POWER HARMONICS ANALYSIS FOR ELECTRICAL DEVICE SIGNATURE IDENTIFICATION USING THE ARTIFICIAL NEURAL NETWORK AND SUPPORT VECTOR MACHINE

NG WIN SIAU
(B.Eng.(Hons), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgement

I would like to convey my most sincere thanks to the following persons whose help and guidance have contributed greatly to this project's success.

First and foremost, my supervisors, Dr. Dipti Srinivasan and Prof. A.C. Liew who have provided me with invaluable advices, so as to guide me through the whole project. Their profound knowledge and experiences in the field of computational intelligence techniques and power system harmonics had been my greatest source of inspiration and insights into the potentials of the project.

Secondly, Mr. Looi Fook Chee from Power Systems Technology lab who provided technical assistance in creating the experimental setup to analyze the power harmonics of electrical devices in the project.

Also, Mr. Y. C. Woo and M. Chandra from Electrical Machine and Drives Laboratory who helped me in setting up measurement systems for the 3-phase devices used in the project.

Last but not least, Mr. Seow Hung Cheng from Power Systems Laboratory who helped me in preparing a workstation for MATLAB simulations.

Table of Contents

Summary iv

Nomenclature vi

List of Figures viii

List of Tables x

List of Relevant Publications xii

Chapter 1 Introduction 1
1.1 Power Harmonics Analysis 1
1.1.1 Transient and Steady State Signals 1
1.1.2 Signal Analysis Techniques 2
1.2 Application of Computational Intelligence in Power Harmonics Analysis 5
1.2.1 Harmonic Detection 6
1.2.2 Harmonic Source Detection 7
1.2.3 Power Disturbance Classification 8
1.3 Device Signature Identification in Nonintrusive Appliance Load Monitoring 9
1.3.1 Electrical Device Categories 10
1.3.2 Device Signature 11
1.3.2.1 Steady State Power 11
1.3.2.2 Transient Characteristics 13
1.3.2.3 Higher Harmonics Information 14
1.3.3 Application of Computational Intelligence Techniques in NALM 15
1.3.4 A Novel Approach to Device Signature Identification 17
1.4 Proposed Power Harmonics Analysis for Electrical Device Signature Identification Using the ANN and SVM 17
1.5 Report Organization 20

Chapter 2 Feature Vector Characteristics 21
2.1 Data Collection 21
2.2 Harmonics Signature Characteristics 27
2.3 Feature Vector Analysis Results 35

Chapter 3 Proposed ANN and SVM Architectures 36
3.1 Input and Output Vector Dimensions 36
3.2 Performance Definition 38
3.3 ANN Architecture 40
3.3.1 MLP and RBF Neural Networks 40
3.3.2 Time Delay Neural Networks 43
3.4 SVM Configuration 46
3.4.1 SVM for Combinations of Classes 47

Chapter 4 Performance of Developed ANN and SVM Classifiers 51
4.1 Classification Using Multi-class SVM-based Model 51
4.1.1 Identifying Combinations of Devices 52
4.1.2 Noise Filtering 53
4.1.3 Scaling of Input to Improve Performance 55
4.1.4 Resource Usage 57
4.1.5 Feasibility of Developed Multi-Class SVM for Power Harmonics Signature Identification 58
4.2 Performance Comparison of ANN and SVM-based Models 59
4.2.1 Training Using Complete Dataset 59
4.2.2 Reduction of Training Set Size 61
4.2.3 Noise Tolerance 64
4.3 Performance on Different Datasets 67
4.4 Harmonic Signature Identification of Three Phase Devices 69
4.5 Identification of Multiples of Similar Model Devices 70
4.6 Performance of TDNN Architectures 73

Chapter 5 MLP Weights Optimization 76
5.1 Preparation of Training Samples 76
5.2 MLP Architecture 77
5.3 GA Algorithm 78
5.4 GA-ANN Combination 81
5.5 Results 83
5.5.1 Evolution of MLP Weights with GA 83
5.5.2 Evolution of MLP Initial Weights Coupled With Backpropagation 86
5.6 Performance of GA-ANN Combination 88

Chapter 6 Conclusion and Recommendations 89

References 92

Appendix A Feature Vector Sets 97

Appendix B MATLAB Codes 102

Appendix C Detailed Classification Results 112

Appendix D ANN and SVM Techniques 118
D.1 ANN Architectures 118
D.1.1 Self Organizing Map (SOM) 118
D.1.2 Multilayer Perceptron (MLP) 120
D.1.3 Radial Basis Function (RBF) Neural Networks 123
D.1.4 Other ANN Architectures 125
D.2 Support Vector Machines (SVM) 127
D.2.1 SVM in Multi-class Classification 130
D.3 Genetic Algorithm – Artificial Neural Network (GA-ANN) Hybrid 132

Summary

In this thesis, a novel idea of nonintrusively identifying the electrical loads present in a power system by analyzing the current waveform harmonics at the power supply mains using the artificial neural network (ANN) or support vector machine (SVM) was developed.

In general, electrical devices' current waveforms are distorted due to the inherent nonlinearity of the devices. From the study of each device's current waveforms, it was shown that different devices produced distinctly different current harmonics, which were used as signatures for the devices. Some devices also produced different harmonic signatures under different modes of operation.

Various ANN architectures such as the multilayer perceptron (MLP), radial basis function (RBF) and time delay neural network (TDNN) and SVM-based classifiers with various kernels including the linear, polynomial and RBF kernels were applied to the harmonic signature identification. A new multi-class SVM technique for non-mutually exclusive classes was developed, to cater to the multiple outputs requirement of this research, and its feasibility was verified.

The ANN and SVM-based classifiers were trained to map phase angles and magnitudes (represented in the complex form) of the current waveform harmonics to the combinations of devices present in the system. The trained ANN and SVM classifiers were then applied to a test set to obtain the classification accuracy. The

generalization performance from a reduced training set size and noise tolerance limits of the classifiers were explored. The results were favorable with the ANN and SVM-based models being able to correctly determine the combinations of the devices present with high accuracy.

Differences in the harmonic signatures from electrical devices of the same model were studied and the MLP was shown to perform classification accurately on current harmonics of a system containing multiples of similar model devices. Classification performance of three phase devices was shown to be higher than that of single phase devices because it had three times the amount of harmonic information from its three phases. Besides that, using the time delay neural network, step change information was utilized to allow identification of large numbers of devices. As the number of devices increases, the process of disaggregating individual signatures from the total load harmonic information would become more difficult without tracking step changes.

The MLP was concluded as the best classifier due to its high accuracy yet low computational resource requirement. However, it also suffered from the problem of having a large number of local minima, thus causing difficulty in optimizing its weights. Evolution of MLP weights using genetic algorithm (GA) was successfully implemented in the search for an optimal initial set of MLP weights while backpropagation algorithm was used to update the MLP weights towards the optimal values.

Nomenclature

Subscripts/Superscripts

i	input/output number
j	input vector number
k	device combination number
n	odd harmonic number
p	connection number to a neuron
q	neuron number

Symbols and Abbreviations

x_i	ith input
X	input vector
z_i	ith output from a neural network's output layer neuron
y_i	ith output
Y	output vector
y'	desired output or class label
nmax	number of harmonics taken into consideration
mmax	number of electrical devices in the setup
E_i	misclassification rate for device i
$\check{E}_i$	false alarm rate for device i
F_i	classification accuracy for device i

F_{avg} average classification accuracy for all devices in the setup

w_p weight of pth input to the neuron

$w_{p,q}$ weight of the connection between neuron p and neuron q

W weight vector

σ Gaussian function width variable

C cost parameter of the support vector machine

I_n RMS magnitude of the nth odd current harmonic

Φ_n phase of the nth odd current harmonic

$s_{i,k}$ minimum magnitude of the ith input, x_i, for all input vectors representing device combination k

$d_{i,k}$ fluctuation range of the ith input, x_i, for all input vectors representing device combination k

$n_{i,k}$ ratio of $d_{i,k}$ to the mean magnitude of the ith input, x_i, for all input vectors representing device combination k

$r_{i,k}$ ratio of the difference between actual laboratory measurements and the mathematically calculated data to the value of the mathematically calculated data

ANN artificial neural network

SVM support vector machine

MLP multilayer perceptron

RBF radial basis function

TDNN time delay neural network

GA genetic algorithm

BP backpropagation

NALM nonintrusive appliance load monitoring

List of Figures

Fig. 1-1	Transient and steady state signals	2
Fig. 1-2	Distorted waveform in time domain	3
Fig. 1-3	Distorted waveform in frequency domain	4
Fig. 1-4	Example of a P-Q chart [24]	12
Fig. 1-5	Harmonics signature of a PC power supply in complex plane [30]	15
Fig. 1-6	Power harmonics analysis for device identification system proposal	19
Fig. 2-1	Experimental setup	22
Fig. 2-2	Signature identification feature vector	23
Fig. 2-3	Summation of individual feature vectors to form two new feature vectors	27
Fig. 2-4	Harmonic signatures	32
Fig. 2-5	Mean fluctuation magnitude of harmonic	34
Fig. 2-6	Maximum fluctuation magnitude of harmonic	34
Fig. 2-7	Ratio of fluctuation to the mean harmonic magnitude	35
Fig. 3-1	ANN and SVM block diagram	38
Fig. 3-2	Average accuracy, F_{avg} against no. of odd harmonics in feature vector	42
Fig. 3-3	Average accuracy, F_{avg} against no. of hidden neurons	42
Fig. 3-4	Proposed MLP architecture	43
Fig. 3-5	TDNN-1 architecture	45
Fig. 3-6	TDNN-2 - Elman Network	45
Fig. 3-7	Mutually exclusive classes	47
Fig. 3-8	Non-mutually exclusive classes	48
Fig. 3-9	Multi-class SVM signature identification	49

Fig. 4-1	Difference between laboratory measurements and mathematical sums	63
Fig. 4-2	Effect of random noise on classification accuracy	66
Fig. 4-3	Signature difference between devices of the same model	71
Fig. 5-1	Effect of varying the number of neurons in the hidden layer on performance	78
Fig. 5-2	Stochastic universal sampling	80
Fig. 5-3	GA algorithm	80
Fig. 5-4	Proposed GA chromosome	81
Fig. 5-5	Fixing stagnant MLP performance by introducing fresh individuals	83
Fig. 5-6	Effect of varying population size on performance	84
Fig. 5-7	Effect of varying child population size on performance	85
Fig. 5-8	Effect of varying the mutation probability on performance	85
Fig. 5-9	Evolution of the MLP performance	86
Fig. 5-10	Effect of varying population size on performance	87
Fig. 5-11	Effect of varying child population size on performance	87
Fig. 5-12	Effect of varying the mutation probability on performance	87
Fig. 5-13	Evolution of the MLP performance	88
Fig. C-1	K-fold test algorithm	113
Fig. D-1	SOM with n inputs and 6 output neurons in a 2-dimensional lattice	119
Fig. D-2	MLP with a single hidden layer	121
Fig. D-3	Perceptron architecture	122
Fig. D-4	RBF neuron architecture	124
Fig. D-5	Elman TDNN Architecture	127
Fig. D-6	SVM Margin	129
Fig. D-7	Example of the Directed Acyclic Graph SVM (DAGSVM)	131
Fig. D-8	Crossover and mutation genetic operations	133

List of Tables

Table 2-1	Individual device feature vectors	24
Table 2-2	Source voltage harmonic components	28
Table 3-1	Variation of SVM parameters	46
Table 4-1	Performance of multi-class SVM on identifying combinations of devices	53
Table 4-2	Range of added noise	54
Table 4-3	Performance of multi-class SVM on filtering noise	55
Table 4-4	Average amplitude of harmonics	56
Table 4-5	Effect of input scaling on performance of multi-class SVM	57
Table 4-6	CPU time and memory usage	58
Table 4-7	Classification accuracy when using laboratory measurements	60
Table 4-8	Classification accuracy when using mean of laboratory measurements	61
Table 4-9	Classification accuracy after reduction of training set size	64
Table 4-10	Magnitude of random noise for each harmonic	65
Table 4-11	Classification accuracy when random noise was added	66
Table 4-12	Classification accuracy of 10-devices set A	68
Table 4-13	Classification accuracy of 10-devices set B	68
Table 4-14	Classification accuracy of three phase devices	70
Table 4-15	Classification accuracy on combinations of multiple devices of the same model	73
Table 4-16	Classification accuracy of TDNNs	74
Table A-1a	10-devices set A name list	97
Table A-1b	10-devices set A feature vectors	97
Table A-2a	10-devices set B name list	98

Table A-2b 10-devices set B feature vectors 98

Table A-3 Combination of devices 98

Table A-4a Three phase devices set name list 99

Table A-4b Three phase devices set feature vectors 99

Table A-5a 4-devices set name list 100

Table A-5b 4-devices set feature vectors 100

Table A-6 8-devices setup database structure 101

Table C-1 Detailed results of SVM-based classifiers in Table 6-1 112

Table C-2 Detailed results of SVM-based classifiers in Table 6-3 112

Table C-3 Detailed results of SVM-based classifiers in Table 6-5 113

Table C-4a Detailed results of ANN classifiers in Table 6-7 114

Table C-4b Detailed results of SVM-based classifiers in Table 6-7 114

Table C-5a Detailed results of ANN classifiers in Table 6-8 114

Table C-5b Detailed results of SVM-based classifiers in Table 6-8 114

Table C-6a Detailed results of ANN classifiers in Table 6-9 115

Table C-6b Detailed results of SVM-based classifiers in Table 6-9 115

Table C-7a Detailed results of ANN classifiers in Table 6-11 115

Table C-7b Detailed results of SVM-based classifiers in Table 6-11 115

Table C-8 Detailed results of ANN classifier in Table 6-12 116

Table C-9 Detailed results of ANN classifier in Table 6-13 116

Table C-10 Detailed results of ANN classifier in Table 6-14 116

Table C-11 Detailed results of ANN classifier in Table 6-15 116

Table C-12 Detailed results of TDNN classifiers in Table 6-16 117

List of Relevant Publications

W.S. Ng, D. Srinivasan, A.C. Liew, "SVM in Multiclass Signature Identification", 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (2003)

W.S. Ng, D. Srinivasan, A.C. Liew, "Evolving Feedforward Neural Network for Harmonics Signature Identification", 3rd International Conference on Hybrid Intelligent Systems (2003)

D. Srinivasan, W.S. Ng, A.C. Liew, "Neural Network-based Signature Recognition for Harmonic Source Identification", manuscript submitted for publication in IEEE Trans. on Power Delivery.

Chapter 1 Introduction

1.1 Power Harmonics Analysis

Power harmonics analysis refers to the study of waveform distortion resulting from the non-linearity of electrical loads such as power electronic devices, fluorescent lighting, inverters, saturated transformers and arc furnaces. There has been a rapid increase in the quantity and power rating of highly non-linear power electronic devices, especially in computer systems and the control of power apparatus and systems. Power harmonics cause voltage distortion which affects sensitive equipment, nuisance tripping of circuit breakers and alter meter readings that are based on zero crossings. Besides that, higher order harmonic currents may cause overheating problems in transformers or electrical wirings. Hence, power harmonics remains as a major power quality problem [1,2,3,4].

1.1.1 Transient and Steady State Signals

Power systems waveforms can be broadly divided into two main categories, namely transients and steady state signals. Transients refer to short duration signals that usually occur during sudden transition of states of electrical devices whereas steady state signals are constant or cyclical signals that are repeated with time when the electrical devices have settled down to a stable state of operation (Figure 1-1).

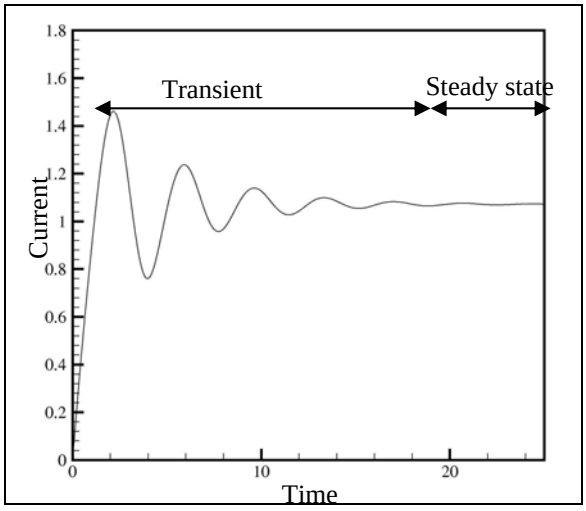


Fig. 1-1 Transient and steady state signals

Due to their short duration, transients are inherently more difficult to analyze compared to steady state signals. Generally, transients require continuous monitoring and recording of the signal at high sampling rate. Techniques such as edge detection are often employed to initiate the capturing of the transient waveforms. On the other hand, steady state signals are obtained only when the waveform has stabilized and thus require lower resolution. Both transients and steady state signals provide vital information in the analysis of power harmonics.

1.1.2 Signal Analysis Techniques

Power harmonics analysis is performed in either the time domain or frequency domain. The frequency information is obtained through techniques such as Fourier Transform and Wavelet Transform of the original time domain signal. The two domains offer two different perspectives of a waveform and are widely used for analysis of power harmonics.

State space simulations of the power system are performed in the time domain, using the available modern digital computers to perform high speed difference equation calculations. Edge detection to track transients or step changes in waveforms is also carried out in the time domain by measuring the rate of change of a signal and comparing it to a threshold.

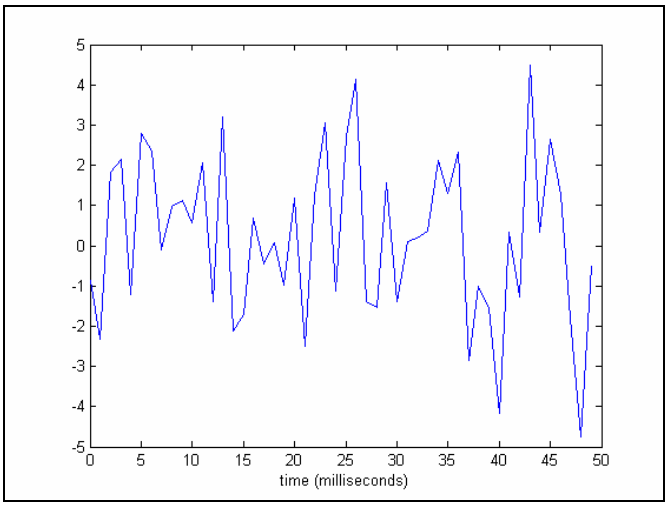


Fig. 1-2 Distorted waveform in time domain

The frequency information allows the extraction of characteristic features in terms of frequency components of the waveform. Frequency domain analysis greatly reduces the data size by representing the time domain information (Figure 1-2), which was recorded at high sampling rate, with the two frequency peaks in the frequency counterpart (Figure 1-3). Therefore, it also simplifies the process of characterizing the waveform using computational intelligence techniques.

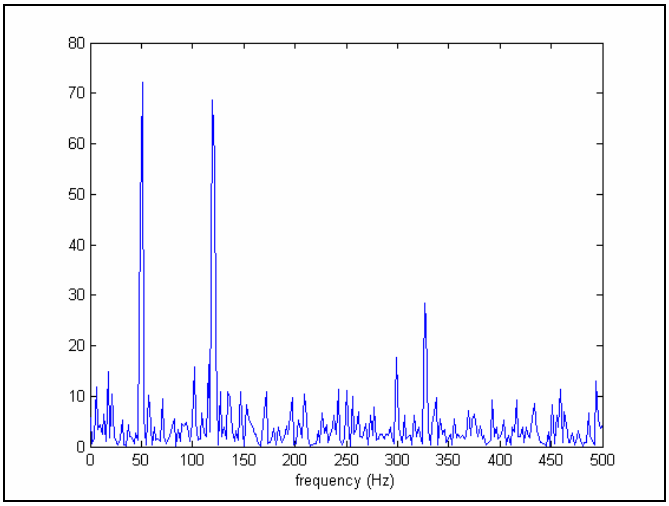


Fig. 1-3 Distorted waveform in frequency domain

Both the Fourier Transform and Wavelet Transform extract the frequency information from a waveform. However, the areas of application for the two methods are slightly different. Fourier Transform is more suited to steady state signal analysis where the frequency components are constant with time. On the other hand, Wavelet Transform is able to analyze transient signals by providing both frequency information and the corresponding locations in time simultaneously. The Fourier Transform and Wavelet Transform are given by equations (1.1) and (1.2) respectively.

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi ft}\, dt \qquad (1.1)$$

$$W(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t) \psi\left(\frac{t-b}{a}\right) dt \qquad (1.2)$$

where *x(t)* is the time domain signal, *X(f)* is the Fourier Transform of *x(t)*, *W(a,b)* is the Wavelet Transform of *x(t)*, *Ψ((t-b)/a)* is the baby wavelet obtained by the stretching by a factor *a* and shifting in time by *b* of the wavelet *w(t)*. In the Wavelet Transform, scale *a* can be converted into frequency *f* while *b* provides time locality for the frequency information *a*.

Apart from Fourier Transform and Wavelet Transform, various other techniques for signal processing are available in the literature [5,6,7]. The techniques include autoregressive modeling (AR) that uses polynomials to perform regression on the waveform [5]. The coefficients of the AR are used in the feature vector for signature identification. There are various other coefficient states to model the waveform, namely cepstral, autocorrelation (ARC), reflection (RC) and Mel-frequency cepstral (MFCC) [6].

Besides that, threshold crossing (TC), differentiation algorithm and sine wave crossing algorithm were discussed in [7]. Due to it computational complexity, FFT cannot be used in real time computation. On the other hand, while the TC is less computational intensive, the DC bias had to be removed manually. Noise is generated by the differentiation algorithm causing it to be less accurate in recognizing some signatures.

1.2 Application of Computational Intelligence in Power Harmonics Analysis

Computational intelligence techniques have been widely applied in the field of power harmonics analysis, especially in situations where conventional methods require high computational power or human expertise.

Expectedly, deployment of computational intelligence techniques for power harmonics analysis has traditionally been focused in the field of power quality. The classification and function approximation capabilities of an artificial neural network

(ANN) have been used in power quality disturbances categorization, fault diagnosis and harmonic sources classification, particularly for power quality analysis or active filter applications [8,9,10,11,12,13,14]. Similarly, support vector machines (SVM) has shown promising potential in power harmonics related pattern recognition [15,16,17].

1.2.1 Harmonic Detection

The extraction of harmonics information from the waveform is conventionally performed using Fourier Transform. However, ANN has been shown to be capable of outperforming the Fourier Transform in terms of speed in extracting harmonics information [9,10,12]. The higher speed of harmonic detection is vital to the operation of active filters that need to react fast to remove harmonics from the power system.

In [9] and [12], the ANNs proposed were trained with samples of simulated current waveforms distorted with odd harmonics from the 3^{rd} to the 7^{th} harmonic with magnitude of up to 33.33% and varying phase angles. The ANNs only required the data from ½ a cycle of the fundamental component, hence the claim of faster processing compared to the Fourier Transform. The ANNs output the magnitude and phase angle of the current waveform harmonic contents. The authors in [10] offered an ANN design that is capable of detecting harmonic components up to the 11^{th} harmonic.

1.2.2 Harmonic Source Detection

The IEEE Std 519-1992 (IEEE Recommended Practices and Requirements for Harmonic Control in Electrical Power Systems) established a set of limits to the amount of current harmonics that is acceptable in the power system [18]. The standard indicates the importance of harmonic sources detection at the power system level, to ensure that each power consumer will play his part in keeping the harmonic distortion level low.

The conventional approach to harmonic source detection is to remove shunt capacitors to eliminate possible redirection of harmonic flow before performing the analysis. However, it tends to disrupt the normal operating condition in certain cases, causing undervoltage problems and missing resonance phenomena. The authors in [18] proposed to go through an elimination process of possible sources instead of changing the system. Paper [19] suggested that negative harmonic power is a sufficient but not necessary condition of being an active harmonic source in a branch and developed a method of monoparameter variation to identify the existence of a harmonic source.

In [20] and [21], state estimation using least square estimators to identify the location of harmonic sources with a few properly placed measurements was used. Measurement placement was based on observability analysis [21]. The proposed method was applied on a large interconnected transmission network. Some inaccuracies in the detection could be due to losses that were not accounted for, estimation errors and modeling errors. In [22], the Kalman filter estimation model with the harmonic injection as a random state variable was used instead. The error

covariance analysis of harmonic injection was used to determine the optimal metering locations.

A constrained ANN was proposed in [8] for the identification of harmonic producing buses. Both data from permanent instruments and portable instruments placed on specific buses were used as the input to the ANN. The ANN was constrained by data from some permanent harmonic instrumentation. The ANN showed high accuracy in determining the harmonic sources.

1.2.3 Power Disturbance Classification

Power harmonics analysis presents a useful form of power disturbance classification, especially since power harmonics is a major contributor to power quality problems. Power disturbance can be divided into steady state events such as supply interruption, undervoltage and overvoltage or transient events such as impulsive transients, oscillatory transients, voltage swell and voltage sag.

Papers [11], [13] and [14] used the self organizing map class (SOM) of ANN to perform the classification of disturbances. In [14], feature extraction for the steady state events and transient events were performed using Fast Fourier Transform (FFT) and Discrete Wavelet Transform (DWT) respectively. Disturbance detection mechanism such as the edge detection was employed to capture the waveform of potential disturbances.

In [23], power harmonics analysis was applied to the stator current and voltages of induction motor drives to perform fault diagnosis using artificial intelligence techniques. The types of faults included rotor and stator asymmetry or dynamic eccentricity and bearing failures. Under stationary condition or steady state, the spectrum lines formed provided the fault signature. With suitable normalization, it could even be extended to a family of induction motors. Unfortunately, in some cases, load anomalies introduced harmonics, thus leading to confusion. An expert system threshold handler was used together with an unsupervised ANN to perform the clustering of spectrum lines and thus fault types, while fuzzy logic was used to determine the severity of the fault.

1.3 Device Signature Identification in Nonintrusive Appliance Load Monitoring

Nonintrusive appliance load monitoring (NALM) refers to the monitoring of electrical circuits from a central location to identify electrical devices that have been connected to the circuit and to track their states of operation. No access to the individual devices is necessary for installing sensors or making measurements. Therefore, instead of having dedicated hardware and wiring to monitor the states of the electrical devices, complex software for signal processing and analysis is required for device signature identification. This section discusses past publications on electrical signal information or analysis techniques to be used in NALM, thus the novelty of the approach taken in this thesis.

1.3.1 Electrical Device Categories

The electrical device operation states can be broadly divided into three categories:- on/off state machine, finite state machine and continuously variable machine [24].

For the purpose of signature identification, the on/off state machines are the easiest to identify because of their binary state. The signatures of this device category are constants that can be accurately determined at any time period of their operation. Examples of on/off state machines include light bulbs and fluorescent lamps.

Finite state machines have multiple distinct states of power consumption. Therefore, a more complicated signature model is required to identify a finite state machine. The model will need to include all the available states and possibly a time reference to characterize cyclical transitions between the states. The state transitions may be automatic (e.g. a washing machine which moves from washing to spinning) or require human input (e.g. a multi-speed fan switched from low speed to high speed).

In the third category, the continuously variable machines, as the name implies, have an infinite number of states within an operational range. Using conventional NALM techniques which will be discussed in the chapter, it is very difficult to accurately identify this category of devices. Examples of the continuously variable machines are the light bulb with a dimmer, sewing machines and variable speed drills.

1.3.2 Device Signatures

In order to form the signature to represent an electrical device, different types of information obtained from the voltage and current measurements of the electrical circuit have been used. These information or features are usually stored in a vector format and thus a device signature is also referred to as the feature vector of the electrical device. Some of the commonly used information are the steady state powers, the transient characteristics and the higher harmonic components [24,25,26,27,28,29].

1.3.2.1 Steady State Power

In [24,25,26,29], the step changes in the steady state aggregate complex power consumption of various commonly used electrical devices were plotted in the P-Q chart (Figure 1-4). Different regions of the P-Q plane would represent different devices and formed the feature vector of the devices. The positive and negative clusters of the P-Q chart were matched to correspond to the on and off state changes of the electrical devices respectively. The Zero Loop-Sum Constraint in [24] stated that the sum of the power changes in any cycle of state transition would be equal to zero. In [25], other information such as the modeling of the state transition cycles of a device, the functional sequences of a group of devices and the time of event that may determine the likelihood of a device being used were associated to the P-Q chart for better identification accuracy.

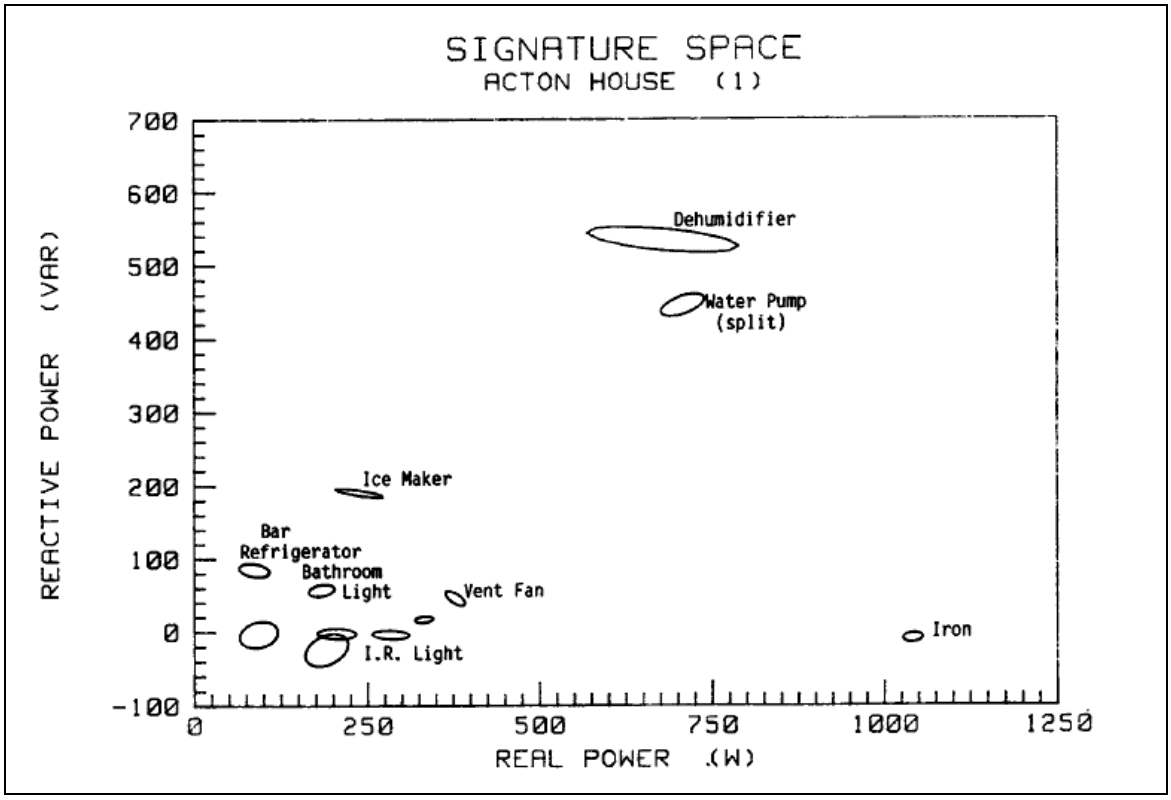


Fig. 1-4 Example of a P-Q chart [24]

Unfortunately, the P-Q analysis did not have sufficient resolution for some devices or device combinations that had almost similar power consumption. Current fluctuation and drift also posed some error in the signature obtained [25]. In [24], the Switch Continuity Principle, which stated that in a small time interval only a small number of appliances were expected to change state in a typical load, was implemented to avoid the deduction of sudden change of combinations of devices that had similar power consumptions. However, electrically identical devices were still indistinguishable.

The detection of events was also a complicated task [24]. It required continuous monitoring of the electrical system. If more than one device were to change its state simultaneously, the event might be misjudged as an unknown device

or some other devices. Devices with multiple states required a corresponding time stamp to track its state transition cycles. Transient effects could also cause the magnitude of the events detected to be inaccurate.

1.3.2.2 Transients Characteristics

In [29], it was proposed a feature vector consisting of the time domain waveform of detected transients to be compared with waveform templates that were shifted in time or offset in magnitude to determine the signature. However, as stated in [25] and discussed in section 1.1.1, transients remain difficult to detect and analyze.

To be able to derive useful information about the transient waveform, the electrical circuit needs to be monitored continuously and at high sampling rate. For the recognition of current demand signatures in the space shuttle telemetry data [27], the actual current waveform was sampled at 10Hz and the time domain data after an edge detection trigger event was used as the signature.

