be given by

$$e(n) = s(n) + n_2(n) - N_1^T(n)W(n). \quad (9)$$

The equation (8) could be transformed through taking the averages of W:

$$W(n+1) = W(n) + \frac{1}{n^{\gamma}} N_1(n)e(n) \quad (10)$$

$$\overline{W}(n) = \frac{1}{n}\sum_{k=1}^{n} W(k)$$

$1/2<\gamma<1$

The analysis presented in (Astrom, Goodwin, Kumar, 95) shows that algorithm based on the principle from above may be unstable in the initial period. In order to improve the stability, we undergo the second step, namely averaging not only the approximation sequence, but also the observed sig-

nals N_1 and e. This leads us to an adaptive filtering algorithm with averaging (AFA):

$$\overline{W}(n) = \frac{1}{n}\sum_{k=1}^{n} W(k)$$

$$W(n+1) = \overline{W}(n) + \frac{1}{n^{\gamma}}\sum_{k=1}^{n} N_1(k)e(k) \qquad (11)$$

$1/2<\gamma<1$

The required steps for the utilisation of the AFA algorithm for noise cancellation are presented in Table 1.

Considering Table 1 it can be concluded that first, the averaging here does not create additional burden since the terms $\overline{w}_i(n)$ and $\overline{n_1 e_i}(n)$ can be recursively computed from their past values. Second, the algorithm does not use the covariance matrix, so there is no need of covariance estimate. This implies low computational complexity and escape from stability issues related to P(n).

TABLE 1
The AFA algorithm applied to noise cancellation.

Noise estimation:

$$n_3(n) = \sum_{i=0}^{N} w_i(n)\, n_1(n-i)$$

N – filter order

Error estimation:

$$e(n) = s(n) + n_2(n) - n_3(n)$$

Coefficients update:

$$\overline{w}_i(n) = \frac{1}{n}\sum_{k=1}^{n} w_i(k)$$

$$\overline{n_1 e_i}(n) = \sum_{k=1}^{n} n_1(k-i)e(k)$$

$$w_i(n+1) = \overline{w}_i(n) + \frac{1}{n^{\gamma}}\overline{n_1 e_i}(n)$$

for $0 \le i \le N$ and $1/2<\gamma<1$

The NLMS, RLS and AFA algorithms are implemented and experimented with for the following parameters: for the NLMS algorithm - μ= 0.02; for the RLS algorithm - δ= 0.98; and for the AFA algorithm - γ =

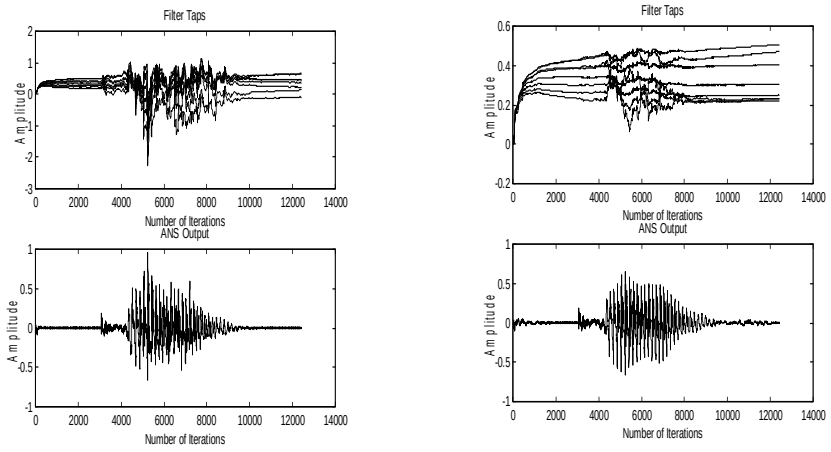
0.5.The averaging is started after a suitable transient period, here chosen to be 40 iterations.

First, the original speech (the word "two") is corrupted with office noise (SNR=8dB) and the results after noise cancellation are shown in Fig. 8 and 9. Second, an experiment with car noise (SNR=3dB) is conducted and the results for the different algorithms are presented in Fig. 10 and 11 (here the original speech is the word "seven").

Comparing the results of the different algorithms it is clear that RLS and AFA outperform NLMS algorithm. The latter manifests a high deviation in its coefficients that results in a poorer performance.

The main goal of this section is to investigate the application of an algorithm based on adaptive filtering with averaging for the noise cancellation problem. Here the main concern is to achieve a high convergence rate in order to meet the requirements imposed by different applications where rapid changes in the signal characteristics could occur. In this aspect the obtained results show that the AFA algorithm is very promising. Its main advantages could be summarised as follows:

- high adaptation rate, comparable to that of the RLS algorithm;
- low computational complexity and robustness when fixed-point implementation is used.



(a) (b)

Fig. 8. Office noise, SNR=8dB. Noise suppression with the use of: (a) The NLMS algorithm. (b) The RLS algorithm.