The characteristics of a transient signal depend heavily on the instantaneous state of the electrical system when it occurs. Therefore, the shape of the transient waveform may differ significantly for each measurement instance. Accurate modeling is complicated and will require all the possible states to be taken into consideration. A high time resolution transient detection system is required to tear apart overlapping transients.

1.3.2.3 Higher Harmonics Information

References [24,25,26,29,30] have proposed the use of higher harmonics information to provide higher resolution in disaggregating multiple loads including continuously variable loads in the P-Q chart. The advantage is especially evident with the rapid increase in use of power electronic devices and other devices that produce high amounts of harmonic distortions. Current measurement was preferred to the voltage measurement in power harmonics analysis because of the inherent series inductive and shunt capacitive natures of the electrical wirings causing higher attenuation of voltage with frequency [20].

The use of harmonics information for NALM have been suggested in references [24,25,26,28,29,30]. Reference [30] recorded fuzzy harmonics patterns from various loads. The measurement records from the different types of loads were used to create fuzzy templates consisting of harmonic magnitudes and phase angles in the complex plane (Figure 1-5) for signature identification. Three main harmonic sources were used including the full wave converter, fluorescent tube and iron core. Reference [28] proposed the use of Discrete Wavelet Transform (DWT) preprocessing of the signal to obtain a feature vector based on the normalized energy of each DWT level coefficient.

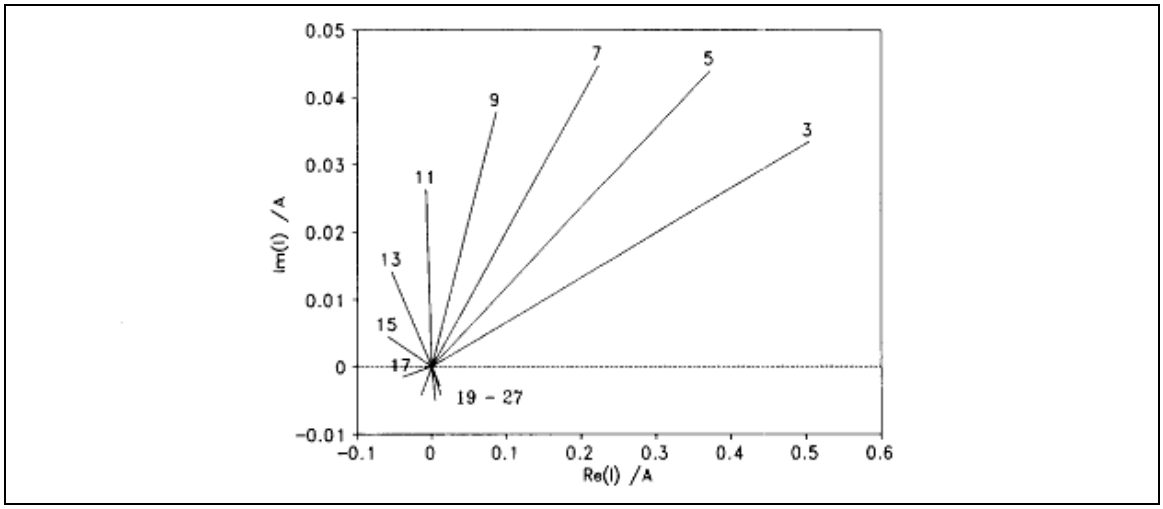


Fig. 1-5 Harmonics signature of a PC power supply in complex plane [30]

However, the references discussed have neither performed a thorough research nor focused on the utilization harmonics information as signatures for NALM. Most of the devices taken into consideration were generally of high power rating and produced low harmonic distortion.

The focus of this thesis was to perform a full study on the potential of the harmonics information in signature identification. This thesis aimed to use the higher harmonics information of the current waveform, drawn at the main incoming source by the electrical devices, to accurately disaggregate the waveform and obtain the individual device signatures.

1.3.3 Application of Computational Intelligence Techniques in NALM

Several researches have applied computational intelligence techniques such as expert systems, fuzzy arithmetic, ANN and SVM for non-intrusive load monitoring [15,25,26,27,30,31].

In identifying an electrical device, the expert system identification algorithm in [25] used the multiple features or characteristics of the household appliances including the effective current, effective voltage, active power, duration and shape of the current transient and current harmonics. The household appliances used for experiment in [25] were divided into categories such as resistive, pump-operated, motor-driven, electronically fed, electronic power control and fluorescent lighting. However, the expert system in [25] depended on the engineer's domain knowledge and required accurate knowledge representation.

In [30], unknown harmonic patterns were identified by calculating the possible contribution from each type of load through solving a set of fuzzy linear equations. The reference only presented a small number of experiments for the case of combinations of devices present in the electrical system.

Reference [26] proposed the use of cascaded ANNs to identify industrial loads. The main feature used was the step change recorded in the P-Q chart and distortion power, D. Varying loads were identified as a locus in the P-Q-D space. The ANN performed binary classifications and formed a family tree in the identification algorithm. Reference [15] performed an experimental analysis of the SVM performance using different kernels. Measurement data in terms of the harmonic information of 10 electrical appliances were used to train and test the SVMs. The reference highlighted the significance of a proper choice of SVM cost parameter to improve classification accuracy.

1.3.4 A Novel Approach to Device Signature Identification

The performance of the ANN and SVM in device signature identification for NALM based on higher harmonics information has yet to be evaluated in the current literature. Higher harmonics information provides highly accurate signatures for the classification of the various electrical devices. While other electrical information can also be obtained from the electrical waveform to further improve the accuracy of the identification, the additional complexity and processing time required do not justify the minimal improvement in accuracy.

The main advantage of the ANN and SVM lies in their ability to perform accurate classification and generalization after training based on available information. The two techniques also eliminate the need for human expertise in determining the necessary structure of the identification models.

1.4 Proposed Power Harmonics Analysis for Electrical Device Signature Identification using the ANN and SVM

The literature review as shown in section 1.2 and 1.3 highlighted the potential of power harmonics analysis using computational intelligence techniques in signature identification problems. Thus far, computational intelligence techniques applied to power harmonics analysis had been limited to system level harmonics source identification and power quality disturbances classification. The notion of power harmonics analysis was directed towards the elimination of power harmonics for power quality improvement. In NALM, various electrical device signature

information such as the steady state power and transient characteristics of the electrical signal have been used to identify the load in an electrical system. However, little research has been done on the use of power harmonics analysis for signature identification.

In contrast, this thesis analyzed the harmonic components from a non-power quality point of view, as valuable information for signature identification. Due to the different power conditioning involved, different categories of devices produce different current waveform distortions and thus different current harmonics. Current harmonics can be used as a form of signature for the device, distinct from that of other devices. From only its current waveform measurement, it is possible to identify the devices present in the electrical power system.

The objectives of the thesis research are as follows:-

- To study the characteristics of the signatures obtained from power harmonics analysis of electrical devices.
- To propose and train the artificial neural network and support vector machine based models for classification of electrical devices based on their power harmonics signatures.
- To non-intrusively identify the electrical devices present in an electrical system using the trained ANN and SVM and power harmonics data from the main incoming supply.

In the course of the research, several ANN architectures such as the multilayer perceptron, radial basis function neural network and time delay neural network were

implemented. A new multi-class SVM for classification of patterns involving non-mutually exclusive classes was proposed to cater to the requirements of the research. Lastly, evolution of multilayer perceptron weights using genetic algorithm was implemented to search for the optimal multilayer perceptron architecture.

In summary, this thesis proposed a novel concept of electrical device identification through power harmonics analysis using computational intelligence techniques such as ANN and SVM. The ANN and SVM models for multi-class classification were developed and trained with current harmonics information from a power source mapped to a corresponding output to show the devices present in the electrical power system. Finally, the trained ANN or SVM was used to identify the devices present from just the current harmonics at the power source (Figure 1-6).

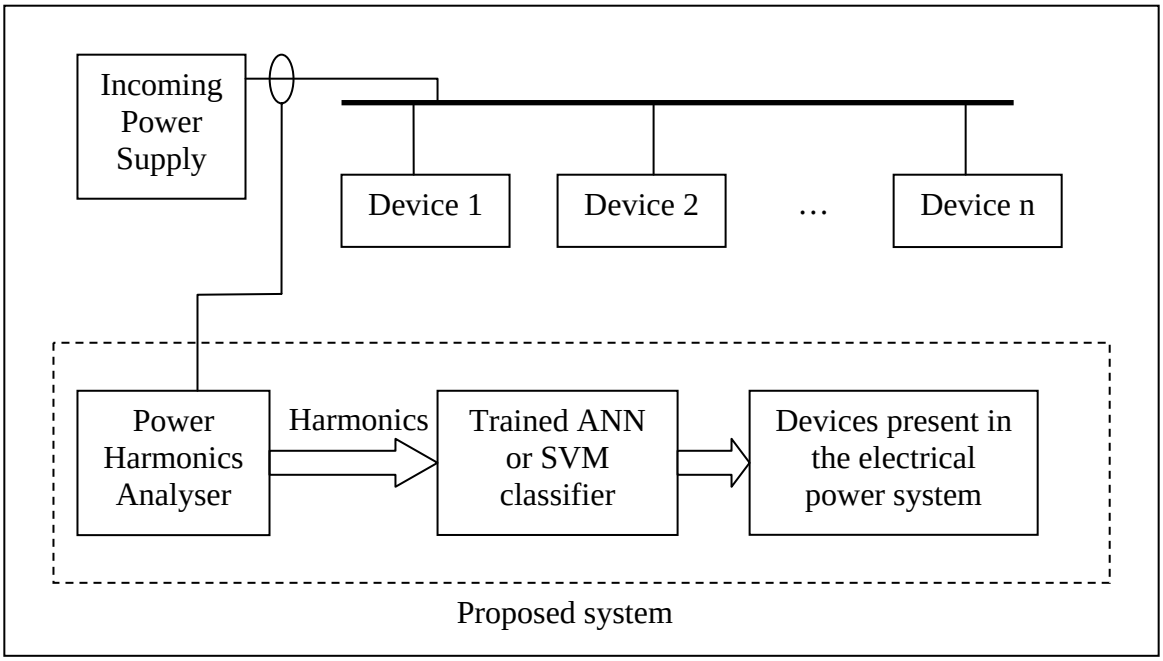


Fig. 1-6 Power harmonics analysis for device identification system proposal

1.5 Thesis Organization

This thesis consists of eight chapters that describe the different aspects of the whole research project. Chapter 2 presents the characteristic analysis of the feature vector of various electrical devices used in the research. Chapter 3 illustrates the proposed artificial neural network and support vector machine architectures for electrical device signature identification based on current harmonics. Chapter 4 contains the training and testing results of the developed artificial neural networks and support vector machines based on different sets of devices and different criteria. Chapter 5 is a discussion of the artificial neural network architecture optimization by using genetic algorithm for evolving the multilayer perceptron weights to improve the classification accuracy. Chapter 6 gives the conclusion of the research and recommendations on some potential future research areas.

Chapter 2 Feature Vector Characteristics

From the literature reviews of nonintrusive appliance load monitoring in section 1.3, the importance of selecting distinctive features to form the signature of an electrical device was highlighted. This chapter discusses the analysis of the proposed feature vectors based on the higher current harmonics information collected from several setups of electrical devices in the laboratory. It aims to verify the presence of distinctive features within the feature vectors before proceeding with the identification using the ANN and SVM classifiers.

2.1 Data Collection

In this research, an experimental setup to represent a simple electrical system with various devices was created. For the main phase of the research, eight single-phase electrical devices were connected in parallel using the laboratory's existing electrical wiring as shown in Figure 2-1 to allow full measurement of all possible combinations of devices. Besides the primary 8-device setup, several 10-device and 4-device setups (Appendix A) were created to allow preliminary tests using the multi-class SVM for signature identification and optimization of ANN weights using genetic algorithm. Measurements were made at the main incoming source of the laboratory using Fluke 41 Power Harmonics Analyser.

The eight devices were switched on and off in steps to allow current waveform measurement of all possible combinations of devices being present at a specific time. With eight devices, 256 discrete states were obtained, each representing different

combinations of devices switched on in the electrical system. For each combination, 18 current waveform readings (each 10 seconds apart) were recorded. In total, 4608 current waveform readings were recorded. The Fluke 41 immediately calculated the harmonics contents (magnitude and phase angle) of the current waveform through fast fourier transform.

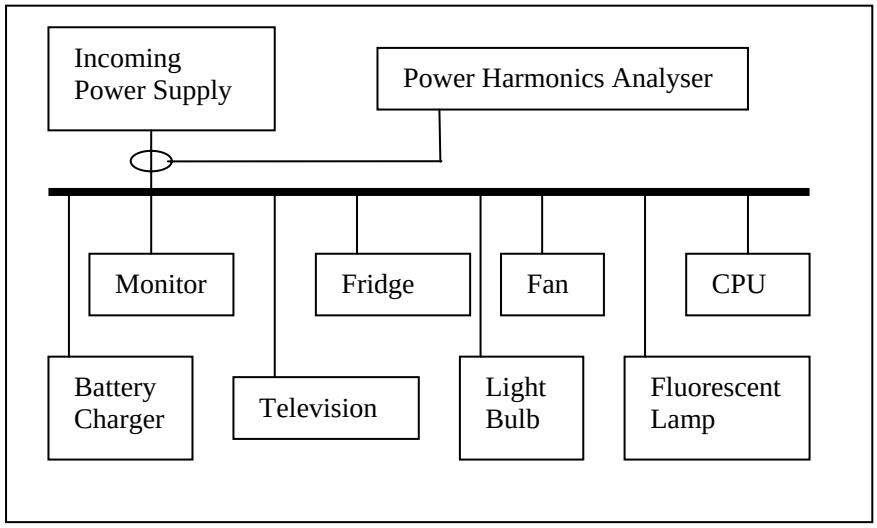


Fig. 2-1 Experimental setup

With the Fluke 41, transient signals were neglected due to the low time resolution of the measuring device. Moreover, as discussed in section 1.3.2.2, transient signals require complex modeling to obtain accurate results.

In the experiments, only the odd harmonics from the fundamental to the 15th harmonic of the current waveforms were of significant magnitude. Therefore, only the first eight odd harmonics (fundamental, 3rd, 5th, 7th, 9th, 11th, 13th, 15th) were chosen as the features of the device signature. For easier representation, the magnitude and phase angle of the harmonics were converted into the complex representation, where each harmonic had a real and imaginary part. With 8 harmonics taken into

consideration, the feature vector presented to the ANN had 16 inputs as shown in equations (2.1a) and (2.1b).

$$x_i = I_{(i+1)/2} \cos\phi_{(i+1)/2} \qquad \text{for i = 1,3,5,7,9,11,13 and 15} \qquad (2.1a)$$

$$x_i = I_{i/2} \sin\phi_{i/2} \qquad \text{for i = 2,4,6,8,10,12,14 and 16} \qquad (2.2b)$$

where x_i is the *ith* input, I_n is the magnitude of the *nth* odd current harmonic and ϕ_n is phase angle of the *nth* odd current harmonic.

Figure 2-2 illustrates how x_1 and x_2 of the input vector were calculated from the real and imaginary parts of the fundamental harmonic according to equations (2.1a) and (2.1b) respectively while the real and imaginary parts of the 11th harmonic were used as x_{11} and x_{12} respectively. Tables 2-1a and 2-1b show the feature vectors of each device in the 8-device setup. The structure and segments of the database containing all the feature vectors created from the 4608 measurements available are shown in Table A-6 in Appendix A.

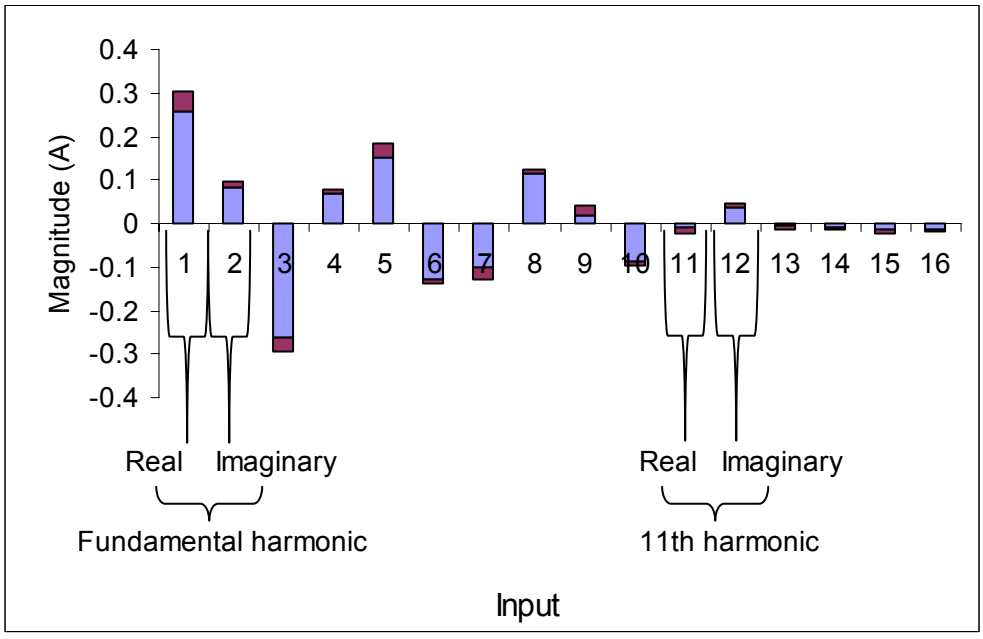


Fig. 2-2 Signature identification feature vector

Table 2-1a Individual device feature vectors

Input No, i	Harmonic	Monitor			CPU			Fluorescent Lamp			Television		
		I (A)	Φ (°)	x_i	I (A)	Φ (°)	x_i	I (A)	Φ (°)	x_i	I (A)	Φ (°)	x_i
1	Fundamental	0.328	15.6	0.316	0.198	29.6	0.172	0.436	-21.7	0.405	0.194	14.5	0.188
2				0.088			0.098			-0.161			0.048
3	3^{rd}	0.282	-15.3	-0.272	0.156	-19.7	-0.147	0.080	-56.9	0.043	0.153	-12.1	-0.149
4				0.075			0.053			-0.067			0.032
5	5^{th}	0.232	-34.6	0.191	0.135	-45.6	0.094	0.021	-65.8	0.009	0.125	-30.5	0.108
6				-0.132			-0.096			-0.019			-0.064
7	7^{th}	0.170	-50.8	-0.108	0.103	-68.1	-0.038	0.036	79.8	0.006	0.085	-46.4	-0.059
8				0.132			0.096			0.036			0.062
9	9^{th}	0.108	-67.2	0.042	0.071	87.0	-0.004	0.048	-88.2	0.002	0.050	-64.3	0.022
10				-0.100			-0.070			-0.048			-0.045
11	11^{th}	0.054	-79.4	-0.010	0.040	59.9	0.020	0.021	-12.7	0.020	0.020	-78.0	-0.004
12				0.053			0.034			-0.005			0.020
13	13^{th}	0.015	-74.9	0.004	0.016	13.5	-0.015	0.020	52.8	-0.012	0.003	71.2	0.001
14				-0.014			-0.004			-0.016			0.003
15	15^{th}	0.014	32.8	-0.011	0.015	-86.1	0.001	0.058	61.1	-0.028	0.015	86.5	-0.001
16				-0.007			-0.015			-0.051			-0.015

Table 2-1b Individual device feature vectors

Input No, i	Harmonic	Battery charger			Fan			Fridge			Light Bulb		
		I (A)	Φ (°)	x_i	I (A)	Φ (°)	x_i	I (A)	Φ (°)	x_i	I (A)	Φ (°)	x_i
1	Fundamental	0.178	-45.5	0.125	0.176	25.0	0.159	0.232	24.1	0.212	0.476	6.0	0.474
2				-0.127			0.074			0.095			0.050
3	3^{rd}	0.047	16.3	0.045	0.025	80.5	0.004	0.181	32.3	-0.153	0.028	11.2	0.027
4				0.013			0.025			-0.097			0.005
5	5^{th}	0.042	-43.8	-0.030	0.010	-10.1	-0.009	0.114	59.2	0.058	0.011	20.0	0.011
6				0.029			0.002			0.098			0.004
7	7^{th}	0.024	42.2	-0.018	0.003	-70.2	0.001	0.067	-66.2	0.027	0.014	26.5	0.012
8				-0.016			-0.003			-0.061			0.006
9	9^{th}	0.001	-19.7	0.001	0.005	-68.4	-0.002	0.069	-5.3	-0.069	0.007	70.4	0.002
10				0.000			0.005			0.006			0.007
11	11^{th}	0.007	-11.4	-0.007	0.004	-32.9	-0.003	0.071	33.8	0.059	0.006	55.9	0.003
12				0.001			0.002			0.039			0.005
13	13^{th}	0.005	-71.7	0.002	0.003	82.5	0.000	0.056	74.0	-0.016	0.007	61.8	0.003
14				-0.005			0.003			-0.054			0.006
15	15^{th}	0.004	37.5	0.003	0.002	-20.8	-0.002	0.043	-54.7	-0.025	0.009	-80.4	-0.001
16				0.003			0.001			0.035			0.009

Similarly, for the 10-device and 4-device setups in Appendix A, the different setups of electrical devices were connected to the main incoming source of the laboratory and the current harmonics were measured and recorded at the incoming point. The feature vectors for each measurement were calculated according to equations (2.1a) and (2.1b). The feature vectors for current harmonics measurements of individual devices of each setup are tabulated in Appendix A.

Due to the large number of possible combinations of devices from the 10-devices setup, instead of experimentally measuring the current harmonics for each combination, the complete data set was created from sums of different combinations of the feature vectors of the individual signatures. Only one of each signature of devices present was added in one combination. The summation process is illustrated in Fig. 2-3. In some experiments, noise of predefined magnitude was added to the sum to simulate a practical situation.

For the 4-devices setup, a total of 26 and 78 feature vectors from random combinations of the four devices were recorded for the training set and test set respectively. The feature vectors recorded were time sequential data (recorded in time steps of 15 seconds), meant to be used for the Time Delay Neural Network (TDNN) that requires the input vectors to be sequential in the time domain.

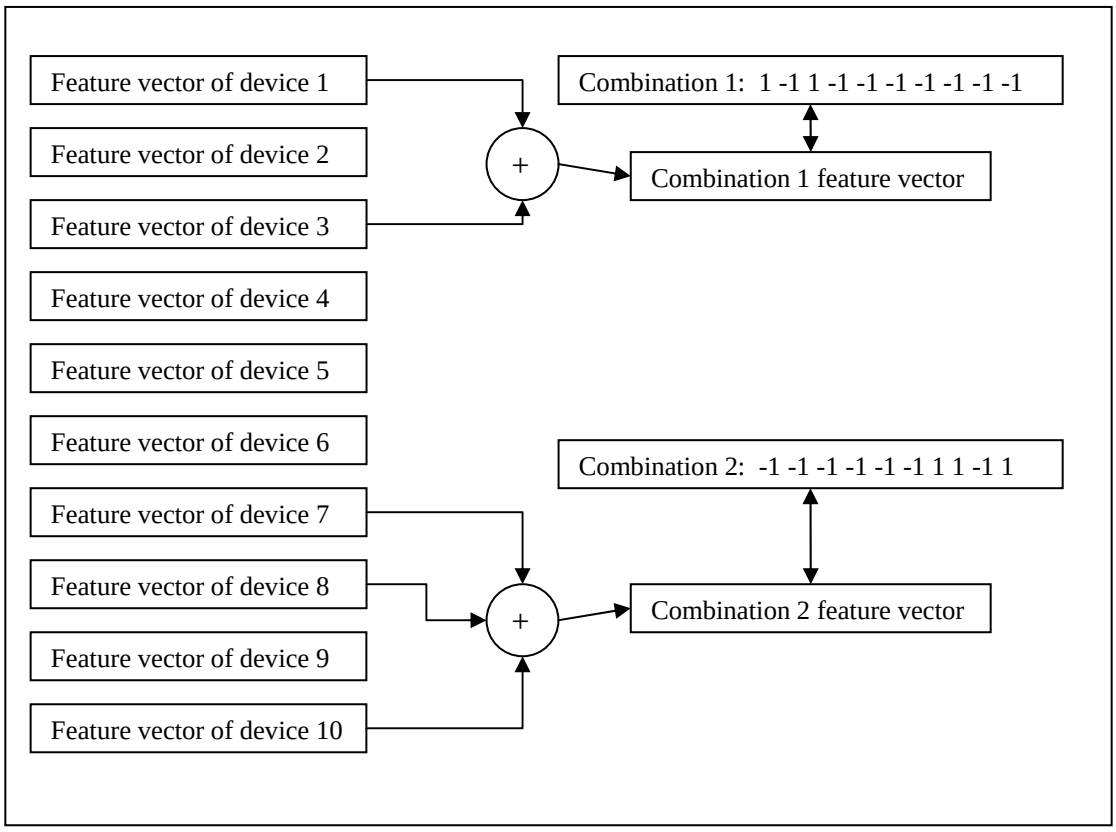


Fig. 2-3 Summation of individual feature vectors to form two new feature vectors

2.2 Harmonics Signature Characteristics

From the measurements made in the experiment, the harmonics magnitude and phase angle of the devices demonstrated small random fluctuations with time. The fluctuations could be due to the fluctuation of the source power supply or inherent electrical characteristics of the devices.

The source voltage measured over the period of experiments had a mean of 231.5V, a standard deviation of 1.26V and higher harmonic contents of below 1.5% of the fundamental (Table 2-2). The effect of the 8 experiment loads on the source voltage was negligible since the wiring resistances were small. High voltage harmonics distortion is expected to change the harmonics signature of the electrical

devices. However, the voltage source of the laboratory was of reasonably low harmonics distortion and also a good representation of the voltage source in common areas where the electrical devices are used. Besides that, highly distorted data due to voltage dips can be pre-filtered and removed to avoid being misclassified by the ANN or SVM.

Apart from these random fluctuations, most of the devices had some short transient states or were capable of multiple modes of operation that could produce significantly different signatures. As a result, for the purpose of this experiment, all the devices were set to operate in a specific mode in all instances and only steady state current harmonics were taken into consideration. This limitation is fair because these devices are expected to operate in the specific mode for large proportions of their operation time.

Table 2-2 Source voltage harmonic components

Harmonic	Magnitude (V)	Standard Deviation (V)
Fundamental	231.46	1.26
3^{rd}	2.70	0.21
5^{th}	2.44	0.24
7^{th}	1.93	0.17
9^{th}	2.14	0.16
11^{th}	0.96	0.09
13^{th}	0.71	0.11

In order to study the range of fluctuations of the various harmonic components, the minimum and maximum values of each harmonic component were calculated according to the following equations:

$$s_{i,k} = \min(x_{i,k}(j)) \qquad (2.2)$$

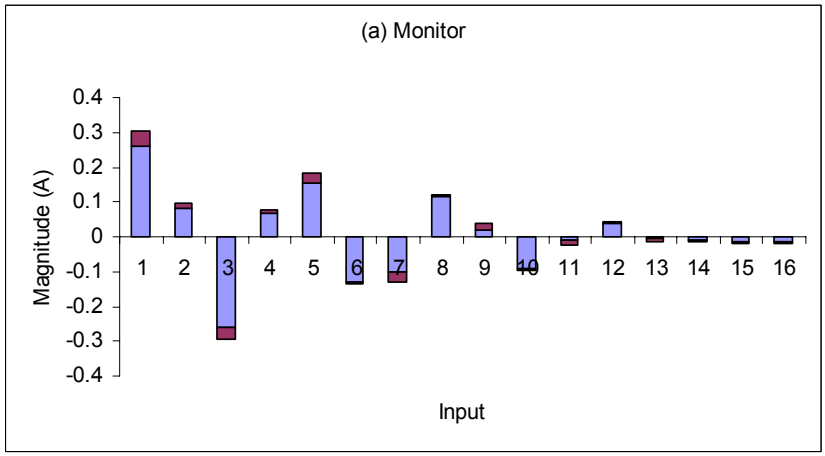
$$d_{i,k} = \max(x_{i,k}(j)) - s_{i,k} \qquad (2.3)$$

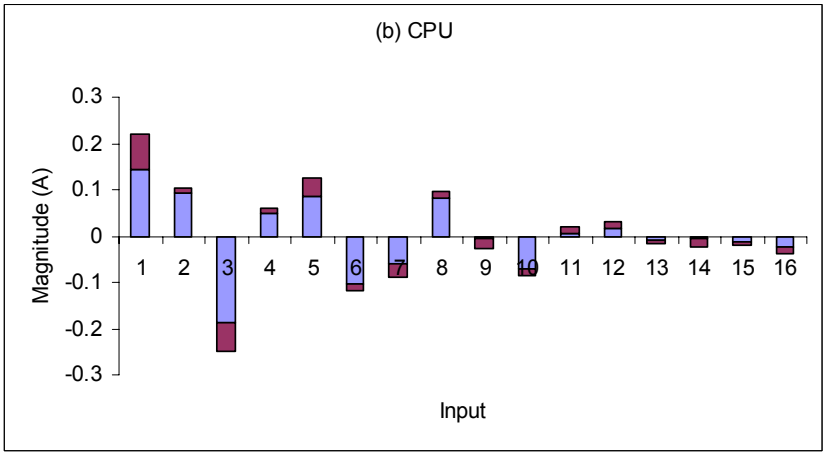
for i = 1, 2, …, 16

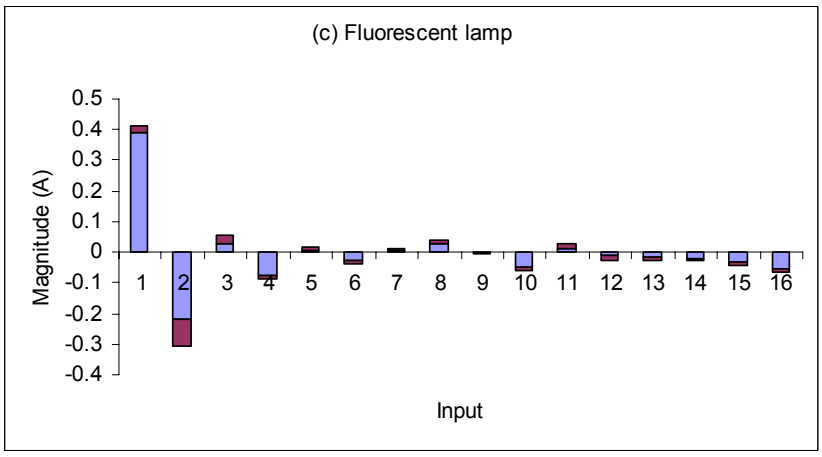
k = 1, 2, …, 256

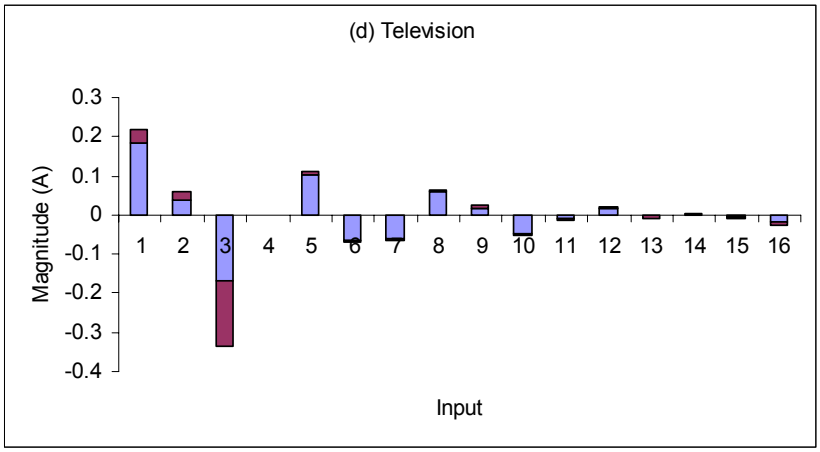
where $s_{i,k}$ is the minimum magnitude, $d_{i,k}$ is the fluctuation range and $x_{i,j,k}$ is the input data in instance, *j*, for input, *i*, of combination, *k*.

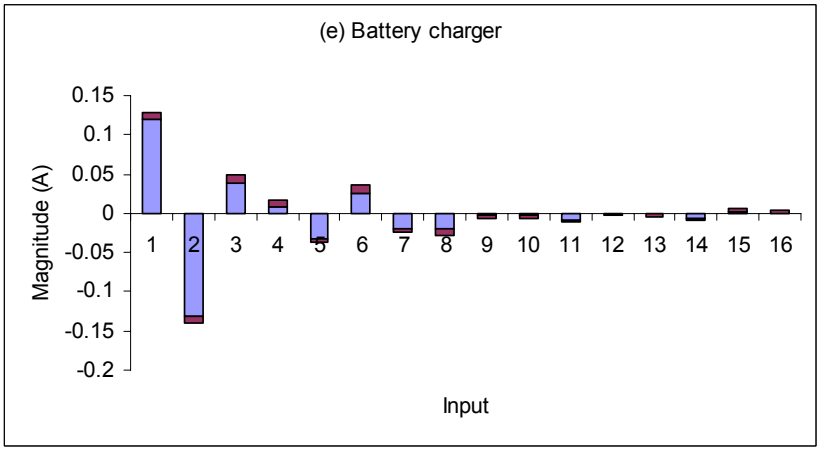
Figure 2-3 shows the distinctive harmonic signatures of all the devices in the experimental setup. The fluctuation ranges, *d*, of the harmonic components are also shown. With these measurements, the fluctuation magnitudes are shown to be at least 5 times smaller than the minimum harmonic magnitudes for each individual device. Besides that, Figure 2-3 shows the characteristics of each harmonic component in the signature. If the fluctuations are large, the characteristic of the harmonic will not be apparent enough for signature identification.

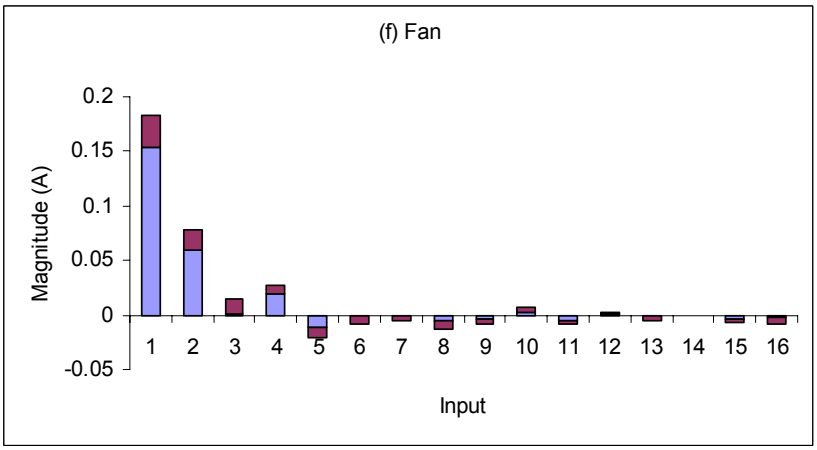

(a) Monitor
Magnitude (A)
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Input


(b) CPU
Magnitude (A)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Input


(c) Fluorescent lamp
Magnitude (A)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Input


(d) Television
Magnitude (A)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Input


(e) Battery charger
Magnitude (A)
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Input


(f) Fan
Magnitude (A)
0.2
0.15
0.1
0.05
0
-0.05
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Input

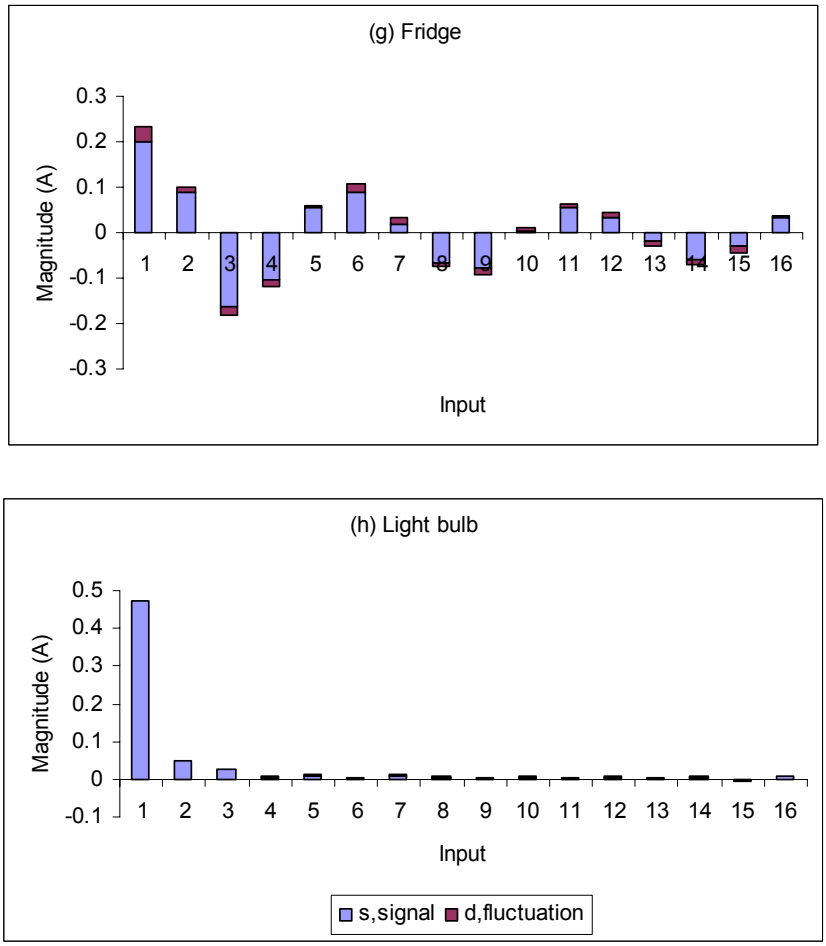


Fig. 2-4 Harmonic signatures

The computer central processing unit (CPU) shows the largest fluctuation magnitude to mean harmonic magnitude ratio (Figure 2-4b) because of the large number of different electrical components inside the CPU that draw independent amount of current from the CPU power supply. When taking measurements of combinations of devices, the computer CPU was set to a constant operation, by repeatedly playing an audio file.

On the other hand, resistive loads such as the light bulb show low harmonic distortion but have characteristically higher power consumption (Figure 2-4h)

compared to the other devices. The monitor and CPU show largely similar shaped signature patterns but are different in terms of magnitude scale (Figures 2-4a and 2-4b).

Equation (2.4) was used to calculate the ratio of fluctuation, d, in equation (2.3) to the mean magnitude of each harmonic component.

$$n_{i,k} = \frac{d_{i,k}}{\sum_{j=1}^{18} x_{i,k}(j) \Big/ 18} \qquad (2.4)$$

where $n_{i,k}$ is the ratio of the fluctuation, $d_{i,k}$, of input, i, to the mean harmonic magnitude for combination, k. There were 18 instances, j, of input data, $x_{i,k}$, for each combination, k.

As the number of devices in a combination increases, the individual fluctuations from each device may sum up to a large fluctuation range for the combination. However, Figure 2-5 shows a fairly low mean fluctuation range, d when averaged over all combinations. The maximum fluctuation, d shown in Figure 2-6 is approximately equal in magnitude to the signature of a single electrical device. Therefore, the ANN or SVM had to be able to model all these fluctuations in order to accurately identify the devices present.

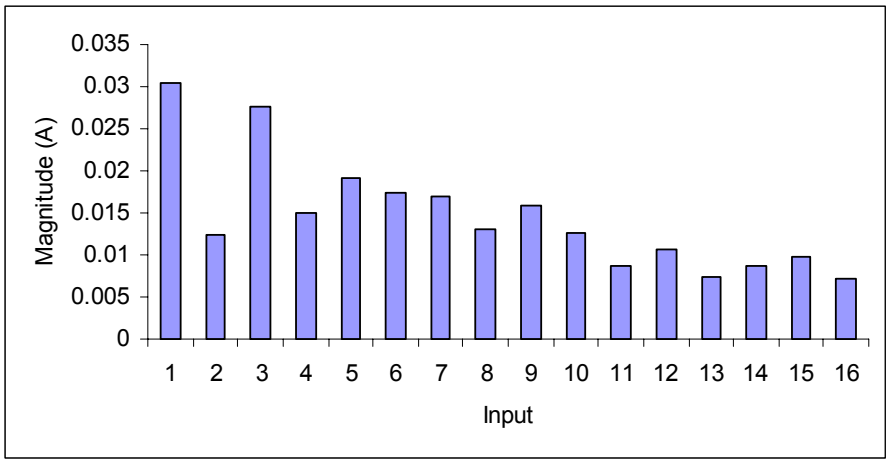


Fig. 2-5 Mean fluctuation magnitude of harmonic

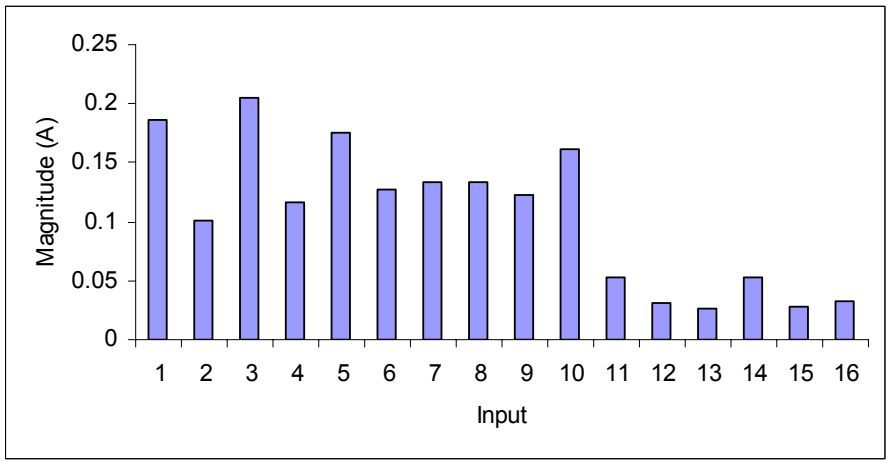


Fig. 2-6 Maximum fluctuation magnitude of harmonic

Figure 2-7 shows the average ratio, *n*, of the fluctuations to the mean harmonic magnitude calculated from the complete dataset of all possible combinations of devices. The spikes at input no 9 and 16 were caused by exceptionally large fluctuations in a few samples. If the fluctuations are treated as noise whereas the mean harmonic magnitudes are treated as the original signal, then the signal to noise ratios (SNR) are above 3dB for most inputs. The SNR is low for the 13th and 15th harmonics (input no 13 to 16) because of the smaller signal magnitude. The ANN and SVM were expected to be able to perform well under these SNR conditions.

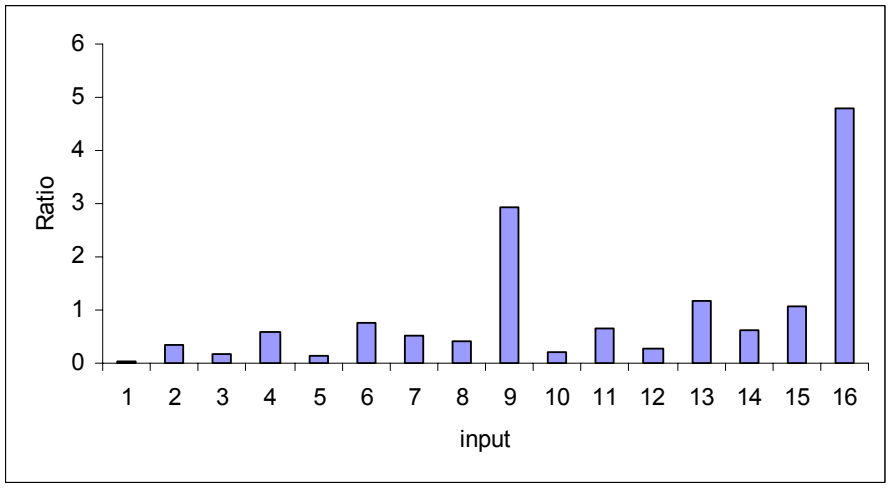


Fig. 2-7 Ratio of fluctuation to the mean harmonic magnitude

2.3 Feature Vector Analysis Results

The study in section 2.2 has shown that the current harmonics feature vectors were able to provide distinctive signatures for both individual devices and combination of devices. Even variable load devices such as the computer displayed prominent signature characteristics that can be identified easily. Although the current harmonics for each combination displayed small fluctuations with time, the SNR conditions were appropriate for the ANN and SVM.

Chapter 3 Proposed ANN and SVM Architectures

After characterizing the feature vectors based on higher current harmonics information to be used for signature identification, this chapter presents the proposed ANN and SVM architectures with the most efficient input and output vector dimensions to perform the classification of devices present in the electrical system.

3.1 Input and Output Vector Dimensions

The ANN and SVM were required to accept the current harmonics information, x_i for i ranging from 1 to 16 as defined by equations (2.1a) and (2.1b), as inputs but in the vector form. The input vector, X, is defined by

$$X(j) = [x_1(j) \quad x_2(j) \quad ... \quad x_i(j) \quad ... \quad x_{n\max\times 2}(j)]^T \qquad (3.1)$$

where $X(j)$ is the jth instance of the input vector, $x_i(j)$ are the jth instances of the ith inputs and $nmax$ is the number of harmonics taken into consideration. Depending on the number of harmonics taken into consideration, the number of input nodes for the ANN or SVM differs. Each node will represent one input x_i. Each harmonic would require two nodes, for the real and imaginary parts of its complex representation. Therefore, the number of input nodes changes in multiples of twos. For example, if the first five odd current harmonics were taken into consideration, then the proposed ANN or SVM would have ten input nodes and the input vector, X, would have a dimension of 10.

For each electrical device taken into consideration, the ANN or SVM would need to have one output node to represent the presence or absence of the device. The output node produced a binary output, y_i, is defined as

$$y_i(j) = \begin{cases} +1 & when \quad device \quad i \quad is \quad present \\ -1 & when \quad device \quad i \quad is \quad absent \end{cases} \qquad (3.2)$$

where $y_i(j)$ is the ANN or SVM output for device *i* corresponding to the *jth* instance of the input vector, $X(j)$.

The output vector, Y, is thus defined as

$$Y(j) = [y_1(j) \quad y_2(j) \quad ... \quad y_i(j) \quad ... \quad y_{m\max}(j)]^T \qquad (3.3)$$

where $Y(j)$ is the *jth* instance of the output vector, $y_i(j)$ is the *jth* instance of the binary output which represents device *i* and *mmax* is the number of electrical devices to be identified. For example, in the 8-device setup where there were eight devices, the output vector, Y, would have a dimension of 8.

Figure 3-1 illustrates the block diagram of the ANN or SVM which shows the inputs and outputs.

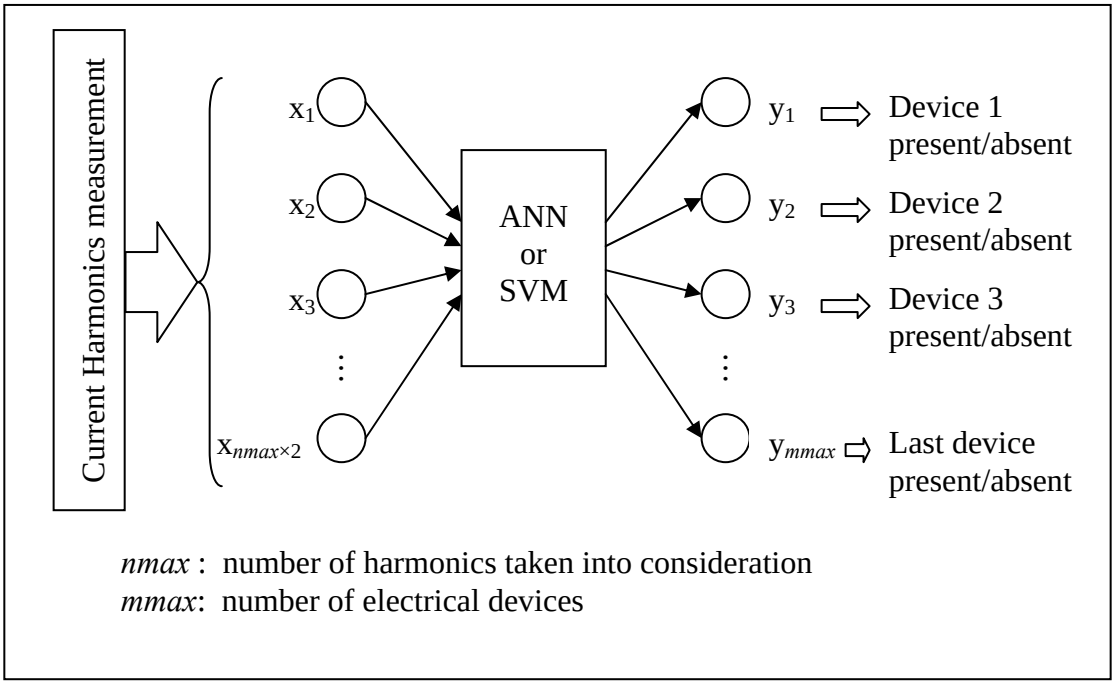


Fig. 3-1 ANN and SVM block diagram

3.2 Performance Definition

The performance of the ANN and SVM were measured by the accuracy in identifying the devices present in the electrical system based on the input vectors in the test set.

For each device, the input vectors in the test set were divided into two groups:- Group A containing all the input vectors of current harmonics recorded when the device was present and Group B containing all the input vectors of current harmonics recorded when the device was absent. In other words, when fed with the input vectors from Group A, the ANN or SVM should produce an output of +1 for that device. On the other hand, when fed with input vectors from Group B, the ANN or SVM should produce an output of -1.

Misclassification refers to the scenario where the device is present but wrongly classified as absent. Therefore, for the calculation of misclassification, only input vectors from Group A were used. The misclassification rate for device *i*, E_i, is defined as

$$E_i = \frac{\sum_{j=1}^{l_{present}} \left| \frac{(1 - y_i(j))}{2} \right|}{l_{present}} \quad (3.4)$$

where $y_i(j)$ is the *jth* instance of the output of the ANN or SVM for device *i* and $l_{present}$ is the total number of input vectors in Group A.

False alarm refers to the scenario where the device is present but wrongly classified as present. Therefore, for the calculation of false alarms, only input vectors from Group B were used. The false alarm rate for device *i*, $\breve{E}_i$, is defined as

$$\breve{E}_i = \frac{\sum_{j=1}^{l_{absent}} \left| \frac{(-1 - y_i(j))}{2} \right|}{l_{abbsent}} \quad (3.5)$$

where $y_i(j)$ is the *jth* instance of the output of the ANN or SVM for device *i* and l_{absent} is the total number of input vectors in Group B.

The classification accuracy, F_i, for device *i* of the ANN and SVM is defined as

$$F_i = 1 - \frac{E_i + \breve{E}_i}{2} \quad (3.6)$$

where E_i and $\breve{E}_i$ are the misclassification rate and false alarm rate of device *i* respectively.

In some tests, the average classification accuracy for all devices was used to evaluate the performance of the ANN and SVM. The average accuracy, F_{avg}, is defined as

$$F_{avg} = 1 - \frac{\sum_{i=1}^{m\max} F_i}{m\max} \qquad (3.7)$$

where F_i is the classification accuracy for device i and $mmax$ is the number of electrical devices.

3.3 ANN Architecture

3.3.1 MLP and RBF Neural Networks

The single-hidden-layer multilayer perceptron (MLP) and radial basis function (RBF) neural networks were used in the classification of the devices present in the experiment. The nodes of any two sequential layers were fully connected.

The number of input nodes in the MLP and RBF neural networks depended on the number of harmonics taken into consideration. To measure the effectiveness of the additional harmonics taken into consideration, the dimension of input vector, X for the MLP was varied from 2 to 16. The fundamental harmonic was first selected. Sequentially, the higher order harmonics were taken into consideration.

The number of hidden nodes for the MLP was also varied from 4 to 60 to choose the optimum configuration for the neural network. On the other hand, the hidden nodes of the RBF neural networks were sequentially added based on maximum variance until a maximum of 300 nodes. The RBF hidden neurons used the Gaussian function with width parameter, σ, equal to 1.

The number of output nodes depended on the number of classes, hence the number of devices to be classified in the experiment. With an initial setup of 8 devices (Table 2-1a and Table 2-1b), the number of output nodes was fixed at 8. The outputs of the neural networks, z_1 *to* z_8, were passed through a signum function such that in the final output, y_1 *to* y_8, all positive values were converted to the integer +1 (device was present) whereas all the negative values were converted to -1 (device was absent).

Figure 3-2 and 3-3 show the plot of average accuracy, F_{avg}, obtained by the MLP against the number of inputs and number of hidden neurons respectively. The optimum number of inputs and hidden nodes were 12 (6 harmonics taken into consideration) and 20 respectively. Above 12 input neurons and 20 hidden neurons, the accuracy performance increase is small compared to the increase in training time. However, in order to minimize the risk of ignoring important information from the 13th and 15th harmonics, a maximum of 16 inputs was still deemed the best. Consequently, the standard network configuration for the MLP to be used in all other tests in this thesis was chosen to be 16-20-8 (16 input neurons, 20 hidden neurons, 8 output neurons) (Figure 3-4).

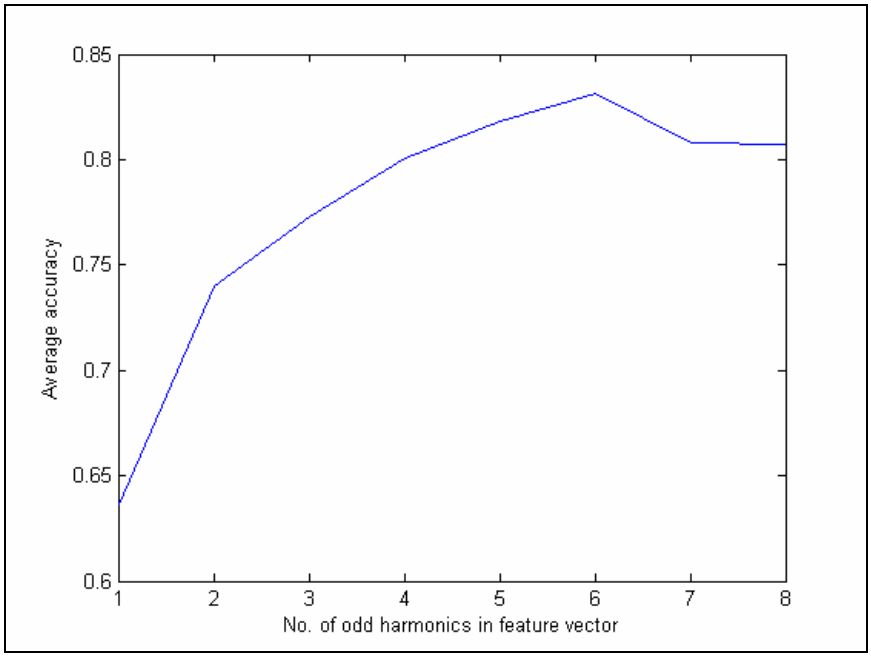


Fig. 3-2 Average accuracy, F_{avg}, against no. of odd harmonics in feature vector

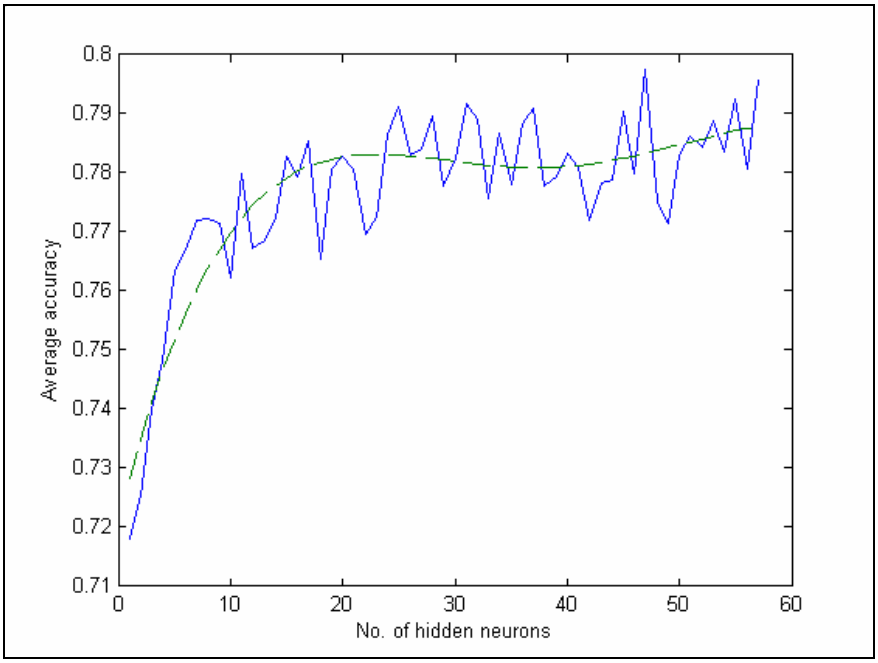


Fig. 3-3 Average accuracy, F_{avg}, against no. of hidden neurons

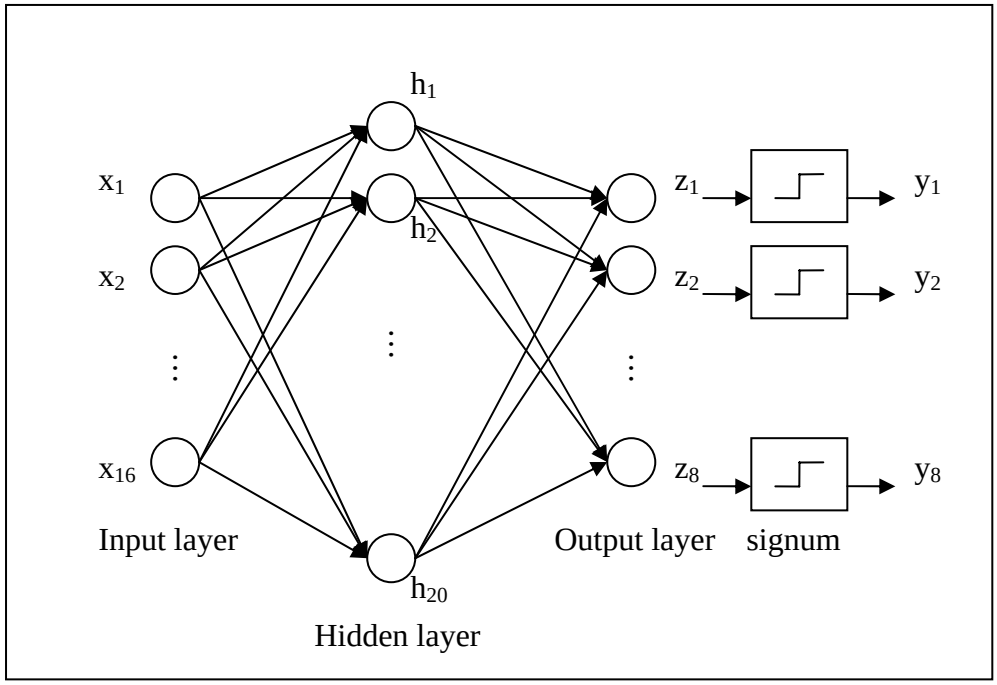


Fig. 3-4 Proposed MLP architecture

Both the gradient descent with momentum [36] and resilient backpropagation [33] training algorithms were used in the training of the MLP. However, the resilient backpropagation performed significantly better than the gradient descent method, showing better and faster convergence within the limit of 3000 epochs set.

3.3.2 Time Delay Neural Networks

To study the effect of using temporal information in the classification of devices, two simple TDNN architectures were designed and trained on a reduced device set of 4 devices (Appendix A – Table A-4). The TDNNs were designed to capture the time step changes in the current harmonics feature vector, distinguishing between changes due to a change in device combination and changes due to small fluctuations in the current harmonics.

With only 4 devices, the dimension of the output vector is 4. The input vectors used for training and testing were time sequential data. Therefore, the time index, *t*, is used to replace the instance index *j*, of the input and output vectors in equations (3.1) and (3.2).

The first TDNN architecture, TDNN-1, was based on the feedforward MLP design with modified input vector (Figure 3-5). The input vector for TDNN-1, $X_{tdnn1}(t)$ is defined as

$$X_{tdnn1}(t) = \left[X(t)^T \quad X(t-1)^T \quad Y(t-1)^T \right]^T \qquad (3.8)$$

where *X(t)* is the current harmonics feature vector at time *t*, *X(t-1)* is the time-delayed current harmonics feature vector and *Y(t-1)* is time-delayed output vector from the TDNN which represents the previous state of the electrical devices. *X(t-1)* and *Y(t-1)* served as the state feedback to the MLP. The training was performed using the backpropagation algorithm.

The second TDNN architecture, TDNN-2, was based on the Elman network, a single-hidden-layer MLP with the addition of a feedback connection from the output of the hidden layer to the input (Figure 3-6). The feedback information underwent a time delay before it was added to the input information. The time delay allowed the Elman network to detect time varying patterns, therefore suitable for the detection of step changes in the current harmonics feature vector. The input of the Elman network was the same as the input vector defined in equation (3.1) except that the instance index, *j*, was replaced with time index, *t*, to represent the time sequential data. Unlike TDNN-1, the time delayed states were stored internally in the network itself. The number of hidden neuron for the Elman network was chosen to be 60 neurons which is

three times the number of hidden neurons in the MLP because Elman networks generally require a much larger number of hidden neurons to perform well [36].

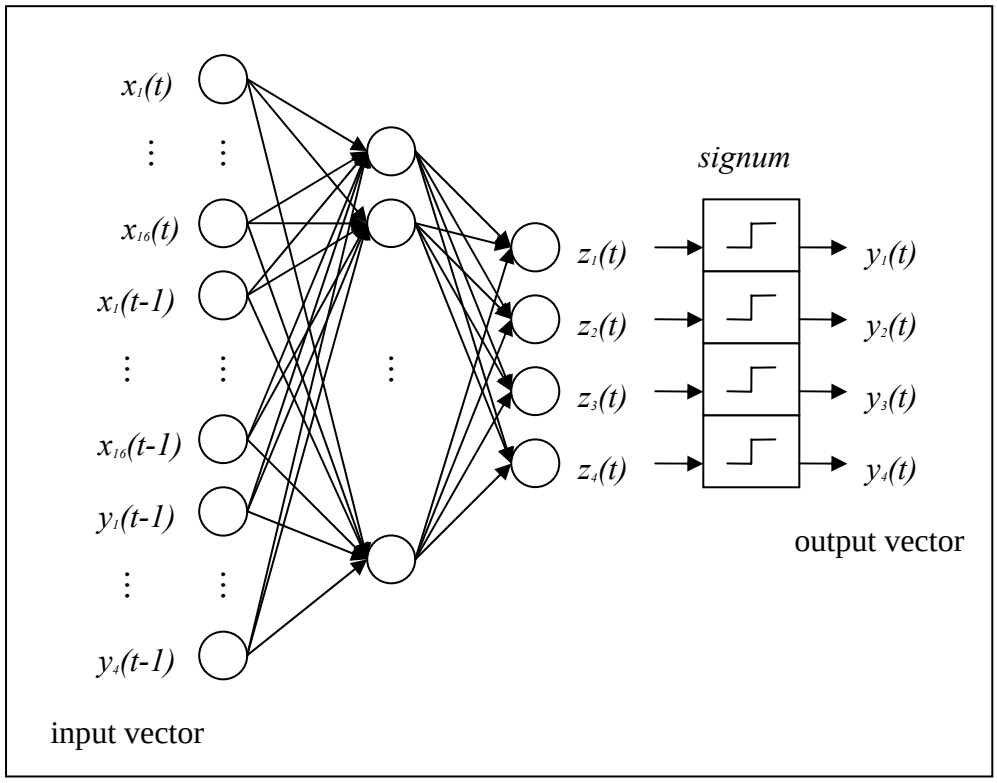


Fig. 3-5 TDNN-1 architecture

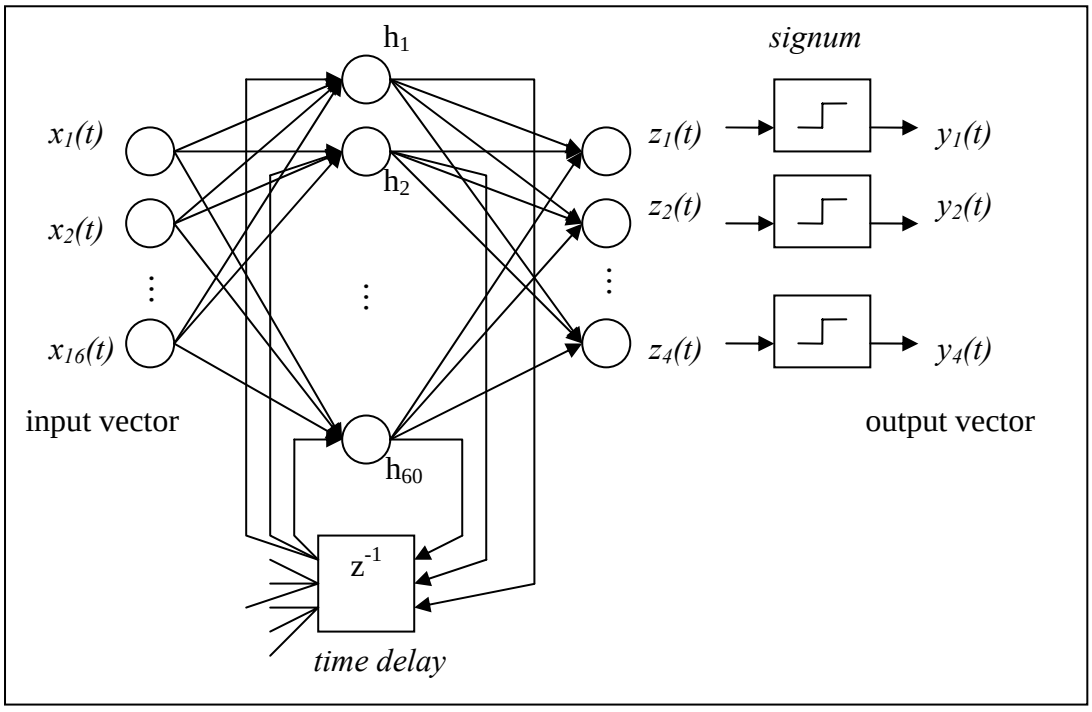


Fig. 3-6 TDNN-2 - Elman Network

3.4 SVM Configuration

In this research, the SVM was implemented in various configurations to obtain the most optimized configuration for each different experimental setup.