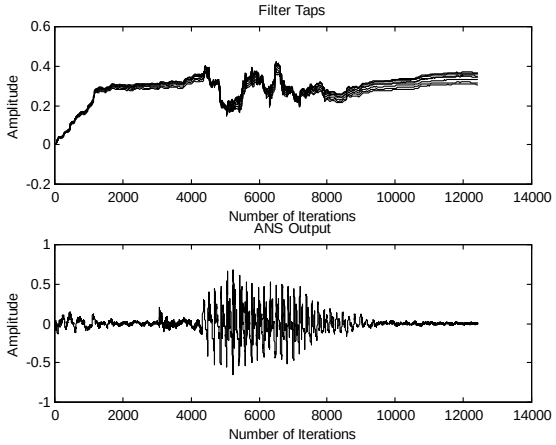
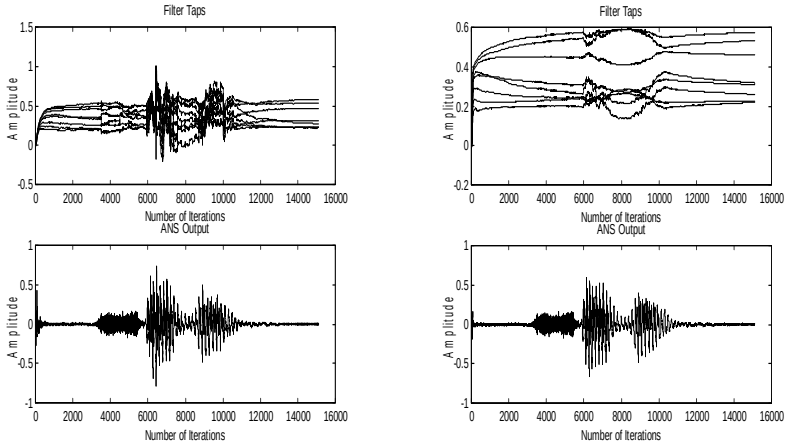


Fig. 9. Noise suppression with the use of the AFA algorithm; office noise, SNR=8dB.



(a) (b)

Fig. 10. Car noise, SNR=3dB. Noise suppression with the use of: (a) The NLMS algorithm. (b) The RLS algorithm.

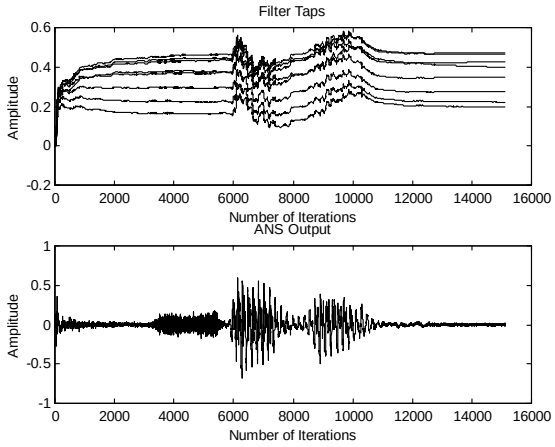


Fig. 11. Noise suppression with the use of the AFA algorithm; car noise, SNR=3dB.

4. Evolving Connectionist Systems and Evolving Fuzzy Neural Networks for Adaptive Learning and Classification

4.1. Evolving Connectionist Systems (ECOS)

The proposed methodology and a system for adaptive speech recognition in a noisy environment uses a method for adaptive learning called evolving connectionist systems (ECOS) (Kasabov, 98,99). ECOS learn in a continuous, incremental way, adapt to changes in data, are self-organised, are controllable (i.e., through pre-setting appropriate values for certain parameters, certain level of learning and generalisation can be achieved).

The ECOS method allows a system to be automatically created. The system consists of nodes (units) that perform pre-defined functions, and connections between them. The system has a minimal initial structure that includes preliminary input and output nodes and few preliminary connections. Data is allowed to flow into the system so that if an input data vector is associated with a desired output vector, the system stores this association into a new node and new connections. Nodes and connections are created automatically to reflect the data distribution. The system's structure is dynami-

cally changing after each data item is introduced. The number of the input and output variables can vary during the learning process thus allowing for more (or less) input and output variables to be introduced at any stage of the learning process. Input and output variables can have 'missing values' at any time of the learning process. If there is no output vector associated with an input vector, the system produces its own output vector (its own solution). If the desired output vector became known afterwards, the system will adjust its structure to produce this output, or one close to it next time the same input vector is presented. The system continuously and adaptively learns from data to associate inputs to outputs and to cluster the data trough allocating nodes to represent exemplars of data.