Various kernels including the linear, polynomial and RBF kernels were compared. When using the linear kernel, the SVM performs the classification in the original input vector space. The polynomial and RBF kernels perform the classification in the higher order polynomial space and Gaussian function space respectively.

As shown in Table 3-1, the cost parameter, C, was slowly varied from 0.3 to 1.5. Polynomial kernels of degree 2 to 5 were compared while the widths of the RBF functions, σ, were varied from 0.5 to 3.0. C and σ are defined in equations (D.4) and (D.3) respectively.

Table 3-1 Variation of SVM parameters

Parameter	Range
Cost, C	0.5 – 2.5
Polynomial degree	2 – 5
Gaussian function width, σ	0.5 – 3.0

In each experiment, the performance of the SVM linear, polynomial and radial basis function (RBF) kernels were compared. The cost parameter, C and various parameters of each kernel such as the degree of the polynomial kernel and σ value of the RBF kernel giving the best performance were obtained.

In the experiment, the input vector to the SVM classifier was fixed at 16 inputs (8 harmonics taken into consideration). Similar to the ANN, the SVM outputs were passed through the signum function to be converted to +1 or -1. The optimization or training of the SVMs were computed using the SVM-Light software by T. Joachims [57].

3.4.1 SVM for Combinations of Classes

In contrast with the multi-class SVM classification methods discussed in Appendix D.2.1, this thesis proposed a novel approach to identify signatures of combinations of classes which are not mutually exclusive. These classes may be present simultaneously in a single input vector. Hence, it is not possible to produce the necessary outputs with the conventional multi-class SVM approaches [38].

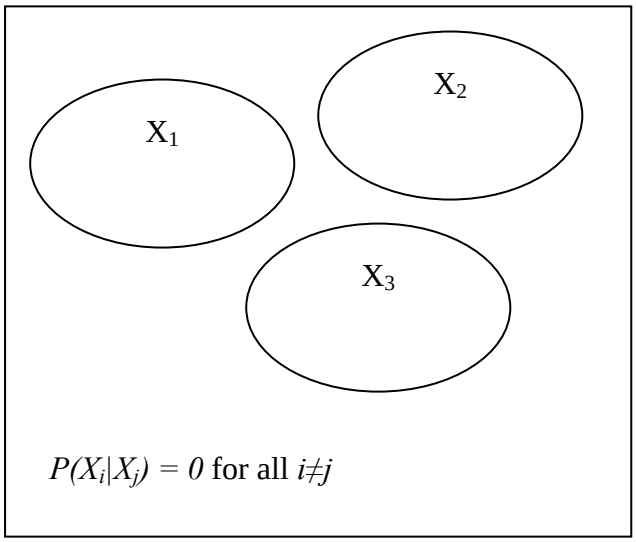


Fig. 3-7 Mutually exclusive classes

Such non-mutually exclusive classes (Figure 3-8) often exist in the current waveform of electrical supply systems with many loads, particularly in this research where the current harmonics in a main electrical cable were analyzed to determine the devices present. In general, the experiments in this thesis required a multi-class output.

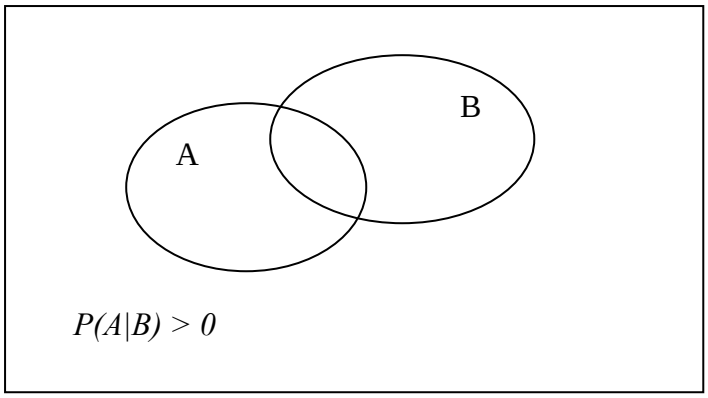


Fig. 3-8 Non-mutually exclusive classes

The SVM was employed to perform multi-class identification to uniquely identify the combinations of devices based on just one set of data, which was the current harmonics. The challenge lies in the fact that SVM is just a two-class classifier whereas for each current harmonics input, it was necessary to produce multiple outputs. As the number of devices increased, the total possible combination also increased exponentially.

An SVM-based model for classifying combinations of classes has been developed. The combination output was first divided into several distinct 2-class (present or absent) problems. The technique was similar to the "one versus the rest" multi-class technique except that no comparison of output values between the SVM classifiers was required to resolve for the final classification.

Due to the unique multi-class nature of the problem, the classification task was divided to several SVM classifiers, one for each device. Each SVM classifier was assigned the task of classifying the presence or absence of its corresponding device. The training for each SVM classifier was done using the same training set or feature vectors but unique output set for its corresponding device.

Finally, the outputs of the individual SVM classifiers were combined to produce the device combination for the particular feature vector. The process is illustrated in Figure 3-9.

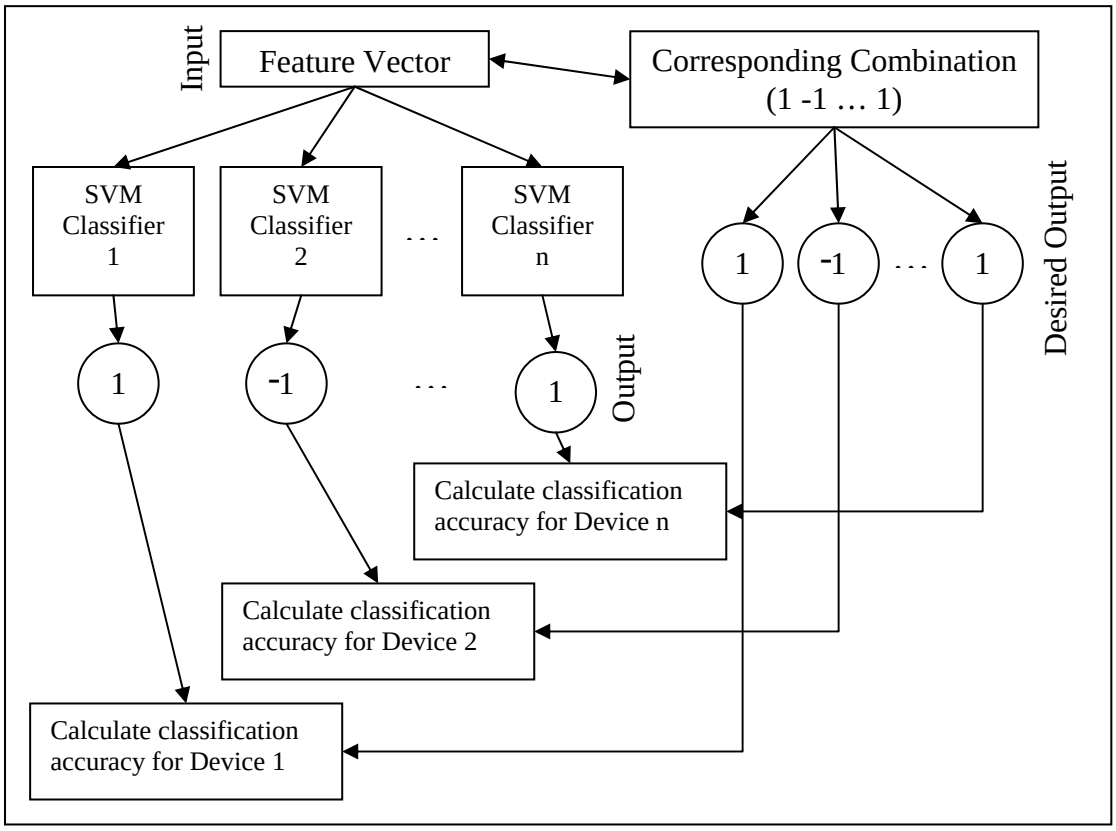


Fig. 3-9 Multi-class SVM Signature Identification

From the same set of training data, the combinations of the various signatures were ascertained. As some combinations in the test set may not be present in the

training set, the classification accuracy relied on the generalization ability of the SVM-based classifier. The SVM-based classifier was required to generalize well and to filter out each individual device signature from potentially noisy signals and all other signatures.

Chapter 4 Performance of Developed ANN and SVM Classifiers

This chapter presents the performance comparison of the developed ANN and SVM classifiers in the identification of electrical devices present in the system from the current waveform harmonics. The performance of the developed multi-class SVM-based models was first evaluated using a 10-device setup. Next, training and testing was performed in several stages to compare the ANN and SVM-based models on the data readings collected from the main setup of eight devices. Subsequently, the MLP was applied on other experimental setups, three phase devices and multiples of devices of the same model. The developed time delay neural network (TDNN) architectures were also tested on time sequential data from an experimental setup of 4 devices.

4.1 Classification Using Multi-Class SVM-based Model

Prior to a full comparison between the ANN and SVM in the identification of electrical devices in an electrical system, a 10-device experimental setup was first employed to study the performance of the newly developed multi-class SVM in section 3.4.1 in generalization and filtering of noise in the signature identification. Through this experiment, the practical performance of the new multi-class SVM was measured to ensure that it would meet the objectives of the thesis.

The current harmonics of a total of 10 individual electrical devices operating at specific modes were measured. The feature vector representing each device is

tabulated in Appendix A (10-devices Set B). The database of input vectors was created using the mathematical summation of individual device's feature vectors as described in section 2.1.

The feature vector used in the classification consisted of the magnitude and phase angle of current harmonics of electrical devices. It included the odd harmonics from the fundamental harmonic up to the 15th harmonic. Each harmonic was represented in the complex form (real and imaginary). Therefore, with 8 harmonics, there were a total of 16 inputs to the SVM.

4.1.1 Identifying Combinations of Devices

By alternating the state of each device, different combinations of devices with unique signatures were generated. From the 2^{10} or 1024 possible permutations of devices as shown in Table A-3 in Appendix A, 1024 input vectors were created where 67% (683 input vectors) were used in the training set while 33% (341 input vectors) were used in the test set. In this experiment, the SVM was forced to generalize well to correctly identify combinations that were not in the training set.

The performance of the multi-class SVM in identifying the combinations of devices is shown in Table 4-1. The classification accuracy, *F*, in Table 4-1 refers to the best percentage correct classification for each device after tuning the SVM parameters. As seen from Table 4-1, the polynomial kernel performed best in this situation. The classification accuracy using the polynomial kernel was above 90% for all devices. The higher inaccuracy in identifying the DC Power Supply at different

loads could be due to the fact that the signatures are similar (scalar multiples of each other). The mobile phone charger also suffered from lower classification accuracy due to the relatively small magnitude of its feature vector.

Table 4-1 Performance of multi-class SVM in identifying combinations of devices

Device	Accuracy, F (%)		
	Linear	Poly	RBF
PC CPU	100	100	100
Monitor	99.42	100	100
PC CPU (Shutdown mode)	100	100	100
DC Power Supply (0.1A)	74.06	98.56	79.25
DC Power Supply (0.5A)	76.95	97.69	76.95
DC Power Supply (0.25A)	65.13	91.64	64.84
DC Power Supply (0.4A)	76.37	93.37	74.93
Notebook computer	100	100	100
Mobile phone charger	53.89	96.25	56.48
Fluorescent lamp	100	100	100

4.1.2 Noise Filtering

To determine the ability of SVM in filtering noise from the signatures, random noise of different maximum magnitude level for different inputs were added to the training and test sets. To create the training database, the input vectors for 2000 random combinations of devices were created according to the summation process explained in section 2.1. Then random noise of the magnitude shown in Table 4-2 was

added to the input vectors. From the total of 2000 random combinations of device signatures with added noise, 67% (1340 input vectors) were used in the training set while 33% (660 input vectors) were used for testing.

Table 4-2 Range of Added Noise

Harmonic	Range of Noise Amplitude
Fundamental	-0.1 – 0.1
3rd	-0.1 – 0.1
5th	-0.1 – 0.1
7th	-0.05 – 0.05
9th	-0.01 – 0.01
11th	-0.01 – 0.01
13th	-0.01 – 0.01
15th	-0.01 – 0.01

Compared to the results in section 4.1.1, the performance of the SVM suffered marginally. The classification results of the DC Power Supply with different loads and the mobile phone charger suffered the most because of weak feature vector characteristics. The SVM with polynomial kernel performed best again, with classification accuracy above 70% for all devices. With this experiment, the Multi-Class SVM technique has proven its ability to perform well under practical situations.

Table 4-3 Performance of Multi-Class SVM on Filtering Noise

Device	Accuracy, F (%)		
	Linear	Poly	RBF
PC CPU	97.75	99.85	98.95
Monitor	98.44	99.84	99.07
PC CPU (Shutdown mode)	93.79	99.70	96.45
DC Power Supply (0.1A)	72.16	91.60	74.81
DC Power Supply (0.5A)	74.38	84.28	77.87
DC Power Supply (0.25A)	66.21	74.89	66.67
DC Power Supply (0.4A)	69.93	73.33	70.64
Notebook computer	98.59	99.29	98.87
Mobile phone charger	61.43	72.49	62.74
Fluorescent lamp	91.39	96.44	93.47

4.1.3 Scaling of Input to Improve Performance

In order to place equal emphasis on every input in the input vector, using the same training database as in section 4.1.2, the magnitudes of all the inputs were scaled according to their average value shown in Table 4-4. After the scaling, the average magnitudes of all inputs were approximately equal. Since the SVM algorithm is structured such that a smaller input magnitude will have less effect on the weights of the SVM hyperplanes, an equal average magnitude of all the inputs will mean equal emphasis on all the inputs. However, there was a risk of amplifying the fluctuations at the higher harmonics, where the ratio of the fluctuation magnitudes to the signal

magnitude was higher as discussed in section 2.2. Therefore, the multi-class SVM would be required to tolerate a higher feature vector fluctuation range for each combination.

Table 4-5 shows an improvement in performance of the multi-class SVM compared to the results in section 4.1.2. With an equal distribution of emphasis on the inputs in the feature vector, the signature characteristics were enhanced. Instead of focusing on the first few inputs that had higher magnitudes, the SVM was able to identify the characteristics more apparent in the higher order harmonics.

Table 4-4 Average Amplitude of Harmonics

Harmonic	Scaling ratio
Fundamental	1:2
3rd	1:2
5th	2:3
7th	4:3
9th	2:1
11th	5:1
13th	5:1
15th	5:1

Table 4-5 Effect of Input Scaling on Performance of Multi-Class SVM

Device	Accuracy, F (%)		
	Linear	Poly	RBF
PC CPU	100	100	100
Monitor	99.85	100	100
PC CPU (Shutdown mode)	99.86	99.86	99.86
DC Power Supply (0.1A)	91.40	91.70	91.10
DC Power Supply (0.5A)	81.86	83.51	82.31
DC Power Supply (0.25A)	71.66	73.55	72.24
DC Power Supply (0.4A)	72.62	73.06	72.91
Notebook computer	99.40	99.55	99.40
Mobile phone charger	70.02	74.73	75.04
Fluorescent lamp	96.57	97.76	97.17

4.1.4 Resource Usage

In the multi-class classification process using the trained SVM classifiers from the developed multi-class SVM, there were 10 comparisons as opposed to the 45 comparisons using the "Pairwise method" or 9 comparisons using the Directed Acyclic Graph SVM method.

The SVM training was performed using a Pentium Xeon 1000MHz processor Linux workstation. For a small training set of about 1300 samples, the CPU time used for the training of the SVM across a fixed variation of the SVM parameters was below 15 minutes. This training time included processing 10 SVM classifiers for the various classes and fine-tuning the parameters.

From Table 4-6, it is shown that the linear kernel SVM used the least amount of CPU time and computer memory. The polynomial kernel SVM which produced the best results used the highest amount of CPU time and computer memory because of the large number of support vectors in the classifier.

Table 4-6 CPU Time and Memory Usage

Kernel Type	CPU Time (seconds)	Memory Usage (MB)
Linear	71	13
Polynomial	779	59
RBF	557	57

4.1.5 Feasibility of Developed Multi-Class SVM for Power Harmonics Signature Identification

The experimental results have shown the feasibility of using the SVM on non-mutually exclusive multi-class classification. Using the methodology shown in section 3.4.1, the binary classifier SVM was extended into a multi-class classifier capable of identifying multiple signatures present in an input signal. In the case of current harmonics signatures from electrical devices, the polynomial kernel performed best and was able to give classification accuracy of between 70%-100% in terms of generalization ability. An accuracy of above 70% is usually considered good for a signature identification problem [38,48]. Scaling of the input vectors had normalized the magnitudes of each input, thus enhancing the characteristics hidden in the higher harmonics and increasing the classification accuracy of the SVM.

4.2 Performance Comparison of ANN and SVM-based Models

After the classification ability of the developed multi-class SVM has been verified, the primary focus of this thesis, which was the study of the ANN and SVM-based model performances in the classification of electrical devices, was executed. In this section, the training and test were based on the 8-device setup elaborated in chapter 2.

4.2.1 Training Using Complete Dataset

In the first stage, the laboratory data measurements (to be referred to as the original dataset from here onwards) as explained in section 2.1 were split randomly into training and testing data to test the ability of the ANN and SVM-based models in classifying the presence or absence of combinations of devices after training on all possible scenarios. In this stage, the original dataset was split such that 66% (3072 input vectors) of the data was used for training whereas the remaining data was used for testing.

A K-fold test was performed to include all data in training and testing. The original dataset was split into 3 equal portions. In three experiments, each of the 1/3 portions (1536 input vectors) was used as the test set sequentially while the remaining data for that experiment was used as the training set. The average classification results from the three experiments were obtained. The K-fold test algorithm and classification equations are described in Appendix C.

Table 4-7 shows the classification result. Although the ANN classified all 8 devices at once for each set of testing sample, the result for each device was tabulated individually. The percentage correct classification was calculated from the number of correct classification of the presence or absence of a device divided by the size of the test set. All classifiers show excellent classification results.

In a further test, instead of using samples from the original dataset for training, the mean value of the input vectors representing each combination was calculated from the original dataset to create a feature vector for each combination. Only the 256 mean value feature vectors (one for each combination) were used for training. The whole of the original dataset (4608 input vectors) was used for testing. Table 4-8 shows the corresponding test result where all classifiers again show excellent classification results.

Table 4-7 Classification accuracy when using laboratory measurements

Device	Accuracy, F (%)				
	ANN		SVM		
	MLP	RBF	Linear	Polynomial	RBF
Monitor	100	100	99.1	99.5	99.5
CPU	99.9	99.8	95.8	99.6	99.2
Fluo. lamp	99.9	99.8	99.7	99.8	99.8
TV	99.6	99.8	86.0	98.9	97.3
Charger	99.9	99.7	99.6	99.8	99.7
Fan	99.9	99.9	57.0	95.4	82.2
Fridge	99.8	99.7	99.9	100	99.9
Light bulb	100	99.8	99.7	99.7	99.7

Table 4-8 Classification accuracy when using mean of laboratory measurements

Device	Accuracy, F (%)				
	ANN		SVM		
	MLP	RBF	Linear	Polynomial	RBF
Monitor	99.9	99.9	94.9	99.5	97.7
CPU	99.4	99.8	77.9	98.5	89.3
Fluo. lamp	99.9	99.9	100	100	100
TV	99.0	99.9	67.5	92.0	70.4
Charger	99.8	99.9	97.4	99.8	99.4
Fan	99.8	98.9	58.6	84.1	58.4
Fridge	99.9	100	100	100	100
Light bulb	100	99.9	99.8	100	99.9

This result proves the ANN and SVM's ability to generalize well to include all the fluctuations in measurement of current harmonics. The reduction in the size of the training set also reduced the training time by a considerable factor especially for the RBF neural network and SVM-based models that performs quadratic programming (QP) optimization.

4.2.2 Reduction of Training Set Size

Theoretically, since all the electrical devices in the experimental setup were connected in parallel and were electrically independent of each other, the current drawn by a combination of the devices should be equal to the sum of the current drawn by each of the devices present in the combination. Based on this assumption, a new

set of training data based on the mathematical sums of current harmonics of individual devices was created.

In order to study the representation accuracy of the mathematical sums to the actual laboratory measurements, the ratio of the difference between the two values to the magnitude of the mathematical sum was calculated as follows:

$$r_{i,k} = \frac{mean(x_{i,k}(j)) - m_{i,k}}{m_{i,k}} \qquad (4.1)$$

where $r_{i,k}$ is the ratio of the difference between actual laboratory measurements, $x_{i,k}$ and mathematically calculated data, $m_{i,k}$, to $m_{i,k}$ for input, i, of combination, k.

When compared with the actual laboratory measurements, the mathematical sums showed some differences. Figure 4-1 shows the average ratio, r, of the difference to the magnitude of the mathematically calculated training data for each input. Inputs 2 and 4 that represent the imaginary components of the fundamental harmonic and 3rd harmonic respectively show a high value of r because the sine component changes more rapidly with phase angle at small phase angles of less than 60^o. After removing samples with large difference resulting from voltage dips and ignoring the spikes at input number 2, 4, 9 and 11, the average ratio was shown to be below 0.5. It was concluded that the mathematical sums provided a good estimate of the actual measurements.

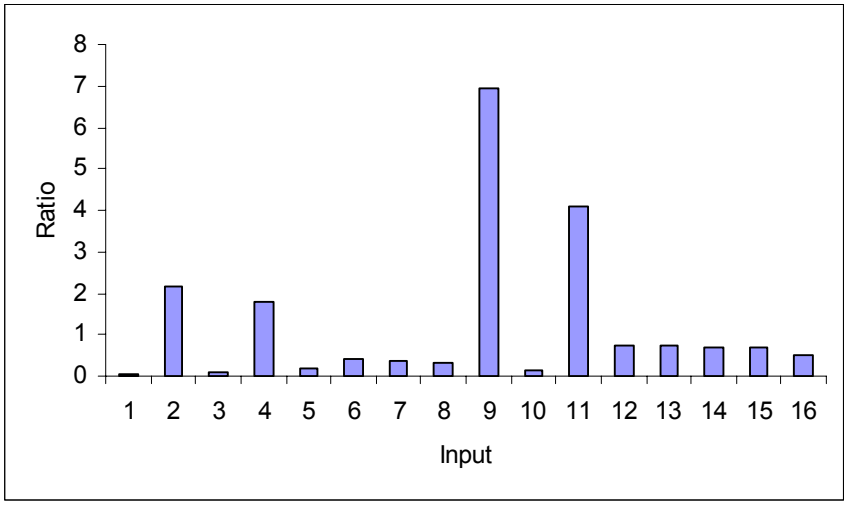


Fig. 4-1 Difference between laboratory measurements and mathematical sums

The newly created training set was used to train the ANN and SVM. The whole original dataset was used for testing of the SVM-based models and RBF neural network. Due to the problem of a large number of local minima, if the initial weights were not selected correctly, the MLP would not achieve the best performance. Therefore, the training and testing for the MLP were repeated 100 times with different random initial weights to reach the global minimum. When training the MLP, 66% (3072 input vectors) of the original dataset was used for validation stop (early stopping) to avoid over-fitting whereas the remaining data (1536 input vectors) was used for testing.

Table 4-9 shows the classification result. There was no apparent best classifier between the ANNs and SVM-based models. Although the average classification accuracy has dropped, it is still above 85%. The reasonable reduction in accuracy is however greatly compensated by the ability of the ANN and SVM to perform classification from just information of individual devices, thus a great step towards non-intrusive monitoring.

Table 4-9 Classification accuracy after reduction of training set size

Device	Accuracy, F (%)				
	ANN		SVM		
	MLP	RBF	Linear	Polynomial	RBF
Monitor	98.5	99.8	92.5	99.4	98.7
CPU	86.9	87.4	74.3	74.5	75.0
Fluo. lamp	99.8	99.5	99.9	99.9	99.9
TV	67.9	88.0	63.6	90.5	78.5
Charger	66.1	65.1	69.7	71.7	70.3
Fan	62.1	69.1	66.0	68.0	68.4
Fridge	98.8	98.8	99.9	99.9	99.9
Light bulb	97.9	79.2	93.1	95.0	94.5

4.2.3 Noise Tolerance

In the final stage of the experiment on 8-device setup, random noise of magnitude specified in Table 4-10 was added to the original dataset. The magnitudes in Table 4-10 were based on the magnitude of the current harmonics of individual devices. After adding the noise, the dataset (4608 input vectors) was split into training and testing sets with a ratio of 2:1.

Table 4-10 Magnitude of random noise for each harmonic

Harmonic	Inputs	Noise magnitude
Fundamental	1,2	0.3
3rd	3,4	0.3
5th	5,6	0.2
7th	7,8	0.2
9th	9,10	0.1
11th	11,12	0.1
13th	13,14	0.1
15th	15,16	0.1

For the MLP ANN, the random noise added was varied from 0.1 to 1.5 times the specified magnitude. For each step, the training and testing were repeated 100 times with different random initial weights and the best result was used. Figure 4-2 shows a decrease in the average classification accuracy of the 8 devices with an increase in noise magnitude. Even at 1.5 times the average magnitude, the average accuracy was still above 70%. Hence, it is shown here that the ANN was capable of filtering noise that may be caused by unknown or faulty devices.

Table 4-11 compares the classification accuracy of ANN and SVM classifiers. The performance of the RBF neural network was greatly affected. Although the test results of the SVM classifiers show favorable classification accuracy, the MLP ANN had a significantly lower computational resource requirement.

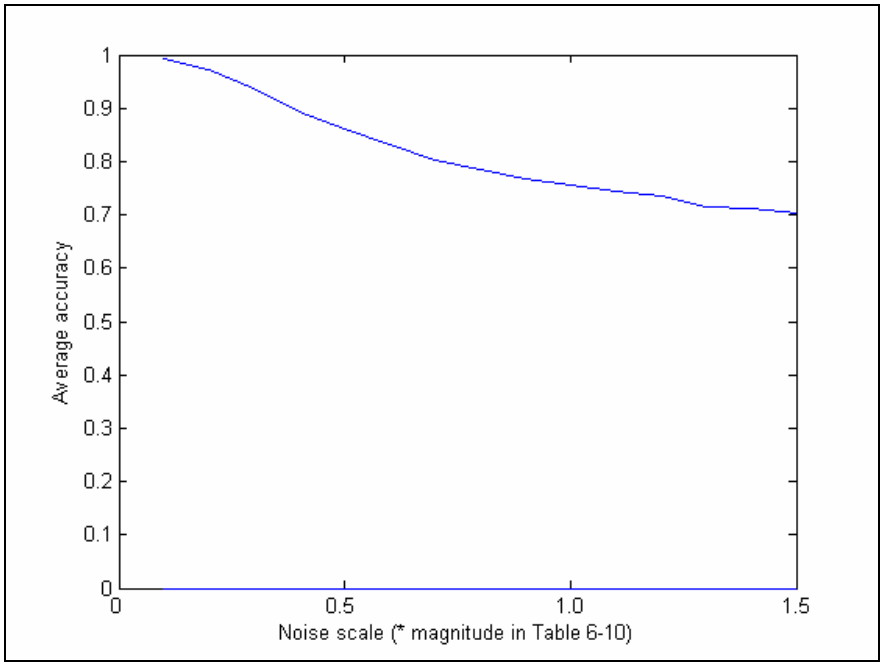


Fig. 4-2 Effect of random noise on classification accuracy

Table 4-11 Classification accuracy when random noise was added

Device	Accuracy, F (%)				
	ANN		SVM		
	MLP	RBF	Linear	Polynomial	RBF
Monitor	88.3	71.8	89.0	89.0	88.3
CPU	71.2	63.8	71.4	71.4	71.4
Fluo. lamp	84.2	58.1	84.7	84.8	84.8
TV	68.0	52.7	64.7	64.7	65.0
Charger	66.8	53.1	66.1	66.5	66.1
Fan	59.5	50.3	61.6	62.0	61.5
Fridge	89.0	67.6	88.6	89.0	88.8
Light bulb	78.6	54.9	78.8	78.8	78.6

4.3 Performance on Different Datasets

In order to test the generalization of the MLP ANN classifier on other types of devices, two additional experiments with 10 devices were set up (Appendix A – Table A-1a and A-2a). Only harmonic signatures of individual devices were measured due to the large number of possible combinations. Using the mean values of the harmonic signatures of individual devices, the harmonic signatures for the combinations of the devices were generated from the sums of the mean values. Random noise was added to simulate the fluctuations expected in the experimental measurements and the resulting dataset was used in the training set.

The first setup (set A) was tested using experimental measurement of random combinations of the device. The training set for set A contained 105 input vectors. The second setup (set B) was tested using mathematical combinations with similar random noise added to it. The training set for set B consisted of 430 input vectors. The training and testing were done on the MLP ANN with 16-20-10 (16 input neurons, 20 hidden neurons and 10 output neurons) configuration. Table 4-12 and Table 4-13 shows the test results of the first and second setup respectively. The MLP ANN has shown to perform well in the two 10-device setups with results comparable to that of the 8-device setup.

Table 4-12 Classification accuracy of 10-devices set A

No	Device	Accuracy, F (%)
1	Monitor	97.7
2	CPU	95.5
3	Fluorescent lamp	99.8
4	Television	75.5
5	Soldering iron	92.0
6	Fridge	99.7
7	Fan	77.8
8	Battery charger	93.8
9	Light bulb	71.4
10	Power drill	93.4

Table 4-13 Classification accuracy of 10-devices set B

No	Device	Accuracy, F (%)
1	PC CPU	99.5
2	PC Monitor	99.5
3	PC CPU (shutdown)	98.8
4	DC Power supply (0.1A)	83.5
5	DC Power supply (0.5A)	76.5
6	DC Power supply (0.25A)	73.0
7	DC Power supply (0.4A)	72.6
8	Notebook computer	99.3
9	Mobile phone charger	77.2
10	Fluorescent lamp	90.5

4.4 Harmonic Signature Identification of Three Phase Devices

An extension of the harmonics signature identification to three phase devices required only an extension of the feature vector dimension. Three phase devices generally have more distinct signatures due to the additional information from the other two phases. The current harmonics of 8 three phase devices (Appendix A – Table A-4a) were measured using the Dranetz 8000-2 Energy Analyser. With three phases and 8 odd harmonics from each phase, there were a total of 48 inputs (real and imaginary for each phase). However, it should be noted that some of the three phase devices have capacitors that could affect the flow of current harmonics thus affecting the current harmonics measured at the supply mains.

Similar to the 10-device setup, the training (1280 input vectors) and testing (1280 input vectors) sets containing combinations of 3-phase devices were created by adding up of the individual 3-phase device signatures. Random noise of the magnitude shown in Table 4-10 was added to the harmonics of each phase to simulate any fluctuation or inaccurate representation of the current harmonics of device combinations. An MLP ANN with 48-20-8 (48 input neurons, 20 hidden neurons and 8 output neurons) configuration was used to perform the classification. Table 4-14 presents the test result that is clearly better than the result of the 8-device single phase devices considering the magnitude of the random noise added.

Table 4-14 Classification accuracy of three phase devices

No	Device	Accuracy, F (%)
1	Motor #1	87.5
2	Motor #2	99.9
3	Motor #2 with capacitors	97.7
4	Inverter #1	99.6
5	Inverter #2 (low frequency)	99.6
6	Inverter #2 (high frequency)	99.6
7	Fluorescent lamp without capacitor	65.1
8	Fluorescent lamp with capacitors	69.1

4.5 Identification of Multiples of Similar Model Devices

In order to study the feasibility of generalizing to devices of the same model and make, the harmonic signatures of 4 computer CPUs and 4 computer monitors of the same model were compared. Figure 4-3 shows the differences between the harmonic signatures.

From Figure 4-3, it was concluded that the differences, which were smaller than the random noise magnitude in Table 4-10, were within the generalization ability of the ANN. The ANN can potentially be used to identify multiples of devices of the same model using only the harmonics signature from one of the devices. Consequently, it would not be possible for the ANN to distinguish two or more devices of the same model.

Referring to set B in the 10-devices setup where 4 DC-power supplies of the same model were used, although each consuming different amount of current, the classification accuracies for the DC-power supplies were low. When studied closely, it was discovered that the harmonics signatures of the 4 DC-power supplies were scalar multiples of each other, hence the difficulty faced by the ANN in distinguishing the individual units.

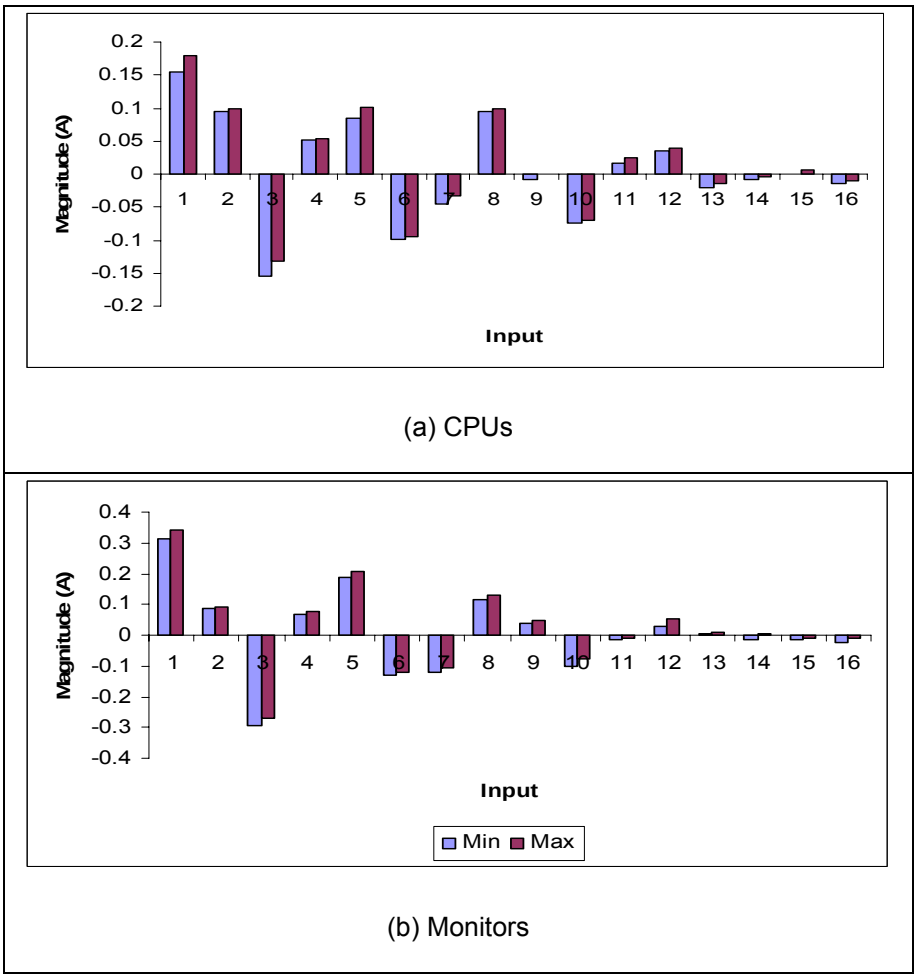


Fig. 4-3 Signature difference between devices of the same model

On the other hand, it should be noted that devices of the same nature or function, but different model or make (such as motor #1 and motor #2 in three phase

devices set) should be separately identified because their harmonics signatures are likely to differ significantly.

Two simulated dataset containing multiples of devices were created using the 10-device set B setup or the same set of devices in Table 4-13. The new datasets aimed to simulate typical office or home setups where multiple devices of the same model would be used simultaneously.

Similar to the technique used in section 4.3, the datasets containing combinations of the devices were created using the mean individual device feature vectors. In the two new datasets, device feature vectors were multiplied by a positive integer to simulate the presence of multiples of the particular device. The first dataset contained multiples of up to 3 of each device whereas the second dataset contained multiples of up to 10 of each device. However, these datasets did not take into account the possible random fluctuations of the feature vectors with time.

Table 4-15 shows the results of the MLP classification accuracy of combinations of multiple devices of the same model. The MLP performed very well with 100% accuracy on the first test set with 818 input vectors. On the second test set with also 818 input vectors, the MLP's classification accuracy began to decrease. The decrease in classification accuracy is expected because as the number of devices increases, the input vector magnitude tends to eclipse the feature vector signature of each device.

Table 4-15 Classification accuracy on combinations of multiple devices of the same model

No	Device	Accuracy, F (%)	
		Up to 3 of each device	Up to 10 of each device
1	PC CPU	100	100
2	PC Monitor	100	100
3	PC CPU (shutdown)	100	100
4	DC Power supply (0.1A)	100	100
5	DC Power supply (0.5A)	100	97.3
6	DC Power supply (0.25A)	100	95.5
7	DC Power supply (0.4A)	100	93.5
8	Notebook computer	100	100
9	Mobile phone charger	100	96.8
10	Fluorescent lamp	100	100

4.6 Performance of TDNN Architectures

Due to the time axis association of the time delay neural network (TDNN), the training data for this part of the experiment were collected in a continuous time sequence. Besides that, more focus had to be placed on the collection of training data that involved changes in the state of the electrical devices. A total of 4 electrical devices (Appendix A – Table A-5a) were switched on and off in random sequences to produce the training data. The whole process was repeated to produce the test set (76 input vectors). Both the training and test sets started from a zero device state so as to allow the TDNN to be initialized properly.

Table 4-16 shows the classification accuracies of the various TDNN architectures in comparison to the accuracy of the MLP architecture proposed in this research. The TDNNs performed reasonably well, but the accuracy was still low compared to the best MLP. However, it should be noted that the results were based on only a limited set and quantity of electrical devices. As the number of devices increases, the classification accuracy of the MLP will drop.

Table 4-16 Classification accuracy of TDNNs

Device	Accuracy, F (%)		
	MLP	TDNN-1	TDNN-2
Fluorescent lamp	100.0	100	100
PC CPU	98.3	96.2	97.2
PC Monitor	95.4	93.4	95.2
Television	91.7	89.5	90.3

During testing since all the inputs of the TDNN-1 were obtained directly from the test set, the TDNN-1 was able to obtain accurate previous state information of the electrical devices, hence the relatively high classification accuracy. However, in actual implementation, the updating of the state information is based solely on the prediction of the TDNN-1 itself. The TDNN-1 output vector itself is routed to the subsequent input vector of the TDNN-1. Consequently, the classification accuracy result in Table 4-16 for TDNN-1 was only an estimate of the actual accuracy, which may be lower. Any prediction error in the current step will be brought forward to the next prediction process thus possibly causing further prediction errors down the chain.

On the other hand, the classification accuracy result for TDNN-2 was a more reliable representation of the practical implementation accuracy since no information other than the current harmonics feature vector sequence was available to the network during testing. The classification accuracy of the TDNN-2 is lower because any misclassification of the previous input vector might cause a misclassification in the next input vector.

The accuracy of the TDNN architectures is low compared to that of the simple MLP. Nevertheless, the TDNN is expected to outperform the MLP when the number of devices taken into consideration increases. As the number of devices increases, the main incoming's current harmonics will be in orders of magnitude larger than that of the individual device's harmonics signatures. Therefore, the MLP will no longer be able to disaggregate the individual harmonics signatures from the instantaneous main incoming current harmonics measurements. Since the TDNN monitors time step changes in current harmonics measurements, it will be able to disaggregate the signatures by keeping track of the devices present and comparing the step changes to the signatures of individual devices.

In order for the TDNN to be used effectively, there is a need to improve the accuracy to nearly 100%. Since the predicted states are used as future inputs, even a small percentage of prediction error may result in a trail of future errors, unless the TDNN is able to correct itself at the next stage.

Chapter 5 MLP Weights Optimization

This chapter illustrates the research extensions towards improving the performance of the MLP towards the objectives of this thesis by evolving the MLP weights using GA.

In the previous experiments, the MLP performance was heavily affected by the choice of its initial weights because of the large number of local minima. Therefore, the GA has been employed to assist the MLP in finding an optimal set of weights. Before the actual selection of the best GA parameters to perform the evolution of the MLP weights, the MLP architecture (number of neurons) to be used in the GA-ANN combination had to be pre-determined. By varying the number of hidden neurons in the MLP, the best performing hidden layer configuration was found.

5.1 Preparation of Training Samples

In studying the feasibility of obtaining the optional set of MLP weights, a small dataset was used for testing and training. The reason for choosing a smaller set was because the number of weights in the MLP increases exponentially with the number of neurons.

In this experiment, 4 devices (Appendix A – Table A-5a) were used for signature identification:- personal computer central processing unit, computer monitor, television and fluorescent lamp. From this, a total of 16 combinations were available and when combined with the transient states and different operating modes of the

devices, a large training (28 input vectors) and test (76 input vectors) set was obtained. For each combination, several readings were taken to ensure that the transient states of the devices were also considered. The whole process was repeated a second time in order to collect the data for creating the validation set.

The feature vector of the samples included the magnitude and phase angle of the odd current harmonics from the fundamental harmonic to the 15^{th} harmonic. The magnitudes and phase angles were represented in the feature vector in the complex forms (real and imaginary). Therefore, in total, each feature vector consisted of 16 inputs, representing 8 harmonics and 4 outputs representing the presence or absence of the 4 devices. The presence and absence were denoted by 1 and –1 respectively in the output.

5.2 MLP Architecture

In this experiment, a single-hidden-layer MLP was used to perform the classification of the device signatures. The hidden layer consisted of 20 neurons which were chosen based on the optimum performance to computational requirement ratio (Figure 5-1). The input and output had 16 and 4 neurons respectively. All neurons in every layer used the tangential-sigmoidal activation function.

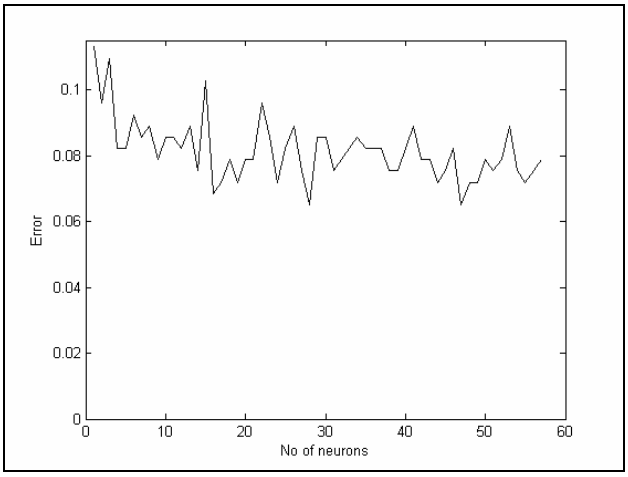


Fig. 5-1 Effect of varying the number of neurons in the hidden layer on performance

The MLP used was a fully connected feedforward network where each neuron in one layer was linked to every other neuron in the previous and next layers. The hidden and output layer neurons each had a bias connected to it.

5.3 GA Algorithm

The GA algorithm used in the GA-ANN combination is illustrated in Figure 5-3. The GA population was evolved up to a maximum of 50 generations. The fitness function given by equation (5.1) was based on the average classification accuracy of the 4 devices. A chromosome was first decoded into MLP weights. Then, an MLP was created according to the MLP weights and its performance was evaluated on the test set. The fitness of the chromosome would depend on the classification accuracy of the corresponding MLP.

$$F_{avg}(u) = \frac{1}{4}\sum_{i=1}^{4}\frac{1}{l}\sum_{j=1}^{l}\left|\frac{y_i'(j) - y_i(j)}{2}\right| \quad (5.1)$$

where $y_i(j)$ was the *jth* instance of the *ith* output of the MLP with the weights defined in chromosome *u* and $y_i'(j)$ was the actual state (present or absent) of the *ith* device and *l* is the total number of input vectors in the test set.

In the evolution process, only mutation was used. Recombination was left out because if two functionally equivalent MLP which order their hidden nodes differently have two different genotypical representations, the probability of producing a highly fit offspring by recombining them is often very low [50]. The lack of exploration ability resulting from the absence of recombination was compensated by the better convergence ability since the accidental destruction of the structure of the hidden nodes was avoided.

The GA employed the stochastic universal sampling (SUS) technique (Figure 5-2) in the selection of the chromosomes for the child population to be mutated. After fitness evaluation, the chromosomes were mapped to contiguous segments of a line. The segment width of each chromosome corresponds to its fitness level. A fitter chromosome would occupy a longer segment. Then a random number was chosen within the range of the line's length. Stochastic universal sampling would create as many pointers as the number of chromosomes to be chosen for the child population and would evenly space them along the line starting from the chosen random number. Finally, the chromosome pointed to by the pointers was selected for mutation.

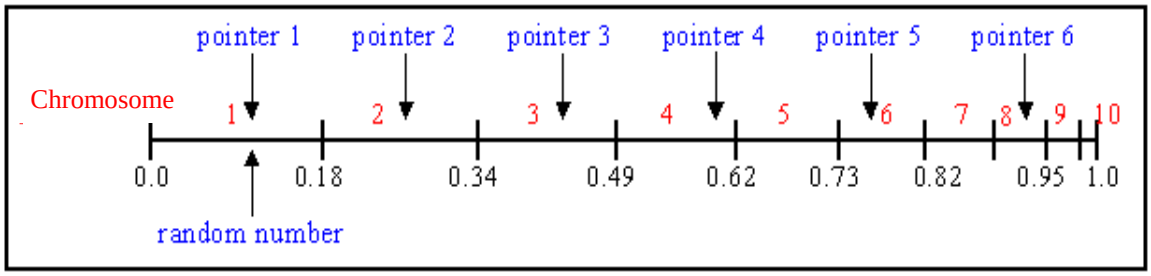


Fig. 5-2 Stochastic universal sampling (SUS)

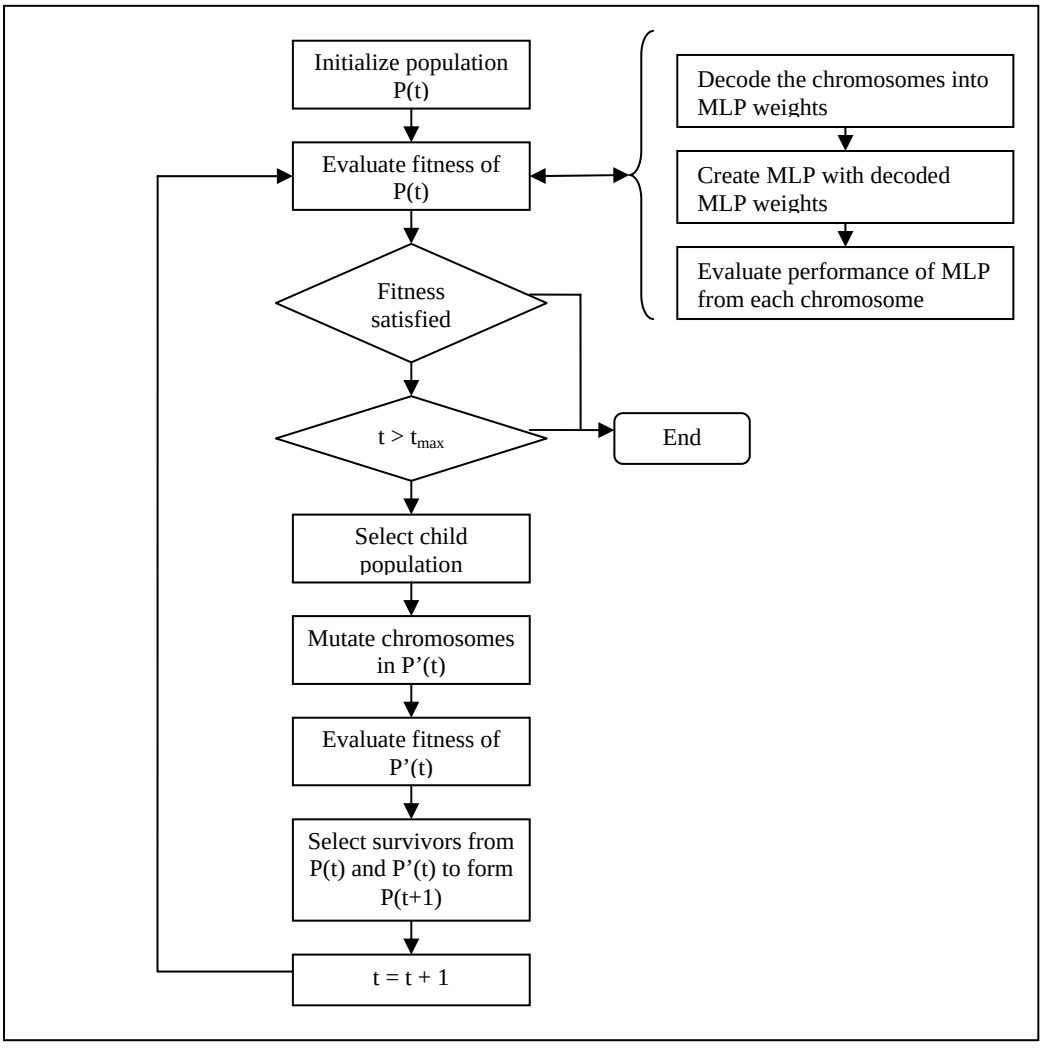


Fig. 5-3 GA algorithm

5.4 GA-ANN Combination

The GA chromosomes were used to represent the weights and biases in the MLP as shown in Figure 5-4. With 16 input neurons, 20 hidden neurons and 4 output neurons, a total of 16 x 20 + 20 x 4 or 400 links were present. In addition, 24 biases were present thus making the total base pairs required for the chromosome to be 424. Direct encoding scheme [50] was used where each base pair had a direct one-to-one mapping to a weight or bias. Using real-valued base pairs, the chromosome was made up of an array of 424 integers.

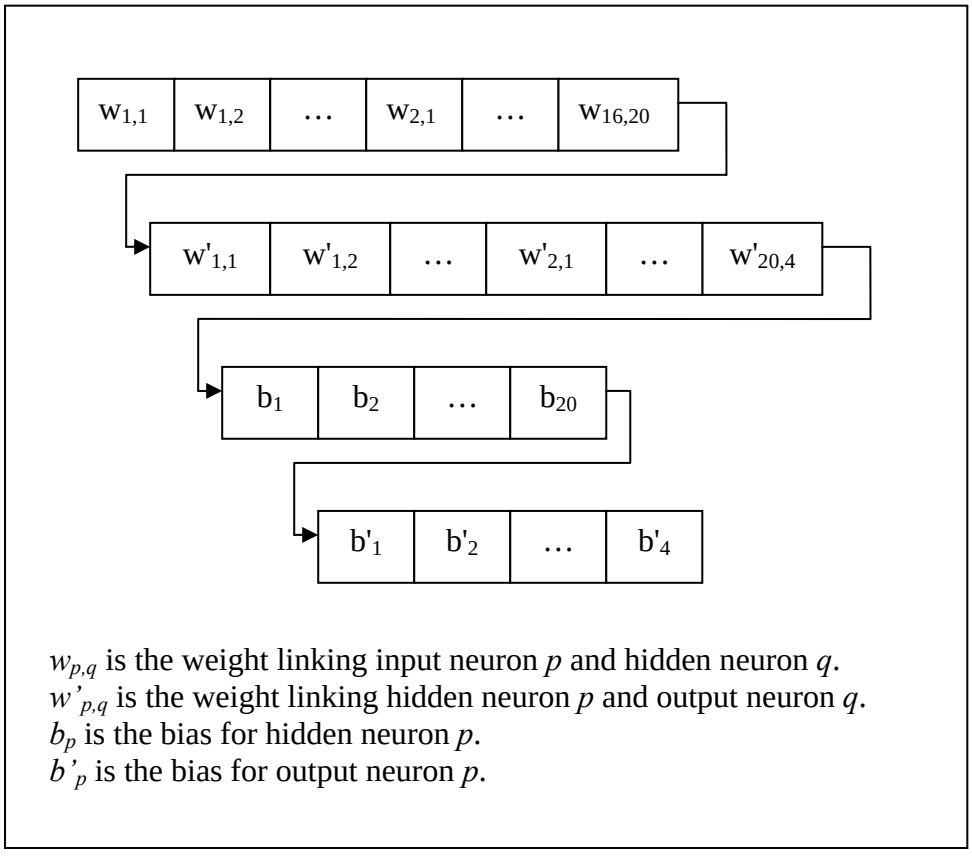


Fig. 5-4 Proposed GA chromosome

Since real-valued base pairs were used, the mutations were executed by adding a random number within –1 to 1 to the original base pairs. This range was decreased linearly with the number of generations that had passed. The probability of mutation was varied from 0.01 to 0.2 through different experiments.

For each experiment, the number of individuals in the population was varied from 10 to 100. Finding the optimum number of individuals required was important due to the large increase in computational resource requirement for each additional individual because of the large chromosome length. The size of the child population was also varied within 70% to 90% of the parent population size. In each generation, the child population was evaluated and reinserted into the parent population based on its fitness.

As the result of a potentially small population chosen, there were times when the fitness level became stagnant after just a few generations. Therefore, in order to avoid this scenario, if the best fitness remains the same after 5 generations, the mutation probability was increased to 100% and the mutation range was increased to the maximum for 1 generation. This step would bring new individuals into the child population which would in turn be reinserted into the parent population if the fitness was found to be better. The algorithm is illustrated in Figure 5-5.

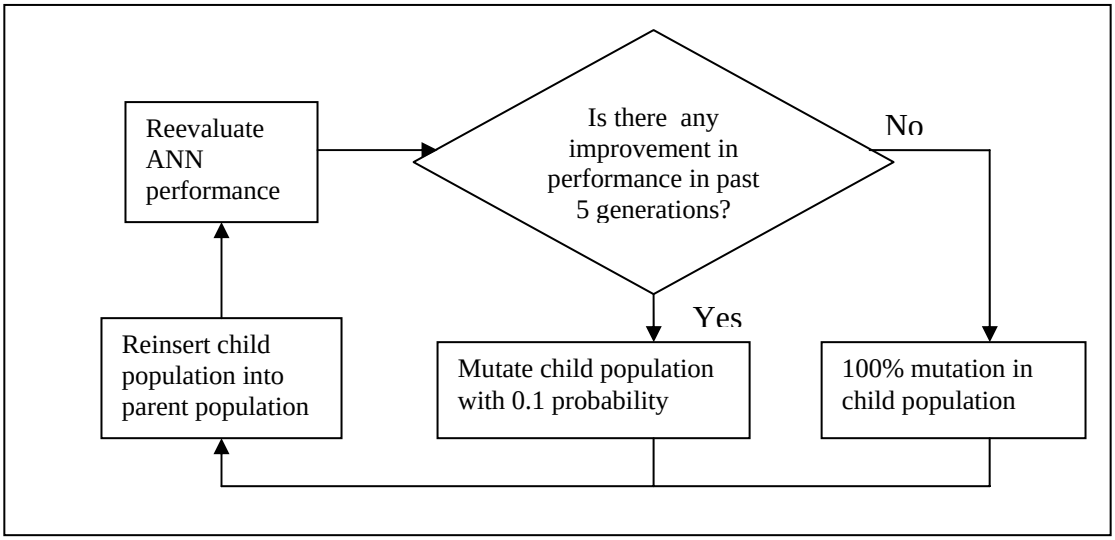


Fig. 5-5 Fixing stagnant MLP performance by introducing fresh individuals

5.5 Results

5.5.1 Evolution of MLP Weights with GA

For the first part of the experiment, the aim was to determine the extent in which the GA would perform in fine-tuning the weights and biases of the MLP. In this section, the values of base pairs from the chromosomes were used to create an MLP with the corresponding weight and bias values. Next, the chromosomes' fitness, tuned only by the GA algorithm, were evaluated based on how the MLP performed on the validation set. Since there was not any backpropagation training of the MLP, the training set was not required.

The GA parameters including population size, child population size and initial mutation probability were varied to fine-tune the GA algorithm. Figures 5-6, 5-7, 5-8, 5-10, 5-11 and 5-12 show the average result for a particular value of the tested

parameter from a variety of other parameters. In other words, for each point on the graph, the value of the tested parameter was fixed while the other parameters were varied and the average result was used. The error referred to the sum of classification errors of the 4 devices.

Figure 5-6 shows that a larger population size would reduce the error, thus increasing the performance of the final MLP constructed. Apparently, the signature identification problem set was plagued with a lot of local minima, hence a need for a larger population size to search through a larger solution set. The main compromise was the larger computational resource requirement.

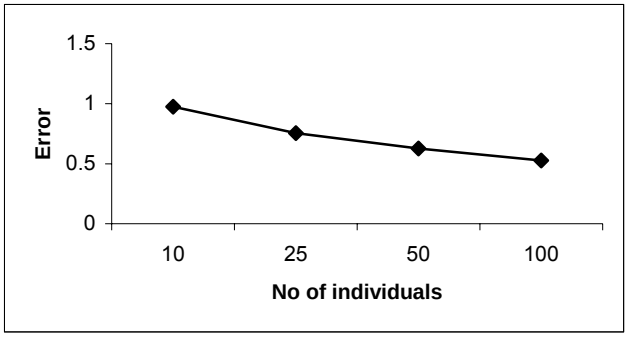


Fig. 5-6 Effect of varying population size on performance

As for the size of child population, Figure 5-7 shows that a larger child population gives only a marginally better MLP performance. On the other hand, Figure 5-8 shows that the optimum mutation probability was around 0.1. A value that is too low will not allow enough mutation to reach the minima while a value that is too high will cause the chromosomes to over-mutate and overlook the minima.

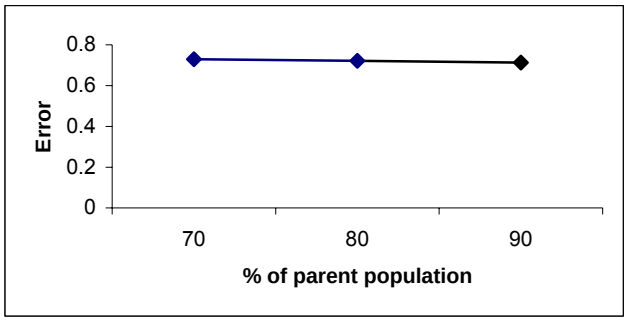


Fig. 5-7 Effect of varying child population size on performance

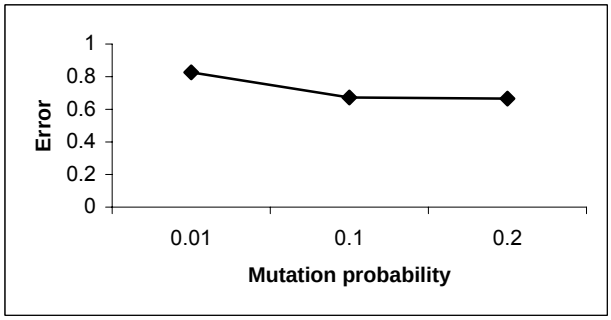


Fig. 5-8 Effect of varying the mutation probability on performance

Figure 5-9 illustrates the convergence of the GA-ANN towards the global minimum. However, even after 50 generations, the minimum error reached, which referred to the average error of the 4 devices, was 0.1006. The error is translated into 0.8994 average accuracy of the classification of the 4 electrical devices. Unfortunately, this value is still much lower than that achievable by a properly tuned MLP that was obtained from a random selection of 1000 initial weights.

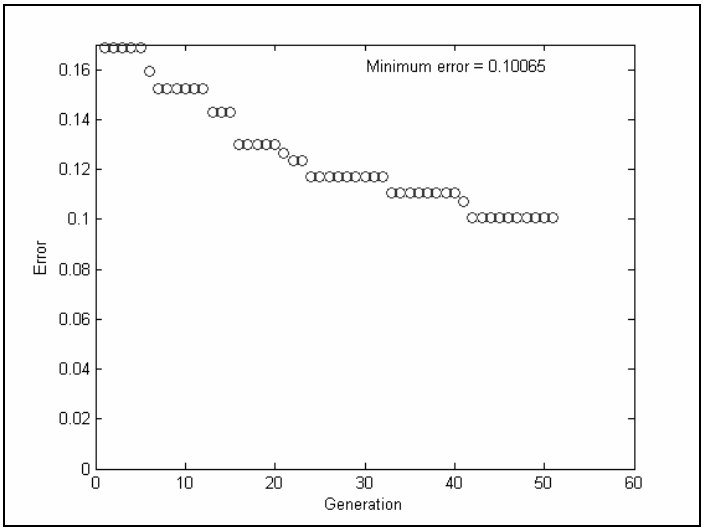


Fig. 5-9 Evolution of the MLP performance

5.5.2 Evolution of MLP Initial Weights Coupled With Backpropagation

Due to the slow convergence of the MLP weights into the best performing MLP by the GA, the GA-ANN combination was modified such that the GA was used to find the best initial weights while backpropagation training was used to converge the MLP weights to the global minimum. From this coupling, the GA could be seen as a search for a potential region for global minimum while the backpropagation training did the local search [50,56].

The MLP was trained using the training set and its performance was evaluated based on the validation set. The stopping criterion was the minimum improvement gradient of the MLP performance on the validation set.

In general, the trend of the MLP performance against the GA parameters is similar to that in section 5.5.1. However, as expected and as shown in Figures 5-10, 5-

11, and 5-12, the general error value is in the scale of 10 times lower. The backpropagation proves to be much more efficient at fine-tuning the weights of the MLP to reach a local minimum in this scenario. The only penalty may be in terms of additional computation time required in executing the large number of backpropagation trainings for each individual in each generation.

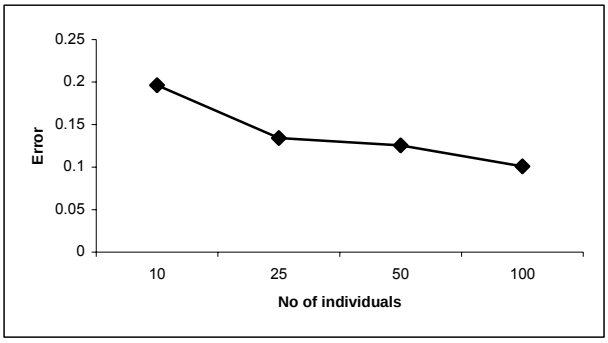


Fig. 5-10 Effect of varying population size on performance

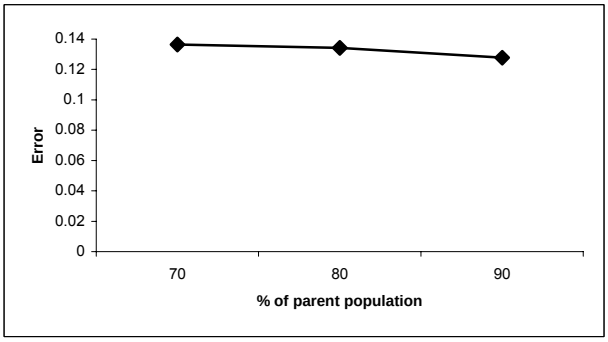


Fig. 5-11 Effect of varying child population size on performance

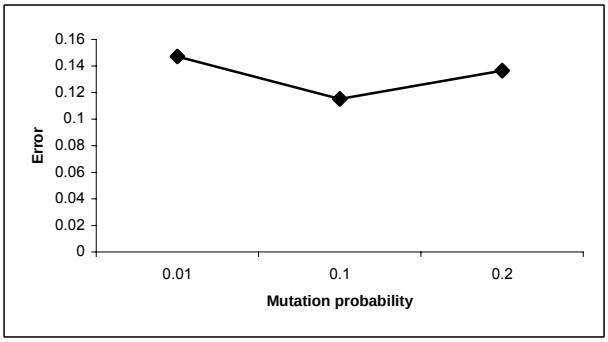


Fig. 5-12 Effect of varying the mutation probability on performance

Figure 5-13 shows the MLP reaching the best classification accuracy of 0.987. This result also outperformed the results from a purely backpropagation optimization alone where 5000 MLP with different random initial weights were trained sequentially. It is deduced that the GA has avoided the local minima and shifted the initial weights to the region of the global minimum.