The learning process in ECOS is achieved through interaction with the environment, which supplies the data flow and reacts to the output produced by the system (see Fig.12). In addition, the system provides the knowledge it has learned in the form of IF-THEN rules. The ECOS method implies that a system evolves through its operation in an interactive way (see Fig. 12). The more data are presented to the system, the more the system evolves. The learning process is on-line, life-long.

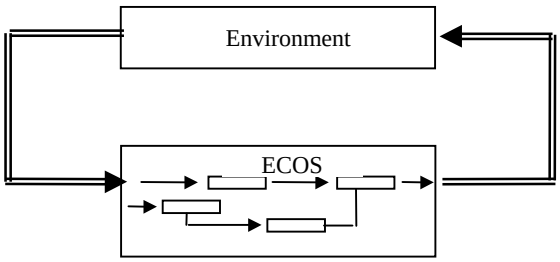


Fig. 12. Learning in ECOS is through interaction with the environment (from (Kasabov, 98, 99)).

Nodes and connections in an ECOS system can be created, modified, merged, and pruned, in a self-organising manner, similar to how the human brain learns through creating and wiring neuronal structures. The system's structure grows or shrinks depending on the incoming data distribution and pre-set parameters. Through the process of evolving from data, the system learns the rules of its own behaviour. The rules that constitute the system's knowledge can be reported and/or extracted at any time of the system opera-

tion (Kasabov, 99). In this way, an ECOS can be considered as a self-programming environment.

One realisation of ECOS is the evolving fuzzy neural network (EFuNN) (Kasabov, 98,99). An example of EFuNN is given in Fig. 13. EFuNNs have a five-layer structure. Nodes and connections are created/connected as data examples are presented. An optional short-term memory layer can be used through a feedback connection from the rule (also called, case) node layer (see for example (Kasabov, 99)). The layer of feedback connections could be used if temporal relationships of input data are to be memorised structurally. The input layer represents input variables. The second layer of nodes (fuzzy input neurons, or fuzzy inputs) represents fuzzy quantisation of each input variable space. For example, two fuzzy input neurons can be used to represent "small" and "large" fuzzy values. Different membership functions (MF) can be attached to these neurons (triangular, Gaussian, etc.). The number and the type of MF can be dynamically modified in an EFuNN. New neurons can evolve in this layer if, for a given input vector, the corresponding variable value does not belong to any of the existing MF to a degree greater than a membership threshold. A new fuzzy input neuron, or an input neuron, can be created during the adaptation phase of an EFuNN. The task of the fuzzy input nodes is to transfer the input values into membership degrees to which they belong to the MFs. The third layer contains rule (case) nodes that evolve through supervised/unsupervised learning. The rule nodes represent prototypes (exemplars, clusters) of input-output data associations, graphically represented as an association of hyper-spheres from the fuzzy input and fuzzy output spaces. Each rule node r is defined by its own values for the system parameters, and by two vectors of connection weights – W1(r) and W2(r), the latter being adjusted through supervised learning based on the output error, and the former being adjusted through unsupervised learning based on similarity measure within a local area of the problem space. The fourth layer of neurons represents fuzzy quantisation for the output variables, similar to the input fuzzy neuron representation. The fifth layer represents the real values for the output variables.

An initial version of the EFuNN evolving algorithm is presented in (Kasabov, 98, 99). The learning process includes also: aggregation of rule nodes (i.e. merging rule nodes); pruning of rule nodes and connections and other operations. Here an optimised evolving algorithm is presented.

4.2. A Self-Optimised Learning Algorithm for Evolving Fuzzy Neural Networks

The basic EFuNN algorithm, to evolve EFuNNs from incoming examples, is further developed and improved as a method for adaptive learning and self-optimisation (Kasabov, 2000) as presented below:

The EFuNN learning algorithm

The algorithm is given below as a procedure of consecutive steps. Matrix operation expressions are used similar to the expressions in a matrix processing language such as MATLAB.

1. Initialise an EFuNN structure with a maximum number of neurons and no (or zero-value) connections. Initial connections may be set through inserting fuzzy rules in the structure. If initially there are no rule (case) nodes connected to the fuzzy input and fuzzy output neurons, then create the first node rn=1 to represent the first example d1 and set its input $W1(r_n)$ and output $W2(r_n)$ connection weight vectors as follows:

<Create a new rule node r_n >: $W1(r_n)$=EX; $W2(r_n)$ = TE, where TE is the fuzzy output vector for the current fuzzy input vector EX.

2. WHILE <there are examples in the input stream> DO

Enter the current example (Xd_i,Yd_i), EX denoting its fuzzy input vector. If new variables appear in this example, which are absent in the previous examples, create new input and/or output nodes with their corresponding membership functions.