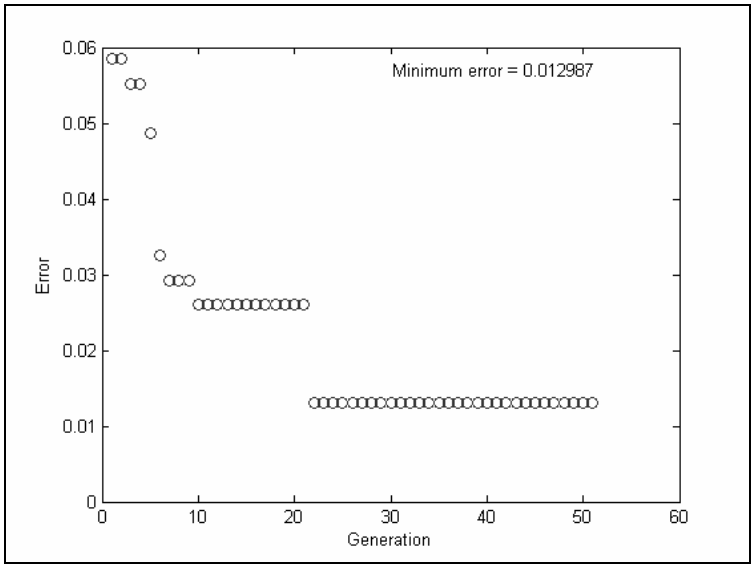


Fig. 5-13 Evolution of the MLP performance

5.6 Performance of GA-ANN Combination

The evolution of the MLP initial weights and biases using GA coupled with local minima search using backpropagation has produced excellent results in the area of signature identification of electrical devices based on their power harmonics. The results based solely on GA optimization suffered from the GA's inability to fully converge to the minima. On the other hand, using backpropagation training alone, the MLP would be susceptible to local minima problems. A classification accuracy of 98.7% is usually considered very high for a signature identification problem.

Chapter 6 Conclusion and Recommendations

The ANN and SVM-based models have been successfully applied in the signature identification of electrical devices based on the current harmonics even under noisy conditions. The harmonics information provides a finer signature resolution compared to the P-Q chart proposed in conventional methods and does not require step change detection. Although transient information, which requires high sampling rate and continuous monitoring, was not used, the harmonics information proved to be sufficient for the signature identification.

From the test results, the conclusion was that the MLP performs well and produces better results or results comparable to that of the RBF neural network and SVM while maintaining the architectural simplicity of the MLP. The RBF neural network and SVM are limited by the size of the training set that can be used practically to avoid huge QP problems. The potential of the time delay neural network was demonstrated but requires further research to reach the required 100% accuracy. Further optimization of the MLP was performed by evolving the MLP weights using GA. For future work, other forms of combination besides the weights evolution can also be used to further optimize the architecture of the MLP.

In the course of the research, this thesis has proposed and implemented a new multi-class SVM for non-mutually exclusive classes and novel application of MLP weights evolution using GA for harmonics signature identification.

Identification of the devices present from the current waveform gives an easier and more reliable access to the information. By nature of an electrical system, most of the electrical wirings are connected to a central location, which is the incoming point from the electrical power supplier. This fact means that a central management system to monitor all the devices from one point can be set up.

The reliability lies in the fact that almost any device, which is functioning properly, shows a normal current waveform. In the case of an electrical fault, the current waveform detected will be abnormal. Therefore, the current waveform signature is a reliable source of information on any abnormalities in the functioning of the devices, without additional sensors or wiring.

Last but not least, this new approach allows us to perform a black box analysis, to determine the possible devices present, from just the electrical wiring that leads to the black box. All devices that draw electrical energy from the external source will leave a corresponding current signature.

Future directions include extending the classification scope to incorporate the various operational modes of each device or operation under different voltage source conditions. From the current approach, training the ANN or SVM to model the various operational modes will require a large training set that may affect performance. Future work should aim towards the search for a better signature modeling technique such as using principal component analysis or SOM.

Currently, the set of electrical devices included in the research work is also still limited. The scope of the types of electrical devices can be further widened. Devices with the same function but different models such as the CPU with different power supply ratings and monitors of different sizes can also be added to the list. More study should be performed on multiples of devices of the same model since it is very likely for a workplace environment to have several such devices.

On the other hand, the approach proposed in this thesis may be applied to special integrated systems where the devices are known and there is a need to continuously monitor the operation of the various components in the system. The size of the system may range from as small as the components in the CPU to as large as a manufacturing plant.

Besides that, research can also be focused on the automatic detection of new devices, marking new regions of correspondence in the feature space, possibly by setting thresholds to signify presence of unknown devices. Adaptive neural networks are potential candidates for this purpose due to its ability to automatically include the new devices without complete restructuring of the ANN architecture.

References

[1] W. Tan and V. I. John, “Nonlinear Fluorescent Systems: Their Impact on Power Quality”, Canadian Conference on Electrical and Computer Engineering, Vol. 1, page 144-147 (1994).

[2] J. Arrillaga, “Power System Harmonic Analysis”, New York: Wiley (1997).

[3] C. W. Smith, Jr., “Power systems and harmonic factors”, IEEE Potentials, Vol. 20, page 10-12 (2001).

[4] W. Mack Grady and S. Santoso, “Understanding Power System Harmonics”, IEEE Power Engineering Review, Vol. 21, page 8-11 (2001).

[5] K. Kuah, M. Bodruzzaman, S. Zein-Sabatto, “A Neural Network-based Text Independent Voice Recognition System”, IEEE Southeastcon ’94 ‘Creative Technology Transfer - A Global Affair’, page 131-135 (1994)

[6] D.G. Childers, Ke Wu, K.S. Bae, D.M. Hicks, “Automatic Recognition of Gender by Voice”, International Conference on Acoustics, Speech, and Signal Processing, Vol. 1, page 603 – 606 (1988)

[7] R.H. Seireg, A.E. Barbour, “A New Algorithm for Pattern Recognition of Voices”, Midwest Symposium on Circuits and Systems, Vol. 1, page 707-710 (1992)

[8] R.K. Hartana and G.G. Richards, “Constrained Neural Network-Based Identification of Harmonic Sources”, IEEE Transactions on Industry Applications, Vol. 29, page 202-208 (1993).

[9] N. Pecharanin, M. Sone and H. Mitsui, “An Application of Neural Network for Harmonic Detection in Active Filter”, IEEE International Conference on Neural Networks, Vol. 6, page 3756-3760 (1994).

[10] P.K. Dash, D.P. Swain, B.R. Mishra and S. Rahman, “Power Quality Assessment Using an Adaptive Neural Network”, International Conference on Power Electronics, Drives and Energy Systems for Industrial Growth, Vol. 2, page 770-775 (1996).

[11] B. Perunicid, M. Mallini, Z. Wang and Y. Liu, “Power Quality Disturbance Detection and Classification Using Wavelets and Artificial Neural Networks”, International Conference on Harmonics and Quality of Power, Vol. 1, page 77-82 (1998).

[12] M. Rukonuzzaman and M. Nakaoka, “Magnitude and Phase Determination of Harmonic Currents by Adaptive Learning Back-Propagation Neural Network”, IEEE 1999 International Conference on Power Electronics and Drive Systems, page 1168-1171 (1999).

[13] S. Santoso, E.J. Powers, W.M. Grady and A.C. Parsons, “Power Quality Disturbance Waveform Recognition Using Wavelet-Based Neural Classifier – Part 1: Theoretical Foundation”, IEEE Transactions on Power Delivery, Vol. 15, page 222-228 (2000).

[14] J.V. Wijayakulasooriya, G.A. Putrus and P.D. Minns, “Electric power quality disturbance classification using self-adapting artificial neural networks”, IEE Proc.- Generation Transmission and Distribution, Vol. 149, page 98-101 (2002).

[15] T. Onoda, H. Murata, G. Ratsch and K.-R. Muller, “Experimental analysis of support vector machines with different kernels based on non-intrusive monitoring data”, 2002 International Joint Conference on Neural Networks, Vol. 3, page 2186-2191 (2002)

[16] S. Poyhonen, M. Negrea, A. Arkkio, H. Hyotyniemi and H. Koivo, “Fault diagnostics of an electrical machine with multiple support vector classifiers”, 2002 IEEE International Symposium on Intelligent Control, page 373-378 (2002)

[17] L.S. Moulin, A.P.A. da Silva, M.A. El-Sharkawi and R.J. Marks, “Support vector and multilayer preceptron neural networks applied to power systems transient stability analysis with input dimensionality reduction”, IEEE Power Engineering Society Summer Meeting, Vol. 3, page 1308-1313 (2002).

[18] M.-C.T. Nguyen, W.J. Lee, “An approach to enhance the harmonic sources identification process”, IEEE Industrial and Commercial Power Systems Technical Conference, page 127-132 (2000).

[19] A.M. Dan, Z. Czira, “Identification of harmonic sources”, International Conference on Harmonics and Quality of Power, Vol. 2, page 831-836 (1998).

[20] G.T. Heydt, “Identification of harmonic sources by a state estimation technique”, IEEE Transactions on Power Delivery, Vol. 4, page 569-576 (1989).

[21] Z.P. Du, J. Arrillaga, N.R. Watson, S. Chen, “Identification of harmonic sources of power systems using state estimation”, IEE Proceedings – Generation, Transmission and Distribution, Vol. 146, page 7-12 (1999).

[22] H. Ma, A.A. Girgis, “Identification and tracking of harmonic sources in a power system using a Kalman filter”, IEEE Transactions on Power Delivery, Vol. 11, page 1659-1665 (1996).

[23] F. Filippetti, G. Franceschini, C. Tassoni, P. Vas, “Recent developments of induction motor drives fault diagnosis using AI techniques”, Vol. 47, page 994-1004 (2000).

[24] G.W. Hart, “Nonintrusive appliance load monitoring”, Proceedings of the IEEE, Vol. 80, page 1870-1891 (1992).

[25] F. Sultanem, “Using appliance signatures for monitoring residential loads at meter panel level”, IEEE Transactions on Power Delivery, Vol. 6, page 1380-1385 (1991)

[26] J.G. Roos, I.E. Lane, E.C. Botha, G.P. Hancke, “Using neural networks for non-intrusive monitoring of industrial electrical loads”, IEEE Instrumentation and Measurement Technology Conference, Vol. 3, page 1115-1118 (1994).

[27] T. Lindblad, S. Hultberg, C.S. Lindsey, R.O. Shelton, “Performance of a neural network for recognizing AC current demand signatures in the space shuttle telemetry data”, American Control Conference, Vol. 2, page 1373-1377 (1995).

[28] W.L. Chan, A.T.P. So, L.L. Lai, “Harmonics load signature recognition by wavelets transforms”, International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, page 666-671 (2000).

[29] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford, P. Armstrong, “Power signature analysis”, IEEE Power and Energy Magazine, Vol. 1, page 56-63 (2003).

[30] W.L. Chan, A.T.P. So, “Fuzzy arithmetic based power harmonics signature recognition”, International Conference on Advances in Power Systems Control, Operation and Management, Vol. 1, page 404-409 (1993).

[31] D. Raisz, M. Sakulin, H. Renner, Y. Tehlivets, “Recognition of the operational states in electric arc furnaces”, International Conference on Harmonics and Quality of Power, Vol. 2, page 475-480 (2000).

[32] D. Alexandrou, D. Pantzartzis, “A Methodology for Acoustic Seafloor Classification”, IEEE Journal of Oceanic Engineering, Vol. 18, page 81 – 86 (1993)

[33] M. Riedmiller, H. Braun, “A direct adaptive method for faster backpropagation learning: the RPROP algorithm”, IEEE International Conference on Neural Networks, Vol. 1, page 586-591 (1993).

[34] Del Boca, D.C. Park, “Myoelectric Signal Recognition using Fuzzy Clustering and Artificial Neural Networks in Real Time”, IEEE International Conference on Neural Networks, Vol. 5, page 3098 – 4103 (1994)

[35] Pacut, A. Czajka, “Recognition of Human Signatures”, International Joint Conference on Neural Networks, Vol. 2, page 1560 – 1564 (2001)

[36] S.S. Warren, “Neural Network FAQ”, ftp://ftp.sas.com/pub/neural/FAQ.html (2001).

[37] Kevin M. Coggins, Jose Principe, “Detection and Classification of Insect Sounds in a Grain Silo using a Neural Network”, IEEE International Joint Conference on Neural Networks, Vol. 3, page 1760 – 1765 (1998)

[38] Chih-Wei Hsu, Chih-Jen Lin, “A comparison of methods for multiclass support vector machines”, IEEE Transactions on Neural Networks, Vol. 13, pp 415-425, 2002.

[39] J.A.K. Suykens, J. Vandewalle, “Multiclass least squares support vector machines”, International Joint Conference on Neural Networks, Vol. 2, pp. 900-903, 1999.

[40] F. Masulli, G. Valentini, “Comparing decomposition methods for classification”, Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies, Vol. 2, pp. 788-791, 2000.

[41] V. Franc, V. Hlavac, “Multi-class support vector machine”, International Conference on Pattern Recognition, Vol. 2, pp. 236-239, 2002.

[42] Hong-Jie Xing, Xi-Zhao Wang, Qiang He, Hong-Wei Yang, “The multistage support vector machine”, International Conference on Machine Learning and Cybernetics, Vol. 4, pp. 1815-1818, 2002.

[43] U. Kressel, “Pairwise classification and Support Vector Machines”, Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge, MA, 1998.

[44] F. Schwenker, “Hierarchical support vector machines for multi-class pattern recognition”, Knowledge-Based Intelligent Engineering Systems and Allied Technologies, Vol. 2, pp. 561-565, 2000.

[45] J.C. Platt, N. Cristianini and J. Shawe-Taylor, “Large margin DAGs for multiclass classification”, Neural Information Processing Systems, Vol. 12, pp. 547-553, 2000.

[46] F. Takahashi, S. Abe, “Decision-tree-based multiclass support vector machines”, Proceedings of the 9th InternationalConference on Neural Information Processing, Vol. 3, pp. 1418-1422, 2002.

[47] B. Kijsirikul, N. Ussivakul, “Multiclass support vector machines using adaptive directed acyclic graph ”, International Joing Conference on Neural Networks, Vol. 1, pp. 980-985, 2002.

[48] Zeyu Li, Shiwei Tang, Jing Xue, “A novel SVM multi-class classifier based on pairwise coupling”, IEEE International Conference on Systems, Man and Cybernetics, Vol. 7, pp. 5, 2002.

[49] Jae-Jin Kim, Bon-Woo Hwang, Seong-Whan Lee, “Retrieval of the top N matches with support vector machines”, International Conference on Pattern Recognition, Vol. 2, pp. 716-719, 2000.

[50] Xin Yao, “Evolving artificial neural networks”, Proceedings of the IEEE, Vol. 87, page 1423-1447 (1999)

[51] D.B. Fogel, “What is evolutionary computation?”, IEEE Spectrum, Vol. 37, page 26, 28-32 (2000)

[52] K.J. Hintz and J.J. Spofford, “Evolving a neural network”, 5th IEEE International Symposium on Intelligent Control, Vol. 1, page 479-484 (1990)

[53] X. Yao and Y. Liu, “A new evolutionary system for evolving artificial neural networks”, IEEE Transactions on Neural Networks, Vol. 8, page 694-713 (1997)

[54] B. Carse, A.G. Pipe, T.C. Forgarty and T. Hill, “Evolving radial basis function neural networks using a genetic algorithm”, IEEE International Conference on Evolutionary Computation, Vol. 1, page 300 (1995)

[55] S. Yao, C. Wei and Z. He, “Evolving wavelet neural networks”, IEEE International Conference on Neural Networks, Vol. 4, page 1851-1854 (1995)

[56] R.K. Belew, J. McInerney, and N.N. Schraudolph, “Evolving networks: Using the genetic algorithm with connectionist learning”, Comput. Sci. Eng. Dep. (C-014), Univ. of California, San Diego, Tech. Rep. CS90-174 (revised) (1991).

[57] T. Joachims, “Making large-scale SVM learning practical. Advances in Kernel Methods – Support Vector Learning”, B. Scholkopf and C. Burges and A. Smola, MIT-Press (1999).

Appendix A Feature Vector Sets

10-devices set A

This set of devices was used in the experiments in Section 4.3.

Table A-1a 10-devices set A name list

No	Name
1	Monitor
2	CPU
3	Fluorescent lamp
4	Television
5	Soldering iron
6	Fridge
7	Fan
8	Battery charger
9	Light bulb
10	Power drill

Table A-1b 10-devices set A feature vectors

	1	2	3	4	5	6	7	8	9	10
x_1	0.265	0.198	0.410	0.183	0.251	0.200	0.166	0.134	0.557	0.600
x_2	0.085	0.098	-0.143	0.059	-0.046	0.113	0.072	-0.119	0.059	-0.038
x_3	-0.230	-0.169	0.044	-0.158	0.012	-0.155	0.007	0.034	0.024	-0.022
x_4	0.069	0.047	-0.063	0.032	-0.036	-0.096	0.028	0.019	0.007	0.030
x_5	0.159	0.113	0.007	0.108	-0.002	0.065	-0.008	-0.038	0.011	0.014
x_6	-0.115	-0.090	-0.017	-0.061	-0.020	0.097	0.005	0.017	0.006	-0.016
x_7	-0.086	-0.056	0.002	-0.060	-0.003	0.016	-0.001	-0.010	0.012	0.019
x_8	0.112	0.088	0.028	0.060	-0.013	-0.068	0.001	-0.021	0.007	-0.005
x_9	0.028	0.011	0.000	0.022	-0.012	-0.065	-0.005	0.002	-0.001	0.003
x_{10}	-0.082	-0.063	-0.045	-0.047	-0.003	0.014	0.003	0.002	0.008	0.003
x_{11}	-0.002	0.009	0.017	-0.004	-0.010	0.066	-0.003	-0.005	0.003	0.006
x_{12}	0.040	0.027	0.005	0.022	0.000	0.031	0.000	-0.001	0.006	-0.005
x_{13}	0.000	-0.008	-0.018	-0.001	-0.005	-0.032	-0.001	0.004	0.002	0.008
x_{14}	-0.006	0.002	-0.022	0.000	0.004	-0.054	0.001	-0.002	0.007	-0.001
x_{15}	-0.010	-0.004	-0.032	-0.001	-0.003	-0.014	-0.001	0.001	-0.002	0.005
x_{16}	-0.013	-0.019	-0.042	-0.015	0.007	0.042	0.000	0.004	0.006	0.000

10-devices set B

This set of devices was used in the experiments in Section 4.1, 4.3 and 4.5.

Table A-2a 10-devices set B name list

No	Name
1	CPU
2	Monitor
3	CPU (Shutdown)
4	DC Power Supply (0.1A)
5	DC Power Supply (0.5A)
6	DC Power Supply (0.25A)
7	DC Power Supply (0.4A)
8	Notebook Computer
9	Mobile phone charger
10	Flourescent lamp

Table A-2b 10-devices set B feature vectors

	1	2	3	4	5	6	7	8	9	10
x_1	0.238	0.254	0.134	0.208	0.274	0.232	0.254	0.099	0.029	0.266
x_2	0.127	0.092	0.105	-0.120	-0.122	-0.118	-0.118	0.099	0.006	-0.087
x_3	-0.209	-0.209	-0.119	-0.042	-0.088	-0.057	-0.075	-0.109	-0.040	0.038
x_4	-0.022	0.068	0.013	-0.056	-0.020	-0.040	-0.029	0.015	-0.004	-0.032
x_5	0.150	0.130	0.091	0.008	0.027	0.012	0.022	0.093	0.028	0.004
x_6	0.001	-0.109	-0.042	-0.029	-0.064	-0.038	-0.056	-0.037	0.011	-0.020
x_7	-0.080	-0.054	-0.054	-0.010	-0.007	-0.007	-0.005	-0.065	-0.016	-0.008
x_8	-0.007	0.107	0.044	0.017	0.039	0.019	0.030	0.047	-0.011	0.006
x_9	0.027	0.001	0.023	-0.019	-0.030	-0.020	-0.020	0.040	0.005	-0.006
x_{10}	0.014	-0.070	0.033	0.007	0.000	0.001	0.000	-0.045	0.009	-0.019
x_{11}	0.011	0.014	0.002	0.020	0.029	0.030	0.029	-0.024	0.000	0.000
x_{12}	-0.017	0.014	0.010	-0.002	-0.009	-0.005	-0.006	0.032	0.000	0.000
x_{13}	-0.033	-0.010	-0.005	0.000	0.000	-0.006	-0.003	0.014	0.000	-0.006
x_{14}	0.022	0.017	0.009	0.000	0.010	0.008	0.009	-0.014	0.000	0.008
x_{15}	0.035	-0.010	0.003	-0.002	-0.007	-0.005	-0.005	-0.010	0.010	0.000
x_{16}	-0.020	-0.028	-0.020	-0.020	-0.007	-0.009	-0.009	-0.001	0.003	0.000

Table A-3 Combinations of devices

No.	Devices									
	1	2	3	4	5	6	7	8	9	10
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	-1	-1	-1	-1	-1	1
3	-1	-1	-1	-1	-1	-1	-1	-1	1	-1
….										
1023	1	1	1	1	1	1	1	1	1	-1
1024	1	1	1	1	1	1	1	1	1	1

Three phase devices set

This set of devices was used in the experiments in Section 4.4.

Table A-4a Three phase devices set name list

No	Name
1	Motor #1
2	Motor #2
3	Motor #2 with capacitors
4	Inverter #1
5	Inverter #2 (low frequency)
6	Inverter #2 (High frequency)
7	Fluorescent lamp without capacitors
8	Fluorescent lamp with capacitors
9	Motor #1
10	Motor #2

Table A-4b Three phase devices set feature vectors

	1	2	3	4	5	6	7	8	9	10
x_1	0.656	0.386	0.495	0.600	0.387	0.196	2.074	0.279	0.656	0.386
x_2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
x_3	0.009	0.173	0.316	0.348	0.019	-0.020	0.060	0.029	0.009	0.173
x_4	-0.006	-0.180	-0.073	-0.140	-0.029	0.011	0.047	0.071	-0.006	-0.180
x_5	0.000	0.027	0.396	0.400	-0.003	0.011	-0.068	0.052	0.000	0.027
x_6	0.011	-0.389	-0.229	-0.400	0.007	-0.006	-0.017	-0.093	0.011	-0.389
x_7	-0.011	-0.025	0.379	0.404	0.002	0.027	0.043	0.006	-0.011	-0.025
x_8	-0.005	-0.362	-0.138	-0.293	-0.004	-0.009	-0.012	0.170	-0.005	-0.362
x_9	0.000	-0.184	0.205	0.109	-0.001	0.024	0.002	-0.006	0.000	-0.184
x_{10}	0.000	-0.144	-0.198	-0.299	0.001	-0.004	0.002	-0.019	0.000	-0.144
x_{11}	0.000	-0.368	0.239	0.035	0.001	0.000	-0.011	-0.031	0.000	-0.368
x_{12}	0.000	-0.019	-0.328	-0.503	-0.001	-0.014	-0.002	-0.064	0.000	-0.019
x_{13}	0.000	-0.317	0.248	0.100	-0.001	-0.004	0.002	-0.028	0.000	-0.317
x_{14}	0.000	0.028	-0.231	-0.402	0.000	-0.007	-0.001	0.038	0.000	0.028
x_{15}	0.000	-0.112	0.069	-0.125	0.000	-0.010	0.000	-0.027	0.000	-0.112
x_{16}	0.000	0.172	-0.241	-0.245	0.000	-0.009	0.000	0.002	0.000	0.172
x_{17}	0.657	0.172	0.179	0.233	0.385	0.196	1.971	0.440	0.657	0.172
x_{18}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
x_{19}	-0.006	0.010	0.020	0.003	0.021	-0.023	-0.032	-0.026	-0.006	0.010
x_{20}	0.003	0.021	0.039	0.023	-0.026	0.012	-0.023	-0.017	0.003	0.021
x_{21}	0.002	-0.120	-0.099	-0.208	-0.003	0.014	-0.076	-0.070	0.002	-0.120
x_{22}	0.013	0.116	0.147	0.106	0.007	-0.006	-0.003	-0.126	0.013	0.116
x_{23}	-0.012	-0.103	-0.080	-0.192	0.002	0.024	0.028	0.143	-0.012	-0.103
x_{24}	0.001	0.131	0.151	0.115	-0.003	-0.004	-0.021	0.010	0.001	0.131
x_{25}	0.000	-0.021	-0.040	-0.009	-0.001	0.021	-0.002	0.020	0.000	-0.021
x_{26}	0.000	-0.003	-0.007	-0.023	0.001	0.004	-0.001	0.008	0.000	-0.003
x_{27}	0.001	-0.006	-0.075	0.122	0.001	0.006	-0.013	-0.044	0.001	-0.006
x_{28}	0.000	-0.158	-0.155	-0.189	-0.001	-0.012	0.004	0.071	0.000	-0.158
x_{29}	0.000	-0.030	-0.085	0.100	-0.001	0.008	0.004	0.043	0.000	-0.030
x_{30}	0.000	-0.156	-0.136	-0.181	0.001	-0.008	-0.001	0.035	0.000	-0.156
x_{31}	0.000	0.012	0.028	0.014	0.000	-0.015	0.000	-0.023	0.000	0.012

x_{32}	0.000	-0.012	-0.026	0.020	0.000	-0.021	0.000	-0.035	0.000	-0.012
x_{33}	0.689	0.406	0.551	0.648	0.388	0.199	2.154	0.445	0.689	0.406
x_{34}	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
x_{35}	-0.001	0.199	0.216	0.269	0.021	-0.023	-0.035	0.006	-0.001	0.199
x_{36}	0.005	0.106	0.195	0.226	-0.028	0.013	-0.012	-0.045	0.005	0.106
x_{37}	-0.001	0.170	0.016	0.056	-0.003	0.006	-0.068	0.123	-0.001	0.170
x_{38}	0.011	0.348	0.446	0.535	0.007	-0.007	0.000	0.011	0.011	0.348
x_{39}	-0.010	0.223	-0.025	0.030	0.003	0.020	0.033	-0.153	-0.010	0.223
x_{40}	-0.003	0.330	0.473	0.564	-0.004	-0.004	-0.020	0.038	-0.003	0.330
x_{41}	0.000	0.029	-0.129	-0.128	-0.001	0.025	0.000	-0.003	0.000	0.029
x_{42}	0.000	0.210	0.215	0.274	0.001	-0.004	0.000	0.021	0.000	0.210
x_{43}	0.001	-0.188	-0.362	-0.434	0.001	0.004	-0.010	0.065	0.001	-0.188
x_{44}	0.001	0.290	-0.006	0.061	-0.001	-0.011	0.003	0.010	0.001	0.290
x_{45}	0.001	-0.156	-0.431	-0.507	0.000	0.002	0.004	0.031	0.001	-0.156
x_{46}	0.001	0.334	-0.030	0.053	0.000	-0.008	-0.004	-0.089	0.001	0.334
x_{47}	0.000	-0.131	-0.218	-0.265	0.000	-0.015	0.000	0.015	0.000	-0.131
x_{48}	0.000	0.131	-0.088	-0.056	0.000	-0.005	-0.001	0.035	0.000	0.131

4-devices set

This set of devices was used in the experiments in Section 4.6 and Chapter 5

Table A-5a 4-devices set name list

No	Name
1	Fluorescent lamp
2	CPU
3	Monitor
4	Television

Table A-5b 4-devices set feature vectors

	1	2	3	4
x_1	0.437	0.179	0.278	0.191
x_2	-0.194	0.112	0.096	0.097
x_3	0.043	-0.150	-0.240	-0.172
x_4	-0.074	0.055	0.064	0.030
x_5	0.009	0.096	0.168	0.125
x_6	-0.013	-0.100	-0.114	-0.064
x_7	0.002	-0.039	-0.091	-0.073
x_8	0.043	0.107	0.117	0.068
x_9	0.009	-0.004	0.032	0.031
x_{10}	-0.048	-0.082	-0.088	-0.050
x_{11}	0.026	0.027	0.000	-0.002
x_{12}	0.008	0.040	0.045	0.024
x_{13}	-0.017	-0.026	-0.006	-0.008
x_{14}	-0.026	-0.007	-0.010	0.000
x_{15}	-0.022	0.007	-0.004	0.003
x_{16}	-0.060	-0.017	-0.012	-0.017

Segment of 8-devices Setup's Database

Table A-6 8-devices setup database structure

Inputs																Device							
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	1	2	3	4	5	6	7	8
0.009	-0.017	0.000	-0.005	-0.002	0.005	0.009	-0.008	-0.002	0.002	-0.006	-0.002	0.007	0.001	0.003	0.001	-1	-1	-1	-1	-1	-1	-1	-1
0.005	-0.015	0.001	-0.002	0.001	0.004	0.010	-0.006	-0.003	-0.001	-0.004	-0.005	0.012	0.000	0.000	-0.006	-1	-1	-1	-1	-1	-1	-1	-1
									….														
0.472	0.050	0.027	0.008	0.009	0.005	0.011	0.006	0.002	0.008	0.001	0.006	0.004	0.006	-0.002	0.009	-1	-1	-1	-1	-1	-1	-1	1
0.473	0.050	0.027	0.006	0.009	0.004	0.010	0.007	0.003	0.008	0.002	0.005	0.004	0.006	-0.001	0.009	-1	-1	-1	-1	-1	-1	-1	1
									….														
0.233	0.094	-0.164	-0.102	0.055	0.109	0.034	-0.064	-0.077	0.007	0.060	0.042	-0.013	-0.055	-0.029	0.035	-1	-1	-1	-1	-1	-1	1	-1
0.233	0.094	-0.164	-0.102	0.055	0.109	0.034	-0.064	-0.077	0.007	0.060	0.042	-0.013	-0.055	-0.029	0.035	-1	-1	-1	-1	-1	-1	1	-1
0.678	0.095	-0.108	-0.076	0.062	0.100	0.033	-0.057	-0.061	0.020	0.056	0.040	-0.017	-0.047	-0.024	0.048	-1	-1	-1	-1	-1	-1	1	1
0.679	0.095	-0.109	-0.074	0.063	0.100	0.031	-0.058	-0.059	0.022	0.056	0.038	-0.018	-0.046	-0.023	0.046	-1	-1	-1	-1	-1	-1	1	1
									….														
1.895	-0.132	-0.560	0.079	0.438	-0.177	-0.214	0.186	0.067	-0.207	0.041	0.114	-0.042	-0.079	-0.013	-0.033	1	1	1	1	1	1	1	1
1.898	-0.133	-0.558	0.078	0.434	-0.175	-0.213	0.185	0.067	-0.206	0.041	0.113	-0.039	-0.077	-0.016	-0.035	1	1	1	1	1	1	1	1