3. Find the *local normalised fuzzy distance* between the fuzzy input vector EX and the already stored patterns (prototypes, exemplars) in the rule (case) nodes rj=r1,r2,…,rn

$D(EX, r_j)$= sum (abs (EX - $W1(r_j)$)) / sum ($W1(r_j)$+EX)

4. Find the activation A1 (r_j) of the rule (case) nodes r_j, r_j , j=1: r_n. Here radial basis activation function, or a saturated linear one, can be used, i.e. A1 (r_j) = radbas (D(EX, $W1(r_j)$)), or A1(r_j) = satlin (1 – D(EX, $W1(r_j)$)). The former may be appropriate for function approximation tasks, while the latter may be preferred for classification tasks. In case of the feedback variant of an EFuNN, the activation A1(r_j) is calculated as:

A1 (r_j) = radbas (Ss. D(EX, $W1(r_j)$) - Tc.W3), or A1(r_j) = satlin (1 – Ss. D(EX, $W1(r_j)$) + Tc.W3).

5. Update the pruning and aggregation parameter values for the rule nodes, e.g. *age*, *average activation*, *sensitivity threshold*, etc.

6. Find all case nodes rj with an activation value A1(rj) above a sensitivity threshold Sthr.

7. IF there is no such case node, then <Create a new rule node> using the procedure from step 1 in an unsupervised learning mode ELSE

8. Find the rule node $inda_1$ that has the maximum activation value (e.g., $maxa_1$).

9. (a) in case of "one-of-n" EFuNNs propagate the activation of the rule node $inda_1$ to the fuzzy output neurons: $A2 = satlin (A1(inda_1) . W2(inda_1)$

(b) in case of "many-of-n" mode, the activation values of all rule nodes that are above an activation threshold of Athr are propagated to the next neuronal layer (this case is not discussed in details here; it has been further developed into a new EFuNN architecture called dynamic, 'many-of-n' EFuNN, or DEFuNN) .

10. Find the winning fuzzy output neuron $inda_2$ and its activation $maxa_2$.

11. Find the desired winning fuzzy output neuron $indt_2$ and its value $maxt_2$.

12. Calculate the fuzzy output error vector: Err=A2 - TE.

13. IF ($inda_2$ is different from $indt_2$) or (D(A2,TE) > Errthr) <Create a new rule node>

ELSE

14. Update: (a) the input, (b) the output, and (c) the temporal connection vectors (if such exist) of the rule node $k=inda_1$ as follows:

- $Ds(EX,W1(k))=EX-W1(k)$; $W1(k)=W1(k) + lr_1.Ds(EX,W1(k))$, where lr_1 is the learning rate for the first layer;
- $W2(k) = W2 (k) + lr_2. Err. maxa_1$, where lr_2 is the learning rate for the second layer;
- $W3(l,k)=W3(l,k)+lr_3. A1(k).A1(l)^{(t-1)}$, here l is the winning rule neron at the previous time moment (t-1), and $A1(l)^{(t-1)}$ is its activation value kept in the short term memory.

15. Prune rule nodes j and their connections that satisfy the following fuzzy pruning rule to a pre-defined level:

IF (a rule node rj is OLD) AND (average activation A1av(rj) is LOW) and (the density of the neighbouring area of neurons is HIGH or MODERATE (i.e. there are other prototypical nodes that overlap with j in the input-output space; this condition apply only for some strategies of inserting rule nodes as explained in a sub-section below)

THEN the probability of pruning node (rj) is HIGH

The above pruning rule is fuzzy and it requires that the fuzzy concepts of OLD, HIGH, etc., are defined in advance (as part of the EFuNN's chromosome). As a partial case, a fixed value can be used, e.g. a node is OLD if it

has existed during the evolving of a FuNN from more than 1000 examples. The use of a pruning strategy and the way the values for the pruning parameters are defined, depends on the application task.

16. Aggregate rule nodes (after every pre-defined number of examples) into a smaller number of nodes.

17. END of the while loop

18. END of the algorithm

Self-optimisation in EFuNN structures:

In case of self-optimisation there is no need to specify (fix) any of the parameters of the EFuNN. The structure defines its parameters based on the incoming data. Each rule node r_j has its own parameters – sensitivity threshold Sj, receptive field radius Rj=1-Sj; error threshold Ej; learning rates $lr_{1(j)}$ and $lr_{2(j)}$. Each of these parameters is automatically defined during learning and aggregation. Aggregation is based on the distance measure D1(W1(r_j), W1(r_i)) between rule nodes r_j and r_i in the fuzzy input space and the distance measure D2(W2(r_j), W2(r_i)) in the fuzzy output space. Two thresholds T1 and T2 define if rule nodes will be aggregated – the two distances D1 and D2 should be both less than the corresponding thresholds T1 and T2.The following are the main points of the method:

- There are initial values for the parameters assigned to each neuron: S=1; R=0; E=0; lr_1=lr_2=0; T1=0.5 (this threshold defines when rule nodes in the same fuzzy subspace of the input space can be aggregated); T2. The threshold T2 could have different values depending on the class of problems, e.g. T2 =0.05 for function approximation and prediction, which gives sufficient tolerance for generalisation, and T2=0.2 for classification.
- Rule nodes are linearly and spatially ordered and perform one-dimensional vector quantisation of the input space. A new rule node is inserted next to the highest activated one, either on the left or on the right side depending on where the second highest activated neuron is located.
- After a certain number of epochs (depending on the task) aggregation is performed as follows:

(a) Let us assume that r_{j-1} and r_j are the first two rule nodes (considering the number of the nodes in an ascending order) such that D1 (W1(r_j), W1(r_{j-1})) <T1 and D2(W2(r_j), W2(r_{j-1})) < T2. The two distances D1 and D2 between the rule node r_j and each of its consecutive neighbouring rule nodes r_{j+1},r_{j+2},…,$r_{j+m,\ldots,}$ etc. are measured and both are less than the corresponding

thresholds T1 and T2, until a rule node r_{j+m+1} is reached for which either $D1(W1(r_{j),}W1_{(}r_{j+m+1)})>T1$, or $D2(W2(r_{j),}W2_{(}r_{j+m+1)})>T2$. The rule nodes r_{j+m} and r_{j+m+1} represent the border between one class (cluster) of data and another – they are the "guards".

(b) The rule node r_j is then aggregated with all rule nodes $r_{j+1},r_{j+2},\ldots,r_{j+m-1}$ such that the new rule node r_j^{agg} is placed at the geometrical centre of all the aggregated nodes:

$W1(r_j^{agg}) = \Sigma_{i= j, j+1, j+2,\ldots, j+m-1} (W1(r_i)) / m$

$W2(r_j^{agg}) = \Sigma_{i= j, j+1, j+2,\ldots, j+m-1} (W2(r_i)) / m$

(c) The new receptive field $R(r_j^{agg})$ and the sensitivity threshold $S(r_j^{agg})$ are defined as follows:

$R(r_j^{agg}) = \max \{D1(W1(r_j^{agg}), W1(r_i))\}$, for i= j, j+1,j+2,…,j+m-1

$S(r_j^{agg}) = 1 - R(r_j^{agg})$

(d) The new error threshold of the aggregated rule node is defined as:

$E(r_j^{agg}) = \max \{D2(W2(r_j^{agg}), W2(r_i))\}$, for i= j, j+1,j+2,…,j+m-1

For all rule nodes r_k that are not aggregated during the current aggregation procedure the sensitivity threshold and the error threshold decay slightly:

$S(r_{k)} = S(r_{k)} - \alpha S(r_{k)}$, where α is a small decay constant, e.g. α=0.001.

$E(r_{k)} = E(r_{k)} + \beta E(r_{k)}$, where β is a small decay constant, e.g. β=0.001.

When a new input-output fuzzy pair of vectors x = (EX,TE) is accommodated into a rule node r_j and its position is changed from $W1(r_j)$ and $W2(r_j)$ to $W1(r_j^{upd})$ and $W2(r_j^{upd})$ in the input and the output fuzzy spaces as explained in the main EFuNN algorithm, the parameters of this node change as follows:

$R(r_j^{upd}) = R(r_j) + D1(W1(r_j^{upd}), W1(r_j))$

$S(r_j^{upd}) = 1 - R(r_j^{upd})$

$E(r_j^{upd}) = E(r_j) + D2(W2(r_j^{upd}), W2(r_j))$

$lr1(r_j^{upd}) = lr1(r_j) + c1.\ D1(W1(r_j^{upd}), W1(r_j))$

$lr2(r_j^{upd}) = lr2(r_j) + c2.\ D2(W2(r_j^{upd}), W2(r_j))$

where: c1 and c2 are small coefficients, for example 0.001.

In the aggregation procedure two thresholds are used T1 and T2, and every two rule nodes for which the distance in the input space is less than T1 and the distance in the output space is less than T2 get aggregated. A normalised distance as a measure for the similarity between two vectors (two rule nodes represented by their connection weight vectors W1 and W2) is used in such a manner that providing the two vectors have been normalised and all their values are inside the interval [0,1], the distance is also inside

this interval. Using Eucledian distance does not maintain this requirement and we have to change the two thresholds used T1 and T2 for different dimensions (N) of vectors. So we define this normalised distance as follows. Here the way the distance is measured is introduced:

Definition: If X and Y are two vectors with dimension N then the normalised distance between them is given by

$$D_{XY}^{norm} = \frac{\sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}}{\sqrt{N}}. \qquad (12)$$

Property: Consider the distance between two vectors X and Y given by (12) where

$D_{XY} = \sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}$ is the Eucledian distance, then if $0 \le x_i \le 1$ and $0 \le y_i \le 1$ for i=1:N the following condition holds

$$0 \le D_{XY}^{norm} \le 1$$

Proof:
If $0 \le x_i \le 1$ and $0 \le y_i \le 1$ for i=1:N then

$$D_{XY\,max}^{norm} = \frac{D_{XY\,max}}{\sqrt{N}} = \frac{\sqrt{N}}{\sqrt{N}} = 1$$

and

$$D_{XY\,min}^{norm} = \frac{D_{XY\,min}}{\sqrt{N}} = \frac{0}{\sqrt{N}} = 0$$

where D_{XYmax} is the maximum Eucledian distance and D_{XYmin} is the minimum Eucledian distance. Under the assumptions for X and Y $D_{Xymax} = \sqrt{N}$ and $D_{XYmin}=0$.