Appendix B MATLAB Codes

Training and Testing of MLP

```
% Training and testing of the MLP ANN with different number of hidden
% neurons and number of inputs.

% Training set -      mathematical combinations using basis.txt which
%                     contains the feature vectors of individual devices

% Validation -       2/3 of dataset.txt which contains the laboratory
% (Early             measurements of the input vectors of all combinations of
%  stopping)         devices

% Testing - remaining 1/3 of dataset.txt

% Loading of the input vectors from dataset.txt.
load dataset.txt;
[datasetR datasetC] = size(dataset);

% Splitting of the dataset.txt input vectors into validation and test sets.
validation = zeros(datasetR*2/3,datasetC);
test = zeros(datasetR*1/3,datasetC);
j = 1;
k = 1;
for i = 1:datasetR
    if mod(i,3) == 0
        test(j,:) = dataset(i,:);
        j = j+1;
    else
        validation(k,:) = dataset(i,:);
        k = k+1;
    end
end

% Loading of the feature vectors from basis.txt.
load basis.txt;

% Creation of the training set by mathematically summing various feature
% vectors from the basis.txt for the various combinations of devices.
training = zeros(256,datasetC);
i = 1;
for dev1 = 0:1
    for dev2 = 0:1
        for dev3 = 0:1
            for dev4 = 0:1
                for dev5 = 0:1
                    for dev6 = 0:1
                        for dev7 = 0:1
                            for dev8 = 0:1
                                array = zeros(1,24);
                                if dev1
                                    array(1,:)=array(1,:)+basis(1,:);
                                end
                                if dev2
```

```
                                    array(1,:)=array(1,:)+basis(2,:);
                                end
                                if dev3
                                    array(1,:)=array(1,:)+basis(3,:);
                                end
                                if dev4
                                    array(1,:)=array(1,:)+basis(4,:);
                                end
                                if dev5
                                    array(1,:)=array(1,:)+basis(5,:);
                                end
                                if dev6
                                    array(1,:)=array(1,:)+basis(6,:);
                                end
                                if dev7
                                    array(1,:)=array(1,:)+basis(7,:);
                                end
                                if dev8
                                    array(1,:)=array(1,:)+basis(8,:);
                                end
                                training(i,:) = array(1,:);
                                i = i+1;
                            end
                        end
                    end
                end
            end
        end
    end
end

% Setting up of the MLP ANN with varying number of input neurons and hidden
% neurons.

% Varying the number of input neurons from 2 to 16 in steps of 2.
for inputs = 1:8

    % Setting up of the training set.
    p = training(:,1:inputs*2)';
    t = training(:,17:24)';
    t = t*2-1;

    % Setting up of the validation set.
    ptest = validation(:,1:inputs*2)';
    ttest = validation(:,17:24)';
    ttest = ttest*2-1;

    % Normalization of training and validation sets.
    [pn, meanp, stdp, tn, meant, stdt] = prestd(p,t);
    pntest = trastd(ptest, meanp, stdp);
    tntest = trastd(ttest, meant, stdt);
    val.P = pntest;
    val.T = tntest;

    % Varying the number of hidden neurons from 4 to 60.
    for nodes = 1:57

        bestaccuracy = 0;

        % Repeating the training using different initial weights and
```

```
% finding the best MLP architecture.
for iteration = 1:100

    % Creation of the MLP ANN.
    % Hidden and output layer neurons uses the tangential sigmoidal
    % activation function.  The training is performed using
    % resilient backpropagation algorithm.
    net = newff(minmax(pn), [nodes+3,8], {'tansig', 'tansig'}, 'trainrp');
    net.trainParam.show = 25;
    net.trainParam.epochs = 3000;
    net = init(net);

    % Initiation of the MLP training with early stopping based on
    % the validation set.
    [net,tr]=train(net,pn,tn,[],[],val);

    % Simulation of the trained MLP on validation set input
    % vectors.
    results = sim(net, pntest);
    results = poststd(results, meant, stdt);
    results = sign(results);

    % Calculation of the error.
    errors = ttest - results;
    [m,n] = size(errors);

    sizepresent = sum((ttest'+1)/2);        % No of input vectors with output = 1
    sizeabsent = n-sizepresent;             % No of input vectors with output = -1
    mispredictpresent = sum((errors > 0)'); % No of input vectors wrongly classified
                                            % with output = -1
    mispredictabsent = sum((errors < 0)');  % No of input vectors wrongly classified
                                            % with output = 1

    % Classification accuracy of each class.
    accuracy(iteration,:) = 1 - (mispredictpresent./sizepresent)*0.5 -
(mispredictabsent./sizeabsent)*0.5;

    % Updating and saving of the MLP in the current iteration
    % if it is better than previous best MLP.
    if sum(accuracy(iteration,:)) > bestaccuracy
        bestaccuracy = sum(accuracy(iteration,:));
        bestnet = net;
    end
end

% Setting up of the test set.
ptest = test(:,1:inputs*2)';
ttest = test(:,17:24)';
ttest = ttest*2-1;

% Normalization of the test set.
pntest = trastd(ptest, meanp, stdp);
tntest = trastd(ttest, meant, stdt);

% Simulation of the best trained MLP on the test set.
results = sim(bestnet, pntest);
results = poststd(results, meant, stdt);
results = sign(results);

% Calculation of the classification accuracy on the test set.
```

```
        errors = ttest - results;
        [m,n] = size(errors);

        sizepresent = sum((ttest'+1)/2);
        sizeabsent = n-sizepresent;
        mispredictpresent = sum((errors > 0)');
        mispredictabsent = sum((errors < 0)');

        accuracy = 1 - (mispredictpresent./sizepresent)*0.5 -
(mispredictabsent./sizeabsent)*0.5;

        % Updating of the classification results for the current number
        % of input and hidden neurons.
        stats{inputs,nodes} = [mispredictpresent' sizepresent' mispredictpresent'./sizepresent'
mispredictabsent' sizeabsent' mispredictabsent'./sizeabsent' accuracy'];

        % Saving of the best MLP architecture for the current number of
        % input and hidden neurons.
        bestnetwork{inputs,nodes} = bestnet;
    end
end
```

Training and Testing of RBF Neural Network

```
% Training and testing of the RBF ANN with different number of hidden
% nodes.

% Training set -    mathematical combinations using basis.txt which
%                   contains the feature vectors of individual devices

% Testing - 1/3 of dataset.txt which contains the laboratory measurements
%           of the input vectors of all combinations of devices

% Loading of the input vectors from dataset.txt.
load dataset.txt;
[datasetR datasetC] = size(dataset);

% Creation of the test set from 1/3 of the input vectors in dataset.txt.
test = zeros(datasetR*1/3,datasetC);
j = 1;
for i = 1:datasetR
    if mod(i,3) == 0
        test(j,:) = dataset(i,:);
        j = j+1;
    end
end

% Loading of the feature vectors from basis.txt.
load basis.txt;

% Creation of the training set by mathematically summing various feature
% vectors from the basis.txt for the various combinations of devices.
training = zeros(256,datasetC);
i = 1;
for dev1 = 0:1
    for dev2 = 0:1
```

```
            ...

    end
end

% Setting up of the training set.
p = training(:,1:16)';
t = training(:,17:24)';
t = t*2-1;

% Setting up of the test set.
ptest = test(:,1:16)';
ttest = test(:,17:24)';
ttest = ttest*2-1;

% Normalization of training and validation sets.
[pn, meanp, stdp, tn, meant, stdt] = prestd(p,t);
pntest = trastd(ptest, meanp, stdp);
tntest = trastd(ttest, meant, stdt);
val.P = pntest;
val.T = tntest;

bestaccuracy = 0;

% Varying the number of centers from 25 to 225 in steps of 25.
for iteration = 1:9
    net = newrb(pn,tn,0,1,iteration*25,25);

    % Simulation of the trained MLP on validation set input
    % vectors.
    results = sim(net, pntest);
    results = poststd(results, meant, stdt);
    results = sign(results);

    % Calculation of the classification accuracy.
    errors = ttest - results;
    m,n] = size(errors);

    sizepresent = sum((ttest'+1)/2);        % No of input vectors with output = 1
    sizeabsent = n-sizepresent;             % No of input vectors with output = -1
    mispredictpresent = sum((errors > 0)');         % No of input vectors wrongly classified with
                                                    % output = -1
    mispredictabsent = sum((errors < 0)');          % No of input vectors wrongly classified with
                                                    % output = 1

    % Classification accuracy of each class in this iteration.
    accuracy(iteration,:) = 1 - (mispredictpresent./sizepresent)*0.5 -
(mispredictabsent./sizeabsent)*0.5
end
```

Evolution of MLP Weights

```
% Evolution of MLP ANN weights using GA.

% Note:     This matlab program requires the GATBX toolbox from
%           http://www.shef.ac.uk/~gaipp/ga-toolbox/

inputs = 16;      % No of input neurons
nodes = 20;       % No of hidden neurons
outputs = 4;      % No of output neurons

% Loading of train.txt which contains the training set.
load train.txt;
p = [train(:,1:inputs)]';
t = train(:,17:20)';
t = t*2-1;

% Loading of validation.txt which contains the validation set.
load validation.txt;
ptest = [validation(:,1:inputs)]';
ttest = validation(:,17:20)';
ttest = ttest*2-1;

% Normalization of training and testing sets.
[pn, meanp, stdp, tn, meant, stdt] = prestd(p,t);
pntest = trastd(ptest, meanp, stdp);
tntest = trastd(ttest, meant, stdt);
val.P = pntest;
val.T = tntest;

% GA Parameters:
MAXGEN = 50;              % maximum Number of generations
NIND = [10 25 50 100];    % Number of individuals per subpopulations
GGAP = [0.7 0.8 0.9];     % Generation gap, how many new individuals are created
MUTPROB = [0.01 0.1 0.2]; % Mutation probability

max_nind = size(NIND,2);
max_ggap = size(GGAP,2);
max_mutprob = size(MUTPROB,2);

% Study of the effects of the various GA parameters.
for nind_counter = 1:max_nind
    for ggap_counter = 1:max_ggap
        for mutprob_counter = 1:max_mutprob

            % Mutation probability initialisation.
            mut_prob = MUTPROB(mutprob_counter);
            mut_shrink = 0;       % Defines reduction in mutation range.

            % Field descriptor which specifies the allowed range of initial
            % weights in the ANN.  It has the same dimension as that of a GA
            % chromosome.
            FieldD = rep([-1;1],[1,inputs*nodes+nodes*outputs+nodes+outputs]);

            % Creation of the initial population
            Chrom = crtrp(NIND(nind_counter), FieldD);

            % Reset counters
            Best = NaN*ones(MAXGEN,1);      % best fitness in current population
            gen = 0;                        % generation counter
```

```
% Evaluation of initial population
ObjV = ones(NIND(nind_counter),1);          % Accuracy matrix of the population

% Creation of neural network.
net = newff(minmax(pn), [nodes,4], {'tansig', 'tansig'}, 'trainrp');
net.trainParam.show = 25;
net.trainParam.epochs = 300;
net = init(net);

% Decoding of the chromosomes into ANN weights and biases.
% Iterate through all the chromosomes in the population.
for iteration = 1:NIND(nind_counter)

    % Pointer in chromosome string initialisation.
    offset = 1;

    % Decoding of weights of connections from inputs to
    % hidden layer neurons.
    for i = 0:(inputs-1)
        net.iw{1}(:,i+1) = Chrom(iteration,(offset+i*nodes):(offset+((i+1)*nodes-
1)))';
    end
    offset = offset+nodes*inputs;

    % Decoding of weights of connections from the hidden
    % layer neurons to the output layer neurons.
    for i = 0:(outputs-1)
        net.lw{2,1}(i+1,:) = Chrom(iteration,(offset+i*nodes):(offset+((i+1)*nodes-
1)));
    end
    offset = offset+nodes*outputs;

    % Decoding of the biases of the hidden layer neurons.
    net.b{1} = Chrom(iteration,offset:offset+nodes-1)';
    offset = offset+nodes;

    % Decoding of the biases of the output layer neurons.
    net.b{2} = Chrom(iteration,offset:offset+outputs-1)';

    % Backpropagation learning of the MLP ANN.
    [net,tr]=train(net,pn,tn,[],[],val);

    % Simulation of the MLP ANN on the test set.
    results = sim(net, pntest);
    results = poststd(results, meant, stdt);

    % Calculation of classification accuracy
    errors = sign(tntest) - sign(results);
    [m,n] = size(errors);
    for j=1:m
        row_error(j) = 0;
        for i=1:n
            row_error(j) = row_error(j) + sign(abs(errors(j,i)));
        end
        accuracy(j) = 1 - row_error(j)/n;
    end

    % Storing of the classification accuracy in ObjV
    ObjV(iteration) = m - sum(accuracy);
```

```
end

% Tracking of best individual and convergence display

% Storing of best fitness result for first generation
Best(gen+1) = min(ObjV); % Best fitness result of each generation
OldBest = Best(gen+1);   % Best fitness result of previous generation
no_improvement = 0;      % Stagnant performance counter

% Plotting of first point in GA results
figure(10);
plot(Best,'ro');xlabel('generation'); ylabel('error');
text(0.5,0.95,['Best = ', num2str(Best(gen+1))],'Units','normalized');
drawnow;

% Generation loop.
while gen < MAXGEN,

    % Assigning of fitness-value to entire population using
    % classification accuracy from ObjV.
    FitnV = ranking(ObjV);

    % Selection of individuals for breeding.  Creation of new
    % offspring population.
    SelCh = select('sus', Chrom, FitnV, GGAP(ggap_counter));
    [SelCh_row SelCh_col] = size(SelCh);

    % Mutation on offspring

    % Stagnant performance counter check.  If it exceeds
    % 5, then change the mutation probability to 1 in this
    % round and reset no_improvement.
    if no_improvement > 5
        mut_prob = 1;
        mut_shrink = 0;
        no_improvement = 0;
    end

    % Mutation.
    SelCh = mutbga(SelCh,FieldD, [mut_prob mut_shrink]);
    mut_prob = MUTPROB(mutprob_counter);

    % Increase of mut_shrink size to reduce mutation magnitude.
    mut_shrink = (1-(gen+1)/MAXGEN);

    % Performance evaluation of offspring

    % Creation of MLP ANN.
    net = newff(minmax(pn), [nodes,4], {'tansig', 'tansig'}, 'trainrp');
    net.trainParam.show = 25;
    net.trainParam.epochs = 300;

    % Initialisation of offspring classification accuracy
    % matrix.
    ObjVSel = ones(SelCh_row,1);

    % Iterate through all the chromosomes in the off spring
    % population.
    for iteration = 1:SelCh_row
        offset = 1;
```

```
                % Decoding of the offspring chromosomes into ANN weights and biases.
                for i = 0:(inputs-1)
                    net.iw{1}(:,i+1) = SelCh(iteration,(offset+i*nodes):(offset+((i+1)*nodes-
1)))';
                end
                offset = offset+nodes*inputs;
                for i = 0:(outputs-1)
                    net.lw{2,1}(i+1,:) = SelCh(iteration,(offset+i*nodes):(offset+((i+1)*
nodes-1)));
                end
                offset = offset+nodes*outputs;
                net.b{1} = SelCh(iteration,offset:offset+nodes-1)';
                offset = offset+nodes;
                net.b{2} = SelCh(iteration,offset:offset+outputs-1)';

                % Backpropagation learning of the MLP ANN.
                [net,tr]=train(net,pn,tn,[],[],val);

                % Simulation of the MLP ANN on the test set.
                results = sim(net, pntest);
                results = poststd(results, meant, stdt);

                % Calculation of classification accuracy
                errors = sign(tntest) - sign(results);
                [m,n] = size(errors);
                for j=1:m
                    row_error(j) = 0;
                    for i=1:n
                        row_error(j) = row_error(j) + sign(abs(errors(j,i)));
                    end
                    accuracy(j) = 1 - row_error(j)/n;
                end

                % Storing of the classification accuracy in ObjVSel
                ObjVSel(iteration) = m - sum(accuracy);
            end

            % Reinsertion of offspring into parent population based on
            % accuracy in ObjV and ObjVSel.
            [Chrom ObjV]=reins(Chrom,SelCh,1,1,ObjV,ObjVSel);

            % Increment of generation counter
            gen = gen+1;

            % Recording of current best individual
            Best(gen+1) = min(ObjV);

            % If there is no improvement, increment the stagnant
            % improvement counter.
            if Best(gen+1) == OldBest
                no_improvement = no_improvement + 1;
            end
            OldBest = Best(gen+1);

            % Display update
            figure(10);
            plot(Best,'ro'); xlabel('generation'); ylabel('error');
            text(0.5,0.95,['Best = ', num2str(Best(gen+1))],'Units','normalized');
            drawnow;
```

```
            end % Loop for next generation.

            % Update of result for parameters NIND, GGAP and MUTPROB
            result{nind_counter,ggap_counter,mutprob_counter} = Best;

            % End of GA

        end % Next MUTPROB value
    end % Next GGAP value
end % Next NIND value
```

Appendix C Detailed Classification Results

Appendix C provides the detailed classification results of the artificial neural network and support vector machine-based classifiers in the experiments in chapter 4.

Experiments in section 4.1.1

Table C-1 Detailed results of SVM-based classifiers in Table 4-1

Device	Linear SVM Accuracy (%)			Polynomial SVM Accuracy (%)			RBF SVM Accuracy (%)		
	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification
PC CPU	0	0	100	0	0	100	0	0	100
Monitor	0.6	0	99.4	0	0	100	0	0	100
PC CPU (Shutdown)	0	0	100	0	0	100	0	0	100
DC P.S. (0.1A)	11.4	14.5	74.1	1.4	0	98.6	8.3	12.4	79.3
DC P.S. (0.5A)	10.8	12.2	77.0	2.1	0.2	97.7	10.8	12.2	77.0
DC P.S. (0.25A)	18.4	16.5	65.1	7.4	1.0	91.6	18.4	16.8	64.8
DC P.S. (0.4A)	13.5	10.1	76.4	6.6	0	93.4	14.1	11.0	74.9
Notebook	0	0	100	0	0	100	0	0	100
M. charger	25.3	20.8	53.9	3.5	0.2	96.3	24.5	19.0	56.5
F. lamp	0	0	100	0	0	100	0	0	100

Experiments in section 4.1.2

Table C-2 Detailed results of SVM-based classifiers in Table 4-3

Device	Linear SVM Accuracy (%)			Polynomial SVM Accuracy (%)			RBF SVM Accuracy (%)		
	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification
PC CPU	1.1	1.1	97.8	0.1	0	99.9	0.6	0.4	99.0
Monitor	0.4	1.2	98.4	0.1	0	99.9	0.3	0.7	99.0
PC CPU (Shutdown)	2.6	3.6	93.8	0.3	0	99.7	1.4	2.1	96.4
DC P.S. (0.1A)	13.7	14.1	72.2	4.5	3.9	91.6	13.4	11.9	74.7
DC P.S. (0.5A)	10.9	14.7	74.4	7.8	7.9	84.3	9.1	13.0	77.9
DC P.S. (0.25A)	17.3	16.5	66.2	16.0	8.8	75.2	18.2	15.3	66.5
DC P.S. (0.4A)	14.6	15.6	69.9	16.0	10.3	73.6	13.1	16.5	70.5
Notebook	0.6	0.8	98.6	0.3	0.4	99.3	0.6	0.5	98.9
M. charger	21.8	17.1	61.1	16.8	11.1	72.1	19.7	17.7	62.6
F. lamp	3.8	4.9	91.3	2.5	1.0	96.5	2.7	3.9	93.4

Experiments in section 4.1.3

Table C-3 Detailed results of SVM-based classifiers in Table 4-5

Device	Linear SVM Accuracy (%)			Polynomial SVM Accuracy (%)			RBF SVM Accuracy (%)		
	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification
PC CPU	0	0	100	0	0	100	0	0	100
Monitor	0	0.1	99.9	0	0	100	0	0	100
PC CPU (Shutdown)	0	0.1	99.9	0	0.1	99.9	0	0.1	99.9
DC P.S. (0.1A)	4.9	3.7	91.4	4.6	3.7	91.7	4.3	4.6	91.1
DC P.S. (0.5A)	10.6	7.3	82.1	11.5	4.4	84.1	10.8	6.5	82.7
DC P.S. (0.25A)	15.2	13.1	71.7	15.6	10.7	73.7	14.9	12.8	72.3
DC P.S. (0.4A)	12.3	15.2	72.5	13.7	13.2	73.1	11.0	16.3	72.7
Notebook	0.3	0.3	99.4	0.1	0.3	99.6	0.3	0.3	99.4
M. charger	15.0	15.0	70.0	10.8	14.4	74.8	10.0	14.9	75.1
F. lamp	2.6	0.9	96.5	1.9	0.4	97.7	2.2	0.7	97.1

Experiments in section 4.2.1

The K-fold test performed is illustrated in Figure C-1.

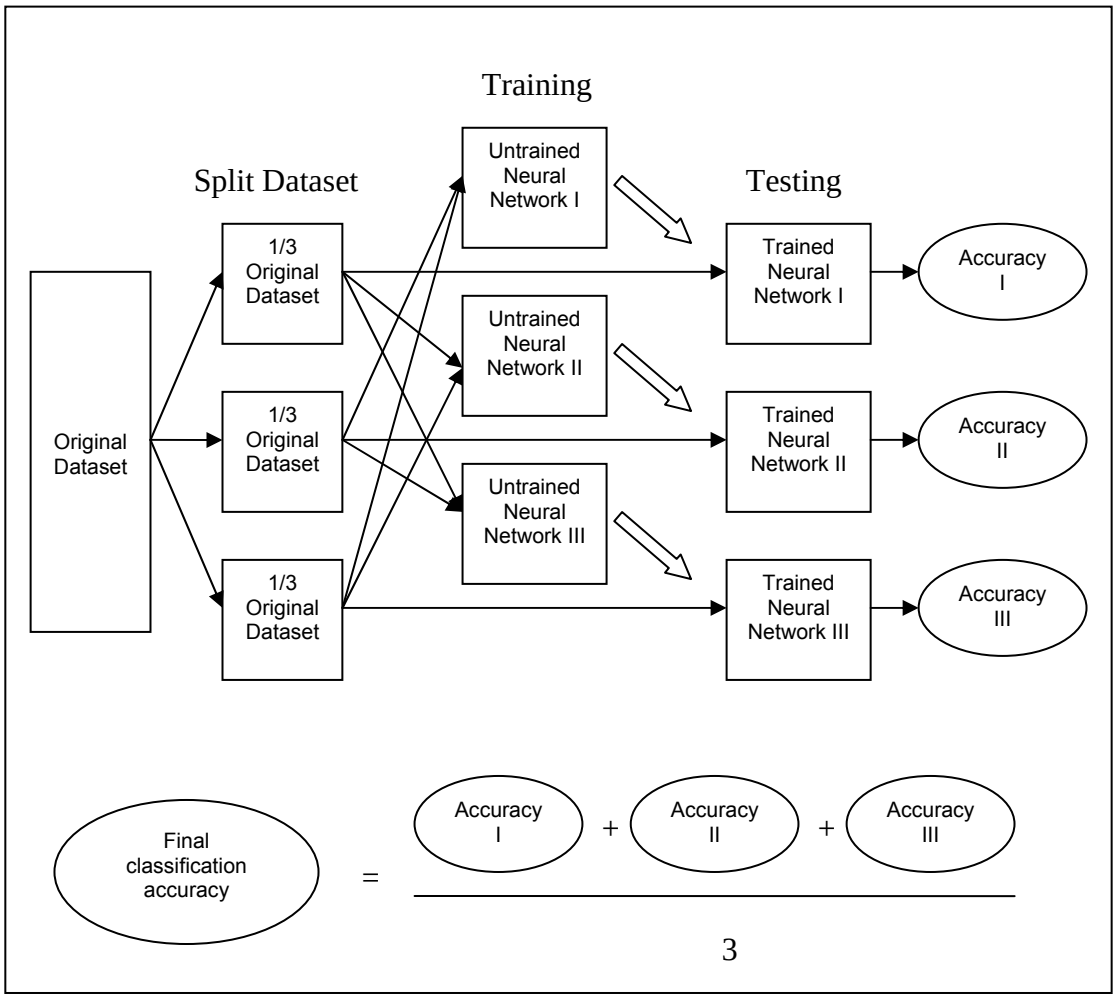


Fig. C-1 K-fold test algorithm

Table C-4a Detailed results of ANN classifiers in Table 4-7

Device	MLP Accuracy (%)			RBF Accuracy (%)		
	Misclassification*	False Alarm+	Correct Classification	Misclassification*	False Alarm+	Correct Classification
Monitor	0	0	100	0	0	100
CPU	0.1	0	99.9	0	02	99.8
Fluo. lamp	0	0.1	99.9	0	0.2	99.8
TV	0.2	0.2	99.6	0.2	0	99.8
Charger	0.1	0	99.9	0.3	0	99.7
Fan	0.1	0	99.9	0.1	0.1	99.9
Fridge	0	0.2	99.8	0	0.3	99.7
Light bulb	0	0	100	0.2	0	99.8

Table C-4b Detailed results of SVM-based classifiers in Table 4-7

Device	Linear SVM Accuracy (%)			Polynomial SVM Accuracy (%)			RBF SVM Accuracy (%)		
	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification
Monitor	0.9	0	99.1	0.5	0	99.5	0.5	0	99.5
CPU	0.7	3.6	95.7	0.2	0.2	99.6	0.7	0.1	99.2
F. lamp	0.3	0	99.7	0.2	0	99.8	0.2	0	99.8
TV	11.7	2.2	86.1	0.3	0.8	98.9	1.0	1.7	97.3
Charger	0.1	0.3	99.6	0.1	0.1	99.8	0.1	0.2	99.7
Fan	18.6	24.4	57.0	2.3	2.3	95.4	11.1	6.6	82.2
Fridge	0.1	0	99.9	0	0	100	0.1	0	99.9
L. bulb	0.1	0.2	99.7	0.1	0.2	99.7	0.1	0.1	99.7

Table C-5a Detailed results of ANN classifiers in Table 4-8

Device	MLP Accuracy (%)			RBF Accuracy (%)		
	Misclassification*	False Alarm+	Correct Classification	Misclassification*	False Alarm+	Correct Classification
Monitor	0.1	0	99.9	0	0.1	99.9
CPU	0.3	0.3	99.4	0.2	0	99.8
Fluo. lamp	0.1	0	99.9	0	0.1	99.9
TV	0.4	0.6	99.0	0	0.1	99.9
Charger	0.1	0.1	99.8	0.1	0	99.9
Fan	0.2	0	99.8	0.4	0.7	98.9
Fridge	0	0.1	99.9	0	0	100
Light bulb	0	0	100	0.1	0	99.9

Table C-5b Detailed results of SVM-based classifiers in Table 4-8

Device	Linear SVM Accuracy (%)			Polynomial SVM Accuracy (%)			RBF SVM Accuracy (%)		
	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification
Monitor	5.0	0.1	94.9	0.5	0	99.5	2.3	0	97.7
CPU	9.4	12.7	77.9	1.4	0.1	98.5	2.1	8.6	89.3
F. lamp	0	0	100	0	0	100	0	0	100
TV	17.5	15.1	67.5	7.1	0.9	92.0	14.4	15.2	70.4
Charger	0.7	1.9	97.4	0.1	0.1	99.8	0.2	0.4	99.4
Fan	21.7	19.6	58.6	11.7	4.2	84.1	22.3	19.3	58.4
Fridge	0	0	100	0	0	100	0	0	100
L. bulb	0	0.2	99.8	0	0	100	0	0.1	99.9

Experiments in section 4.2.2

Table C-6a Detailed results of ANN classifiers in Table 4-9

Device	MLP Accuracy (%)			RBF Accuracy (%)		
	Misclassification*	False Alarm+	Correct Classification	Misclassification*	False Alarm+	Correct Classification
Monitor	0.2	1.3	98.5	0.1	0	99.8
CPU	8.2	4.9	86.9	12.2	0.4	87.4
Fluo. lamp	0.2	0	99.8	0.5	0	99.5
TV	19.5	12.4	67.9	0.4	11.6	88.0
Charger	0	33.9	66.1	0	34.9	65.1
Fan	28.5	9.4	62.1	13.1	17.8	69.1
Fridge	0	1.2	98.8	1.1	0	98.8
Light bulb	2.1	0	97.9	20.8	0	79.2

Table C-6b Detailed results of SVM-based classifiers in Table 4-9

Device	Linear SVM Accuracy (%)			Polynomial SVM Accuracy (%)			RBF SVM Accuracy (%)		
	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification
Monitor	0.6	6.9	92.5	0.5	0.1	99.4	0.5	0.8	98.7
CPU	25.5	0.1	74.4	25.4	0	74.6	24.9	0.1	75.1
F. lamp	0.1	0	99.9	0.1	0	99.9	0.1	0	99.9
TV	19.8	16.6	63.6	2.7	6.9	90.4	4.3	17.3	78.4
Charger	0	30.4	69.6	0	28.5	71.5	0	29.9	70.1
Fan	28.6	5.3	66.1	27.8	4.1	68.1	27.8	3.7	68.5
Fridge	0.1	0	99.9	0.1	0	99.9	0.1	0	99.9
L. bulb	6.8	0	93.2	5.0	0	95.0	5.5	0	94.5

Experiments in section 4.2.3

Table C-7a Detailed results of ANN classifiers in Table 4-11

Device	MLP Accuracy (%)			RBF Accuracy (%)		
	Misclassification*	False Alarm+	Correct Classification	Misclassification*	False Alarm+	Correct Classification
Monitor	6.6	0.5	88.3	4.4	23.8	71.8
CPU	14.9	13.9	71.2	15.9	20.3	63.8
Fluo. lamp	8.1	7.7	84.2	4.8	37.1	58.1
TV	15.6	16.4	68.0	5.0	42.3	52.7
Charger	15.2	18.0	66.8	9.1	37.8	53.1
Fan	18.1	22.4	59.5	4.0	45.7	50.3
Fridge	5.0	6.0	89.0	3.5	28.9	67.6
Light bulb	9.6	11.8	78.6	3.4	41.7	54.9

Table C-7b Detailed results of SVM-based classifiers in Table 4-11

Device	Linear SVM Accuracy (%)			Polynomial SVM Accuracy (%)			RBF SVM Accuracy (%)		
	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification
Monitor	6.4	4.6	89.0	6.4	4.6	89.0	6.8	4.9	88.3
CPU	13.7	14.9	71.4	13.7	14.9	71.4	14.8	13.8	71.4
F. lamp	7.5	7.8	84.7	7.8	7.5	84.8	7.7	7.5	84.8
TV	17.5	17.8	64.7	17.5	17.8	64.7	18.8	16.2	65.0
Charger	16.4	17.5	66.1	14.4	19.1	66.5	15.0	19.0	66.1
Fan	18.9	19.5	61.6	18.0	20.0	62.0	15.9	22.6	61.5
Fridge	5.9	5.5	88.6	5.7	5.4	89.0	5.7	5.5	88.8
L. bulb	10.3	10.9	78.8	10.3	10.9	78.8	10.3	11.1	78.6

Experiments in section 4.3

Table C-8 Detailed results of ANN classifier in Table 4-12

Device	Accuracy (%)		
	Misclassification*	False Alarm+	Correct Classification
Monitor	0	2.3	97.7
CPU	4.0	0.5	95.5
Fluorescent lamp	0	0.2	99.8
Television	23.8	0.7	75.5
Soldering iron	8.0	0	92.0
Fridge	0	0.3	99.7
Fan	18.2	4.0	77.8
Battery charger	0	6.2	93.8
Light bulb	28.6	0	71.4
Power drill	0	6.6	93.4

Table C-9 Detailed results of ANN classifier in Table 4-13

Device	Accuracy (%)		
	Misclassification*	False Alarm+	Correct Classification
PC CPU	0.5	0	99.5
PC Monitor	0.2	0.3	99.5
PC CPU (shutdown)	0.9	0.3	98.8
DC Power supply (0.1A)	8.4	8.1	83.5
DC Power supply (0.5A)	12.8	10.7	76.5
DC Power supply (0.25A)	15.1	11.9	73.0
DC Power supply (0.4A)	8.8	18.6	72.6
Notebook computer	0.7	0	99.3
Mobile phone charger	9.5	13.3	77.2
Fluorescent lamp	4.2	5.3	90.5

Experiments in section 4.4

Table C-10 Detailed results of ANN classifier in Table 4-14

Device	Accuracy (%)		
	Misclassification*	False Alarm+	Correct Classification
Motor #1	6.1	6.4	87.5
Motor #2	0.1	0	99.9
Motor #2 with capacitors	1.3	1.0	97.7
Inverter #1	0.2	0.2	99.6
Inverter #2 (low frequency)	0.2	0.2	99.6
Inverter #2 (high frequency)	0	0.4	99.6
Fluorescent lamp without capacitor	18.4	16.3	65.1
Fluorescent lamp with capacitors	15.1	15.8	69.1

Experiments in section 4.5

Table C-11 Detailed results of ANN classifier in Table 4-15

Device	Accuracy – < 3 of each device (%)			Accuracy – <10 of each device (%)		
	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification
PC CPU	0	0	100	0	0	100
PC Monitor	0	0	100	0	0	100
PC CPU (shutdown)	0	0	100	0	0	100
DC Power supply (0.1A)	0	0	100	0	0	100
DC Power supply (0.5A)	0	0	100	2.7	0	97.3
DC Power supply (0.25A)	0	0	100	4.5	0	95.5
DC Power supply (0.4A)	0	0	100	6.0	0.5	93.5
Notebook computer	0	0	100	0	0	100
Mobile phone charger	0	0	100	3.2	0	96.8
Fluorescent lamp	0	0	100	0	0	100

Experiments in section 4.5

Table C-12 Detailed results of TDNN classifiers in Table 4-16

Device	MLP Accuracy (%)			TDNN-1 Accuracy (%)			TDNN-2 Accuracy (%)		
	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification	Misclassi-fication*	False Alarm+	Correct Classification
F. lamp	0	0	100.0	0	0	100	0	0	100
CPU	1.7	0	98.3	0.9	2.9	96.2	0	2.8	97.2
Monitor	1.0	3.6	95.4	1.0	5.6	93.4	3.0	1.8	95.2
TV	5.2	3.1	91.7	5.2	5.3	89.5	3.5	6.2	90.3

* Misclassification refers to wrong classification of devices present
+ False alarm refers to wrong classification of devices absent

Appendix D ANN and SVM Techniques

Appendix D discusses various architectures of the artificial neural network (ANN) and multi-class support vector machine (SVM) models in signature identification, discussing the benefits and disadvantages of each architecture.