An algorithm for rule extraction from EFuNNs is presented in (Kasabov, Woodford, 99).

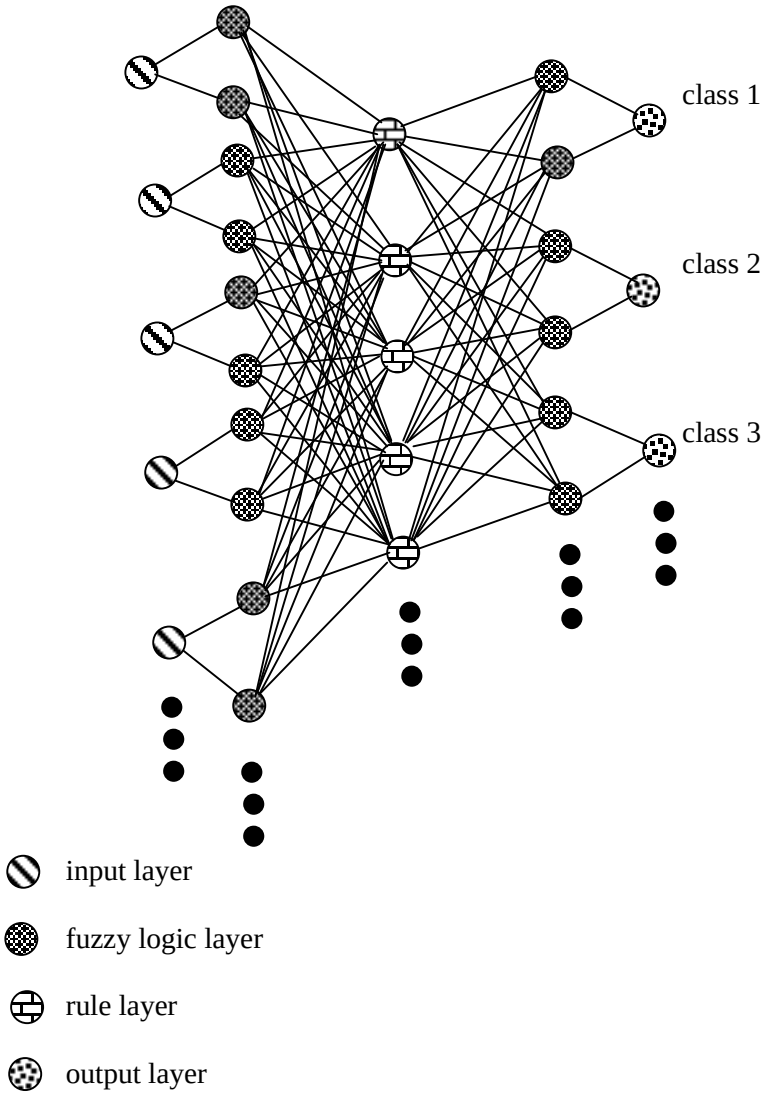


Fig. 13. An exemplar EFuNN structure for classification [5-10-5-6-3].

The next section describes a case study on the application of the ASN from Fig.1 built to recognise English spoken words with different sources of noise applied.

5. Experimental Results

The task is recognition of speaker independent pronunciations of English digits from the Otago Corpus database (http://kel.otago.ac.nz/hyspeech/ corpus/). 17 speakers (12 males and 5 females) are used for training and other 17 speakers (12 males and 5 females) are used for testing. Each speaker utters 30 instances of English digits during recording session in a quiet room (clean data) for a total of 510 training and 510 testing utterances. We use 8 mel frequency scale cepstrum coefficients (MFSCC) and log-energy as acoustic features.

In order to assess the performance of EFuNN in this application, a comparison with Linear Vector Quantisation (LVQ) is accomplish. The clean training speech is used to train both LVQ and EFuNN. Noise is introduced in the clean speech to evaluate behaviour of the recognition systems in a noisy environment. Two different experiments are conducted with the use of the standard EFuNN learning method (Kasabov, 98, 99). In the first instance, car noise is added to the clean speech. In the second instance office noise is introduced over the clean signal. In both cases the SNR ranges from 0 dB to 18 dB.