D.1 ANN Architectures

In terms of learning algorithm, the ANN can be divided into two main classes:- supervised learning and unsupervised learning. Supervised learning ANNs refer to the class of ANNs that are trained to map inputs to specified targets whereas unsupervised learning ANNs are free to form clusters, automatically classifying the inputs. In this thesis, the ANN architectures that will be discussed include the self organizing map (SOM) that belongs to the unsupervised learning class and multilayer perceptron (MLP) and radial basis function (RBF) networks that belong to the supervised learning class [36].

D.1.1 Self Organizing Map (SOM)

The SOM is a powerful ANN that is capable of automatically clustering the input vectors to which it is fed, thus learning the distribution of the input vectors. Therefore, it is also seen as a feature extractor, capable of characterizing or identifying the similarities between input vectors.

The SOM consists of a multidimensional lattice of neurons (Figure D-1). During the training period, when an input vector is fed to the SOM, the neuron with the smallest Euclidean distance from the input vector will be chosen as the winning neuron. The weights of the winning neuron and its neighbors will be updated such that the likelihood of it winning the next time a similar input vector is presented will be higher. As a result, the winning neuron and its neighbors will slowly shift towards the input vector, thus learning its distribution.

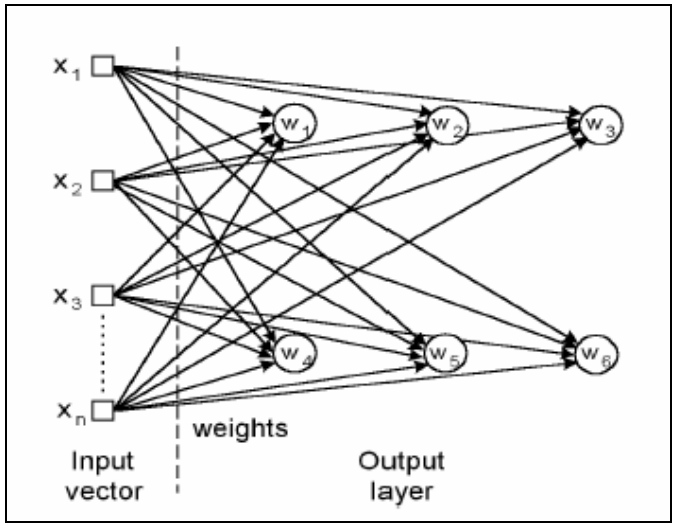


Fig. D-1 SOM with n inputs and 6 output neurons in a 2-dimensional lattice

In the field of signature identification, SOM has been successfully applied in [11,13,14] to automatically classify the input vectors into various power quality disturbance categories. In [14], the SA-ANN, an SOM with automatic structuring of the number of nodes was applied to classify power quality disturbances. Each neuron in the SA-ANN had an additional deviation vector that tracked how groups of input vectors were scattered around the neuron and a counter that contained the number of input vectors represented by the neuron. The proposed algorithm allowed an increase in the number of neurons to improve performance, or the merging of two neurons

without significantly affecting the performance. Paper [13] proposed the use of the learning vector quantization (LVQ), which is a supervised version of the SOM.

In some cases, SOM training requires a large number of epochs to converge to the distribution of the input vectors. The training length also depends on the selection of initial weights, which are usually randomly chosen. Besides that, it is difficult to control the number of clusters that the SOM forms unless the training is supervised. Correspondence between the clusters and the actual input vector characteristic may be unpredictable, with missing or unexpected new characteristics [36].

In this thesis, the training data available have inputs with corresponding outputs. As the number of devices increases, each combination of devices will require an output neuron, thus causing the number of neurons in the SOM to grow exponentially. Therefore, the SOM was not used in this thesis but remains a potential tool when future works on automatic detection of unknown devices require unsupervised learning.

D.1.2 Multilayer Perceptron (MLP)

The MLP is a universal classifier and a non-linear function approximator. Under supervised learning, it can be trained to perform non-linear mapping between input and output vectors.

An MLP consists of an input layer, with the number of neurons equal to the dimension of the input vector, an output layer, with the number of neurons equal to the

dimension of the output vector, and one or more hidden layers with an arbitrarily chosen number of hidden neurons. In most practical applications, only one hidden layer is used. In [32], it was shown that larger numbers of hidden layers and nodes resulted in slower convergence but did not improve performance. Unlike the SOM that activates only one output neuron for each input vector, the multiple output feature of the MLP allows it to represent combinations of devices present by producing positive or negative values at each output neuron which represents a particular device.

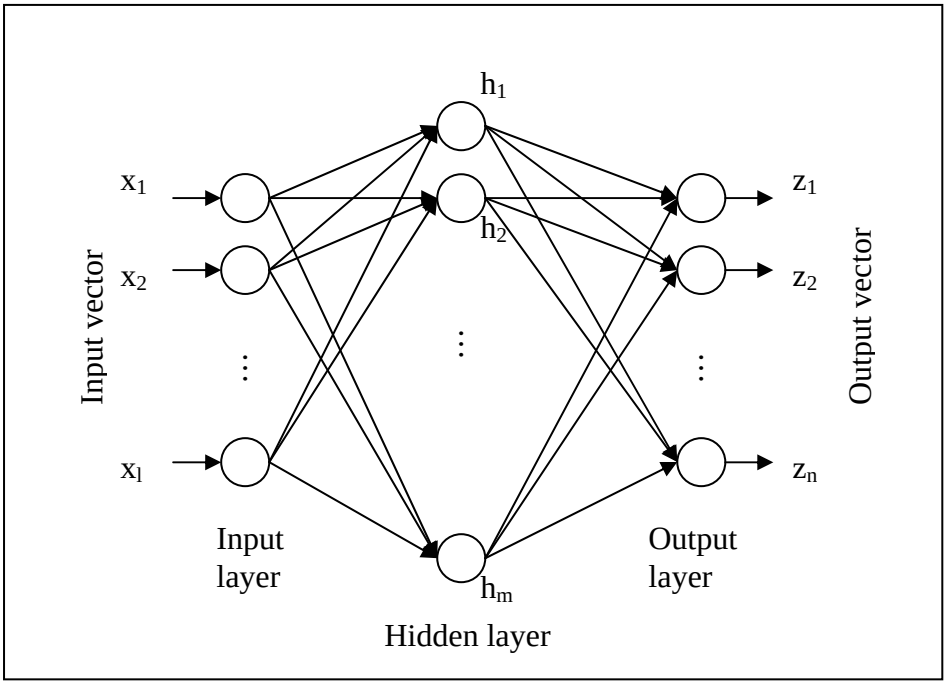


Fig. D-2 MLP with a single hidden layer

During training, the weights and biases of the MLP are updated towards minimizing the least mean squares (LMS) of the error between the neural network output and the desired output. In this thesis, the backpropagation (BP) training algorithm given in equation (D.1) was the primary training algorithm because of its simplicity, smaller memory requirement and higher speed in pattern recognition problems.

$$w(t+1) = w(t) - \eta \frac{\partial E}{\partial w}(t) \qquad \text{(D.1)}$$

$$E = \sum_j \frac{1}{2}\left[y'(j) - z(j)\right]^2 \qquad \text{(D.2a)}$$

$$\frac{\partial E}{\partial w} = -\sum_j \left[y'(j) - z(j)\right]\phi'(v(j))X(j) \qquad \text{(D.2b)}$$

where w is the weight associated to an input of the neuron and η is known as the learning rate which is arbitrarily chosen. w is updated based on the gradient of the error function E with respect to w. $y'(j)$ and $z(j)$ are the desired output and neuron outputs respectively for input j. $\Phi'(v(j))$ is the derivative of the neuron activation function evaluated at preactivation value $v(j)$. $X(j)$ is the input vector value.

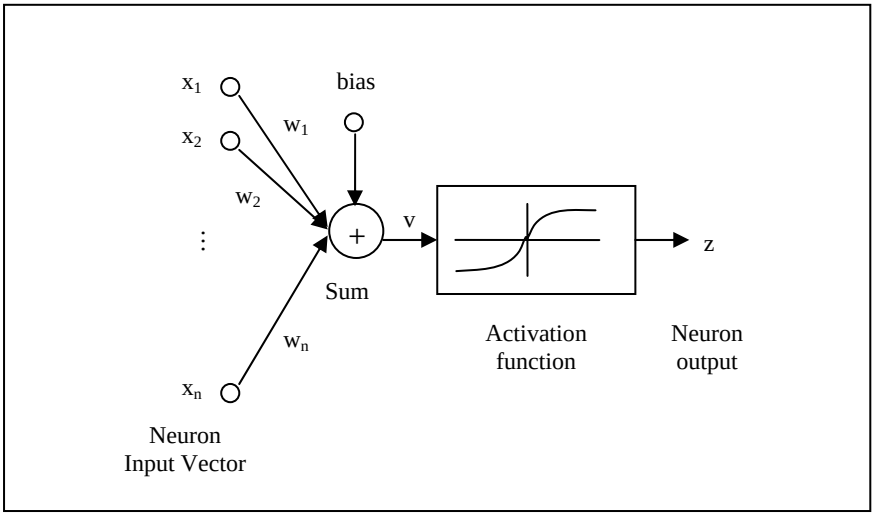


Fig. D-3 Perceptron architecture

Proper structuring of the MLP such as the number of neurons, connection between neurons, number of layers and activation function of the neurons play an important role in its optimization and performance. In [9], partially connected MLP was shown to perform better than a fully connected MLP where a neuron is connected

to all neurons in the adjacent layers. Adaptive learning BP such as the one proposed in [12] or the resilient backpropagation (RP) [33] greatly increased the convergence rate. In [9,10,12], the MLP was used to perform harmonic detection in power systems so as to indicate necessary harmonic compensation from active filters. The MLP was also applied to the identification of electrical load signatures in [26] and [28]. In [26], multiple MLP were placed in cascade to perform binary classifications while traversing a family tree. The MLP's high noise tolerance was demonstrated in [34] where it was applied to recognize the myoelectric control signal used to trigger a functional neuromuscular stimulation.

Nonetheless, the MLP faces the problem of overfitting that may cause it to become susceptible to noise and to lose its generalization ability. Fortunately, overfitting can be avoided by using a test set to measure its performance and to end the training when the test set error increases. Besides that, when using the BP training algorithm, the MLP tends to get trapped at a local minimum. Therefore, care should be taken in the selection of the initial weights.

D.1.3 Radial Basis Function (RBF) Neural Networks

Similar to the MLP, the RBF neural network is also a universal classifier and non-linear function approximator. While also being similar structurally, the fundamental difference between the RBF neural network and MLP lies in the way the hidden neurons combine inputs from the preceding layers in the network; the MLP uses the inner products whereas RBF uses the Euclidean distance. The most popular RBF is the Gaussian RBF. The Gaussian function is given by equation (D.3)

$$g(u) = e^{\left(-\frac{u^2}{2\sigma^2}\right)} \qquad \text{(D.3)}$$

where u is the Euclidean distance of the input vector from the center of the RBF neuron and σ is the Gaussian width variable.

The RBF neural network usually has only 1 hidden layer. Each neuron (Figure D-4) in the hidden layer consists of 1 center, which in the case of the Gaussian RBF is the mean of the Gaussian function. In the literature, there are many methods of choosing the centers, usually from the set of input vectors [36]. In function approximation, each center usually represents a part of the curve. Due to this structure, the Gaussian RBF neural network usually performs well for interpolation but poorly for extrapolation. During training, the Euclidean distances of the input vector from the centers are calculated and the weights of connections between the hidden layer and the output layer are updated towards minimizing the least mean square of the error between the neural network outputs and the desired outputs.

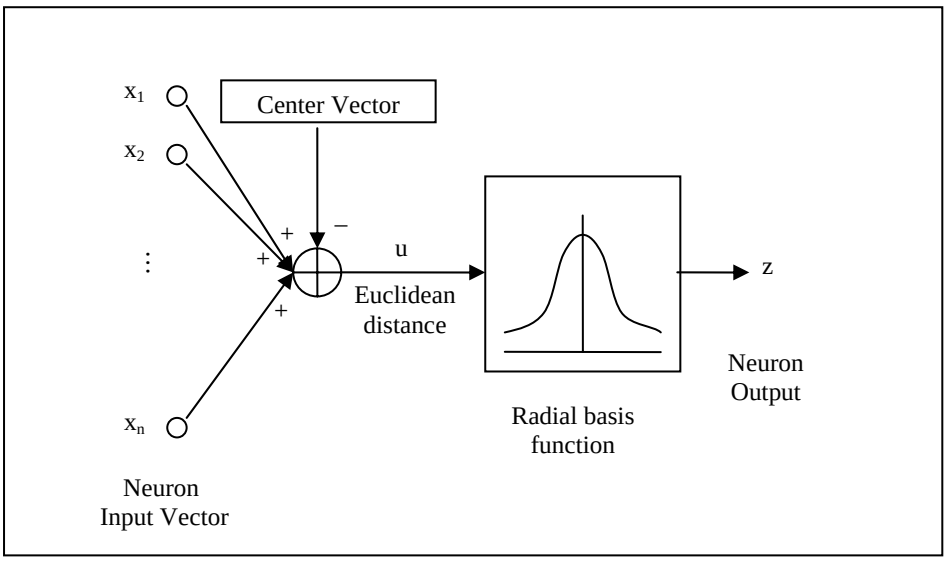


Fig. D-4 RBF neuron architecture

In [35], a two-layer MLP and an restricted Coulomb energy (RCE) network, which is a variety of RBF neural network, were used in the recognition of pen signatures. The RCE network showed better generalization abilities than the MLP in classification tasks but failed in verification tasks.

To optimize the RBF neural network, the centers must be selected properly so as to cover the maximum variance of the input vector set. For the Gaussian RBF, the Gaussian function width can be varied to find the optimum value and preferably should be sufficient to cover the area between neighboring centers. The RBF neural network also faces the problem of overfitting which can be similarly dealt with using a test set. The RBF neural network was also chosen as a classifier in this thesis due to its structural and functional similarity to the MLP that met the requirements of the thesis.

D.1.4 Other ANN Architectures

Some of the architectures beside those mentioned above that may be suitable for signal processing and signature identification are the probabilistic neural network (PNN) [36] and time delay neural network (TDNN) [37].

PNN is a universal approximator for smooth class-conditional densities and therefore should be able to solve any smooth classification problem given enough data. It can be viewed as a normalized RBF network with a hidden unit centered at every training case. If all the inputs are relevant, PNN has the very useful ability to tell

whether a test case is similar to any of the training data; if not, the classification made was based on extrapolation and should be viewed with skepticism.

The PNN required reasonable signal-to-noise ratio (SNR) to achieve good performance. PNN tended to have a long computational time because of the vast amount of neurons it contained. Due to its similarity to the RBF neural network, the PNN was not used in this thesis.

Time delay neural network is a dynamic neural network capable of using temporal information by utilizing time delayed state information in its hidden layers. In [37], the artificial neural network is composed of a preprocessor based on principal component analysis (PCA) and a one-hidden layer time delay neural network trained with backpropagation. The TDNN was used because information to discriminate four classes of sounds was contained in the history of the time series and the TDNN trained on the shape of the waveform.

In the discussion of the conventional NALM techniques (Section 1.3), the step change in steady state power was noted as the main form of signature used to identify the various electrical devices. Although the proposed method in this thesis has only considered the instantaneous current harmonics pattern as the signature, the focus was on a singular quantity of each device. The limitation of using only instantaneous harmonics information appears when the number of devices increases such that it becomes impossible to uniquely disaggregate the individual devices from the total load.

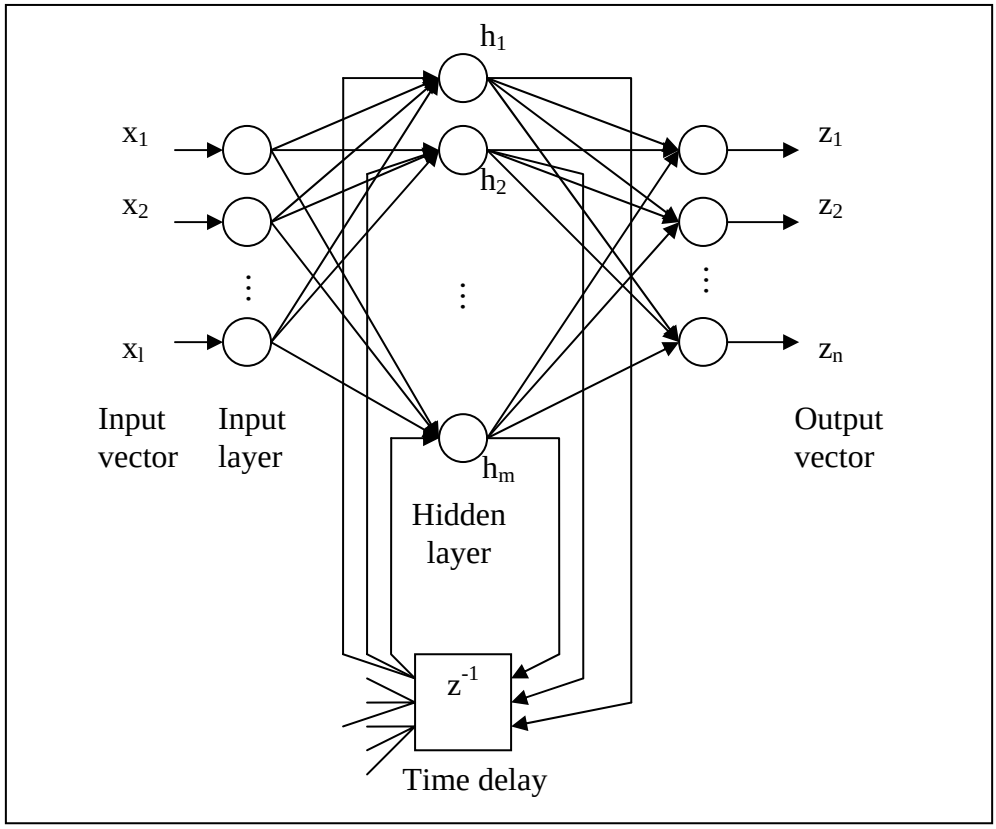


Fig. D-5 Elman TDNN Architecture

The TDNN offers the ability to utilize past information in the input-output mapping of the ANN. Therefore, by continuously tracking the states of the electrical devices and detecting step changes in the current harmonics information, theoretically the TDNN will be able to track an unlimited number of electrical devices. In this thesis, a simple feasibility study was performed on the use of TDNN for device classification in NALM.

D.2 Support Vector Machines (SVM)

SVM is a learning technique that can be seen as a new method for training neural network, polynomial, or radial basis functions classifiers. It is essentially a two-class linear classifier in a high dimensional feature space that may be nonlinearly

related in the input space. Linear equations are used to separate the high dimensional feature space into two regions. SVMs are very well founded from the mathematical point of view, being an approximate implementation of the structural risk minimization induction principle [38]. The decision surfaces are found by solving a linearly constrained quadratic programming (QP) problem.

Most real world problems can not be solved by a linear classifier, and the techniques have to be extended to allow for non-linear decision surfaces. Projecting the original set of variables into a higher dimensional feature space is a possible way to address this problem. Fortunately, using the kernel, the high dimensional feature space computations can be performed directly in the original input space. However, the QP problem is challenging when the size of the data set becomes large.

The generalization ability of SVM is measured by the margin between the two classes (Figure D-6). The size of the margin is governed by the divisibility of the two classes and a cost parameter that needs to be fine tuned when choosing the optimum SVM configuration. The cost parameter allows a "soft" margin to be set between the two regions of classification, ignoring outlier input vectors that could have been distorted by noise and allowing these outliers to cross the dividing hyperplane.

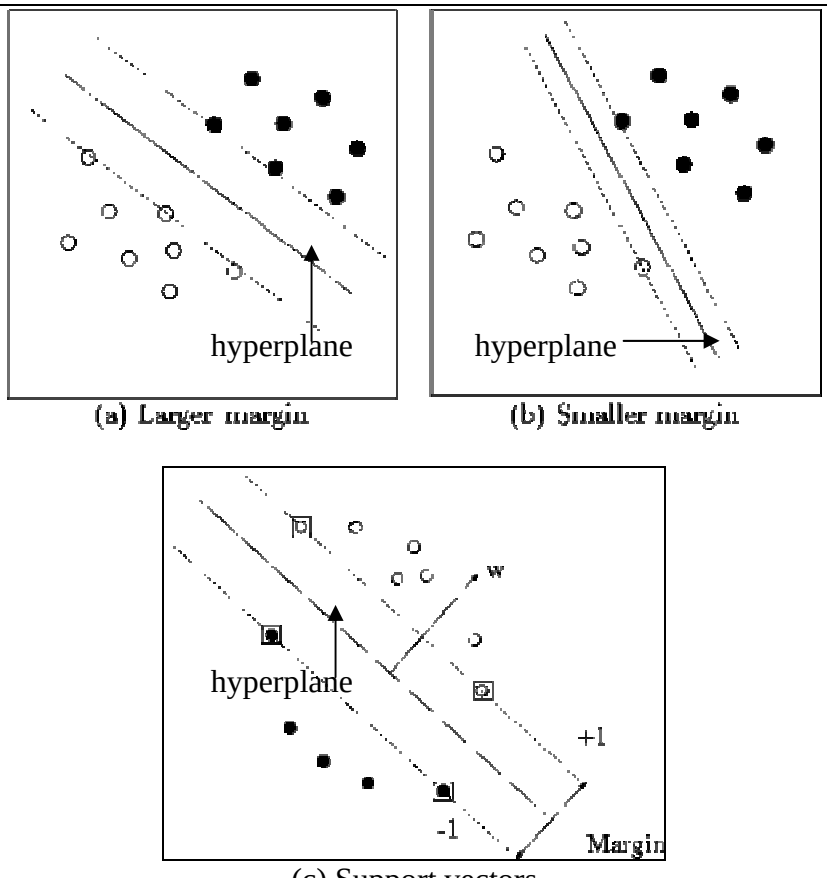
Maximizing the margin is equivalent to minimizing the objective function in equation (D.4) under the inequality constraints given by equations (D.5) and (D.6).

$$J = \frac{1}{2}\|W\|^2 + C\sum_{j=1}^{m}\xi_j \qquad \text{(D.4)}$$

$$y'(j)\left[W.X(j)-b\right]\geq 1-\xi_j \forall j=1,...,l \qquad \text{(D.5)}$$

$$\xi_j \geq 0 \forall j=1,...,l \qquad \text{(D.6)}$$

where W is the hyperplane vector, C is the cost parameter, ξ_j is the slack variable to allow errors when outlier input vectors cross the hyperplane to the wrong class, $y'(j)$ is the class label (+1 or -1), $X(j)$ is the input vector and $W.X(j)-b$ is the hyperplane equation. l is the number of input vectors in the training set.



(a) Larger margin (b) Smaller margin

(c) Support vectors

The hyperplane in (a) gives a wider margin between the two classes compared to (b). In (c) the support vectors are marked by the boxes and the hyperplane vector, w that needs to be optimized is shown.

Fig. D-6 SVM Margin

D.2.1 SVM in Multi-class Classification

Several approaches have been implemented to extend the SVM into a multi-class classifier [38,39,40,41,42]. This includes dividing the problem into several binary classifications such as in the "One versus the rest method", "Pairwise method" and "Directed Acyclic Graph SVM (DAGSVM) method" [38]. There are also approaches to multi-class problems that involve solving one single optimization problem by implementing decomposition method [38,39,40,41]. Reference [42] proposed a multistage SVM which repeatedly clustered the samples into two classes until the final class was obtained.

In the "One versus the rest method", for a *k*-classes classification, a total of *k* SVM classifiers are created. Each SVM classifier will differentiate class *k* from the remaining classes. The SVM classifier which produces the largest value corresponds to the correct class.

In the "Pairwise method" [43], the classes are paired up. A total of *k(k-1)/2* SVM classifiers are created to divide the training data into one of the classes in the pairs. Each SVM classifier produces a vote for one of the two classes. Finally, the class which collects the highest number of votes is identified as the correct class.

Another approach, "DAGSVM method" modifies the "Pairwise method" by putting the SVM classifiers for class pairs into a binary tree (Figure D-7). The correct class is determined by traversing down a tree until reaching a leave node, which represents the correct class [44,45,46,47]. This method significantly reduces the

problem complexity from $O(k^2)$ to $O(k)$. On the other hand, it also introduces the problem of choosing the tree structure, to arrange the order of classes for comparison.

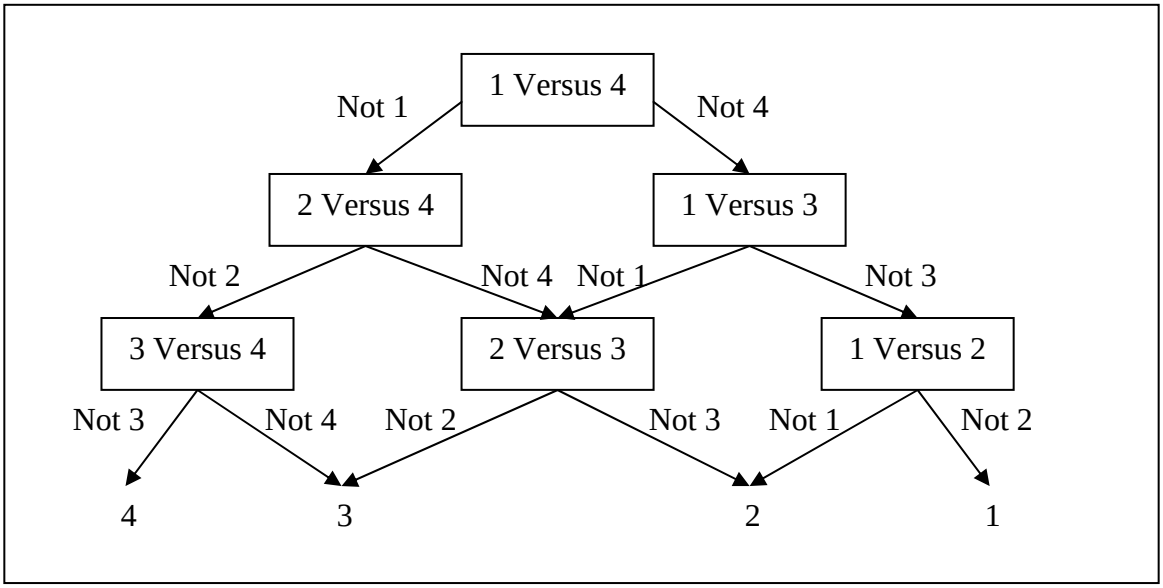


Fig. D-7 Example of the Directed Acyclic Graph SVM (DAGSVM)

However these conventional approaches have typically focused on mutually exclusive classes, and posed certain limitations when applied to problems that require non-mutually exclusive multi-class classifications. All the multi-class SVM approaches discussed above only produce a single class result. The classes must be mutually exclusive, otherwise, the input vector which holds a signature representing more than one class is classified as only the class with the highest SVM classifier output value.

In [48], a "Pairwise method" with associated probability to the results was introduced. In [49], by normalizing the distance from the input vector to each hyperplane of the SVM, the relative distance was obtained to provide a form of ranking. Unfortunately, these approaches did not present a ranking so as to allow

more than one class to be specified. Instead, the main aim was to provide a comparison between classes and thus selecting a single correct class.

D.3 Evolving the ANN Weights Using Genetic Algorithm (GA)

Genetic algorithm (GA) is a class of evolutionary algorithm that is inspired by the concept of natural evolution. In GA, the solutions to a problem are represented by chromosomes or strings of numbers.

The GA randomly creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best solution. In each generation, an arbitrary percentage of the chromosomes from the current population representing the best solutions are chosen to be evolved. The fitness or qualities of the solutions in the new chromosomes are then evaluated. Finally, during reproduction, the chromosomes are ranked and the best chromosomes are used to create the next generation population.

In the crossover genetic operation, a random splicing point is chosen in two chromosomes where the two chromosomes are spliced or cut. Then the spliced regions are mixed to create two new chromosomes. In the mutation genetic operation, each element in the chromosome is randomly changed to a new value with an arbitrary mutation probability. The two genetic operations are illustrated in Figure D-8.

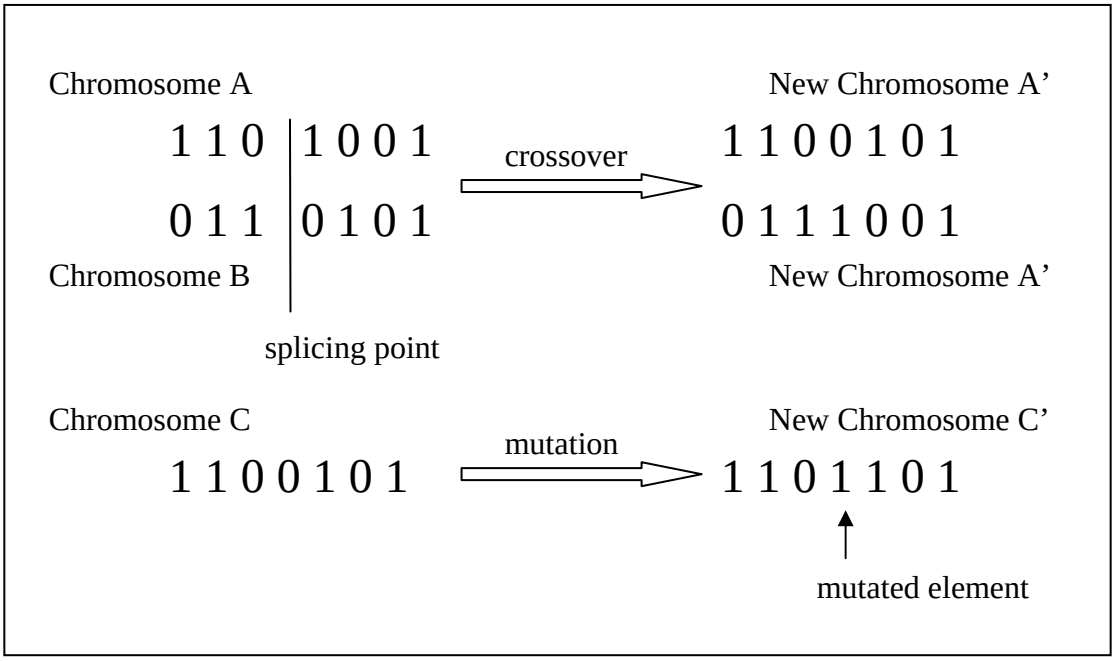


Fig. D-8 Crossover and mutation genetic operations

Artificial intelligence techniques such as GA and ANN have been widely used to solve problems of signature identification in various fields. The ANN, which provides a nonlinear mapping between the input and output vectors, usually minimizes its least mean square output error using gradient descent techniques such as backpropagation (BP) while the GA provides a stochastic mean of minimizing a given cost function. The GA, if tuned correctly, is known to be less affected by the problem of local minima and much less sensitive to initial conditions of training [50,51]. Therefore, by combining the two techniques, it is possible to tap the benefits of both worlds.

Various combinations of the GA and ANN have been researched on including the evolution of the ANN weights and architectures using the GA [50,52,53]. In the evolution of ANN weights, GA replaces the conventional gradient descent method to optimize the weights towards the global optimum. By eliminating the differentiation

process required in gradient descent, GA also removes the requirement for the activation function of the ANN neurons to be differentiable, thus allowing new types of ANN activation function to be explored. Furthermore, the architecture of the ANN such as the number of layers and the number of neurons in each layer can also be modified in the optimization process.

The evolution of the single-hidden-layer perceptron ANN weights using GA was chosen for the optimization of the ANN architecture in this thesis. Although other variants of the ANN such as the radial basis function (RBF) ANN have been successfully evolved using GA [54,55], the simple single-hidden-layer perceptron weights are naturally easier to be encoded into the chromosomes and evolved with the GA. The GA's chromosomes may hold the weight values of all the neuron connections in the ANN. Therefore, by evolving the GA chromosomes, the weights or even the whole architecture of the ANN may converge towards the global minimum of the classification function. On the other hand, GA is relatively inefficient in fine-tuned local search [50] while the BP algorithm of the ANN, which is based on gradient descent, generally has faster training phase angles and has better convergence ability. Therefore, by employing the BP algorithm of the ANN to perform a local search, an efficient and fine-tuned global minimum can be found.