The results for car noise are shown in Fig. 14. The word recognition rate (WRR) ranges from 86.87% at 18 dB to 83.33% at 0 dB in EFuNN case outperforming LVQ, which achieves WRR=82.16% at 0 dB.

The results for office noise are presented in Fig. 15. The WRR ranges from 78.63% at 18dB to 71.37% at 0 dB in EFuNN case and is significantly higher than the WRR of LVQ (21.18% at 0 dB).

Another experiment is conducted with adaptive noise suppression as a pre-processing technique. The results are summarised in Table 2 and they show that our system combining EFuNN and adaptive noise suppression is able to achieve WRR of 91.74% in the case of car noise and 86.63% in the case of office noise with SNR=0 dB. This is well above the results of the LVQ algorithm, which is a recognition system most often used as a benchmark in designing new recognition systems. This experiment illustrates how an ASN system works and what its advantages are. The ASN is based on both the AFA noise suppression method and on the EFuNN training method. EFuNN outperforms significantly LVQ both in WRR and especially in the required time for training, which is 3 to 4 orders of magnitude less. It uses one pass of data propagation, while LVQ (as well as all known connectionist methods for learning from data) requires many iterations.

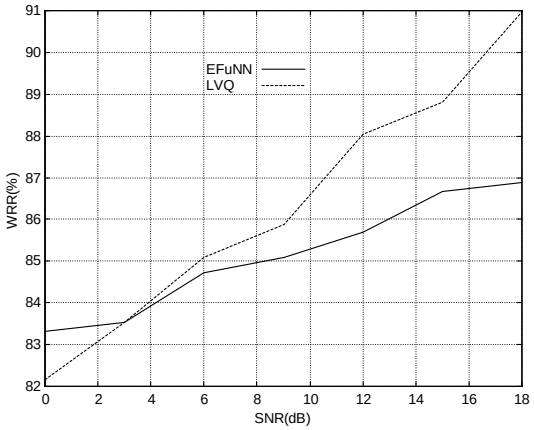


Fig. 14. Word recognition rate (WRR) of two speech recognition systems when car noise is added: LVQ – codebook vectors – 396, training iterations – 15840. EFuNN – 3MF, rule nodes – 157, sthr=0.9, errthr=0.1, lr1=0.01, lr2=0.01, thrw1=0.2, thrw2=0.2, nexa=100, 1 training iteration.

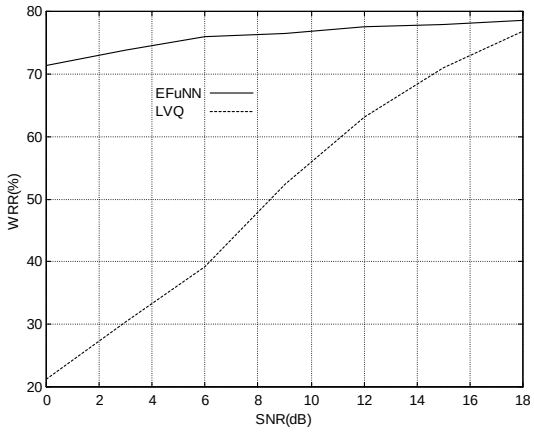


Fig. 15. Word recognition rate (WRR) of two speech recognition systems when office noise is added: LVQ – codebook vectors – 396, training iterations – 15840. EFuNN – 3MF, rule nodes – 157, sthr=0.9, errthr=0.1, lr1=0.01, lr2=0.01, thrw1=0.2, thrw2=0.2, nexa=100, 1 training iteration.

TABLE 2
The performance of an ASN system based on EFuNN and Adaptive Noise Suppression - 3MF, rule nodes – 157, sthr=0.9, errthr=0.1, lr1=0.01, lr2=0.01, thrw1=0.2, thrw2=0.2, nexa=100, 1 training iteration.

Type of noise	SNR (dB)	WRR (%)
car noise	0	91.74
	9	91.87
office noise	0	86.47
	9	86.63

Preliminary results with an ASN system that uses the algorithm from (Kasabov, 2000) show improvement across the noise level scope.

6. Conclusions

The ASN method and system proposed here are characterised by the following characteristics:

- effective adaptive noise cancellation that is regardless of the origin of the noise; the noise suppression module adapts continuously to the noise added to the speech signal;
- a theoretical potential for creating unlimited vocabulary of words in any language (or languages) and accents;
- high recognition and adaptation accuracy;
- different modes of operation depending on the setting of some parameters: (1) speaker-dependent mode, where the system learns to recognise only a single user; (2) multiple speakers mode, where the system is trained on several speakers; (3) speaker independent mode, where the system can potentially recognise any speaker that speaks with a pronunciation and language that is tolerated by the system;
- fast learning and adaptation what concerns adding new words and speakers if necessary.

The ASN method and system are implementation invariant. They can be implemented as software or/and hardware with the use of either conventional or new techniques. The applicability is broad and spans across all application areas of computer and information science where systems that communicate with humans in a spoken language (‘hands-free and eyes-free environment’) are needed. This includes:

- Voice dialling, especially when combined with "hands-free" operation of a telephone system (e.g. a cell phone) installed in a car. Here a simple vocabulary that includes the spoken digits and some other commands would be sufficient. The adaptive noise cancellation reduces significantly the effect of the engine.
- Voice control of industrial processes. Noise cancellation deals with the background noise inevitably present in the industrial environment.
- Voice command execution – the controlled device could be any terminal in an office. This provides a means for people with disabilities to implement simple tasks in an office environment.
- Voice control in an aircraft.
- Speech data entry systems operating in a noisy environment.
- etc.

References

Astrom, K., G. Goodwin, P. Kumar, Adaptive Control, Filtering, and Signal Processing. New York: Springer-Verlag, 1995.

Iliev, G., Kasabov, N., Adaptive noise cancellation for speech applications, Proc. of ICONIP'99, November 1999, Perth, Australia, IEEE Press (1999), pp. 192-197.

Iliev. G., N. Kasabov, Adaptive Filtering with Averaging in Noise Cancellation for Voice and Speech Recognition, (Proceedings of the ICONIP/AMZIIS/ANNES'99 Workshop "Future directions for intelligent systems and information sciences, Dunedin, 22-23 Nov.1999), N.Kasabov and K.Ko (eds), pp. 71-75.

Kasabov, N and B. Woodford, Rule insertion and rule extraction from evolving fuzzy neural networks: algorithms and applications for building adaptive, intelligent expert systems, 1999 IEEE International Fuzzy Systems Conference Proceedings, Seoul, August 1999, v.III, 1406-1409

Kasabov, N., Adaptation in intelligent multi-modular systems: A case study on adaptive speech recognition, R.Trappl (ed), Proceedings of the European Meeting on Cybernetics and Systems Research - EMCSR'98, Austrian Society for Cybernetic Studies, Vienna, 14-17 April (1998) 622-627.

Kasabov, N., ECOS - A framework for evolving connectionist systems and the 'eco' training method, in: S.Usui and T.Omori (eds) Proceedings of ICONIP'98 - The Fifth International Conference on Neural Information Processing, Kitakyushu, Japan, 21-23 October 1998, IOS Press, vol.3, 1232-1235

Kasabov, N., Evolving connectionist and fuzzy connectionist systems – theory and applications for adaptive, on-line intelligent systems, In: Neuro-Fuzzy Tech-

niques for Intelligent Information Systems, N. Kasabov and R.Kozma, eds. Heidelberg, Physica Verlag (1999) 111-146

Kasabov, N., Evolving connectionist systems and applications for adaptive speech recognition, Proc. of IJCNN'99, Washington DC, July 1999, IEEE Press

Kasabov, N., Evolving fuzzy neural networks - algorithms, applications and biological motivation, in: T.Yamakawa and G.Matsumoto (eds) Methodologies for the Conception, Design and Application of Soft Computing, World Scientific, 1998, 271-274

Kasabov, N., Evolving fuzzy neural networks for adaptive, on-line intelligent agents and systems, in: O. Kaynak, S.Tosunoglu and M.Ang (eds) Recent Advances in Mechatronics, Springer Verlag , Singapore (1999): Proceedings of the international conference, Istanbul, Turkey, 24-26 May 1999, 27-41.

Kasabov, N., The ECOS framework and the 'eco' training method for evolving connectionist systems. Journal of Advanced Computational Intelligence (1998) vol.2, No.6, 195-202

Kasabov, N., "Evolving connectionist systems for on-line, knowledge-based learning with self-optimisation," IEEE Transactions on Systems, Man, and Cybernetics, submitted, 2000.

Kuo, S., H. Chuang, P. Mallela, "Integrated automotive signal processing and audio system," IEEE Trans. Consumer Electronics, pp. 522-531, Aug. 1993.

Owens, F., Signal Processing of Speech. London: Macmillan, 1993.

Pelton, G., Voice Processing. New York: McGraw-Hill, 1993.

Pelton, H., Noise Control Management. New York: Van Nostrand Reinhold, 1993.

Proakis, J., Digital Communications. New York: McGraw-Hill, 1995.

Regalia, P., .S. Mitra, P. Vaidyanathan, "The all-pass filter: a versatile signal processing building block," Proc. IEEE, pp. 19-37, Jan. 1988.

Regalia, P., "An improved lattice-based adaptive IIR notch filter," IEEE Trans. Signal Processing, pp. 2124-2128, Sep. 1991.

White, R., J., Walker, Noise and Vibration. Chichester: Ellis Horwood Limited, 1982.

Widrow, B., S. Stearns, Adaptive Signal Processing. Englewood Cliffs, NJ: Prentice-Hall, 1985.