www.s2is.org
International Journal on Smart Sensing and Intelligent Systems
ISSN 1178-5608

Smartphone Application for Fault Recognition

[1]Nishchal K. Verma, [1]Rahul K. Sevakula, [1]Jayesh K. Gupta, [1]Sumanik Singh, [1]Sonal Dixit, [2]Al Salour

[1]Department of Electrical Engineering, Indian Institute of Technology Kanpur, India

[2]Boeing Company, St. Louis, MO, USA

Emails: nishchal@iitk.ac.in, srahulk@iitk.ac.in, jayeshkg@iitk.ac.in, sumanik@iitk.ac.in, dsonal@iitk.ac.in, al.salour@boeing.com

Submitted: June 10, 2013 ***Accepted: Aug 08, 2013*** ***Published: Sep. 05, 2013***

Abstract: ***Smart-phones have become an essential tool for many in daily life. Smart-phone Application development has grown rapidly to fulfill the demands of increasingly diverse usage. This paper presents an application based on a fault recognition model that can be used for recognizing different acoustic patterns. The application is made to detect various faulty conditions of an industrial air compressor based on the sound generated by the machine. The application has been tested in real time and is found to perform very well with classification accuracies above 94.75%. We propose that similar applications and recognition models with some modified specifications could be used for acoustic pattern recognition in a wide range of areas.***

Index terms*:* **Feature Extraction; Android; Feature Classification; Feature Selection; Support Vector Machines**

I. INTRODUCTION

Usage of smart-phones has reached new heights. With the increasing pervasiveness and computing power of the smart-phones, they can do much work which otherwise are done using computers and sensors [1]. Already they have started displacing computer in most of the daily tasks and a little nudge in the right direction could lead to many areas of work being done by a computer being implemented on a smart-phone [2][3][4]. Industrial activities are one area which has not seen a large surge in smart-phone usage yet and have a huge scope for implementing numerous tasks on the smart-phone. The advantage of this approach would be that most of the tasks owing to the user-friendliness and simplicity can be done by unskilled manpower as compared to the traditional approach, which needs skilled human resources. Machine fault-detection is one such important task whose implementation on smartphone can provide extraordinary help to industry [5].

In this paper we present a model that can be used for acoustic pattern recognition. This model uses various feature extraction techniques, Principal Component Analysis (PCA) as feature reduction/selection algorithm and Support Vector Machine (SVM) as the classifier algorithm. All these modules have been implemented on Android platform, which is the most widely used operating system (OS) for mobiles. The paper then presents a case study where this application model is used successfully for detecting and classifying faults in an industrial air compressor.

The rest of the paper is organized as follows. Section II explains the data mining model used in our application. In Section III, details of implementation on Android platform has been discussed. In Section IV, we describe results. Section V gives the conclusion of our work.

II. DATA MINING MODEL

Here, we describe the general outline of the theory behind our application. After getting sampled acoustic data, some pre-processing steps such as clipping which refers to cropping the signal to retain the part that has most of the information, are performed for reducing the effect of various kinds of noises present during recording operation. The preprocessed data is then fed to the feature extraction module.

A. Feature Extraction Module

After preprocessing, the size of the data is still quite large to be for use in the classification process. Therefore certain characteristics need to be extracted which would represent the signal at a much lower dimension. Feature extraction is a form of dimensionality reduction, where one tries to represent the entire input data in the form of a set of features.

1. Time Domain:

   Although signals and their waveform are present in the time domain, this is the most basic representation of data and not much inference can usually be drawn [6]. Nevertheless, the following eight statistical parameters are extracted from time domain - absolute mean, maximum peak, root mean square, square mean root, variance, kurtosis factor, crest factor, shape factor and skewness of data, as exposited in [7]. See Table I.

Table I. Time Domain Features

Absolute Mean	Maximum Peak Value	Root Mean Square	Square root mean value	Variance	Kurtosis	Crest Factor	Shape Factor
$\frac{1}{N}\sum_{i=1}^{N}\lvert x_i\rvert$	$\max \lvert x_i\rvert$	$\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$	$\left(\frac{1}{N}\sum_{i=1}^{N}\sqrt{x_i}\right)^2$	$\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$	$\frac{\frac{1}{N}\sum_{i=1}^{N}(x_i-\bar{x})^4}{s^4}$	$\frac{x_p}{x_{rms}}$	$\frac{x_{rms}}{\bar{x}}$

2. Frequency Domain:

   Special characteristics of the signal which are not obvious in the time domain can be conveyed by frequency domain representation. In frequency domain, we can see how signal energy varies with frequency. For studying the frequency domain properties of the signal, the time domain signal needs to be converted to frequency domain by using Fast Fourier Transform (FFT). FFT allows us to obtain a spectrum of frequency components which can then be divided into 8 equal consecutive segments known as bins. From each bin we can get one feature, by dividing the individual bins' energy with total spectral energy, thus getting 8 features from frequency domain.

3. Wavelet Domain:

The signals that are generally encountered in real world are non-stationary in nature, i.e. their frequency spectrum changes with time. Analyzing these signals in frequency domain gives us average frequency spectrum, hence hides much information w.r.t. how the signal's frequency spectrum changes with time. In time-frequency domain, to certain extent we can study the signal in both time and frequency domains simultaneously. Wavelet transform is a popular choice for analyzing the signal in time frequency domain [8]. Short Time Fourier Transform (STFT) can be used for transforming the signal to time-frequency domain, but it is limited to providing the same at only one resolution. Wavelet Transform, on the other hand, can provide time-frequency information at multiple resolutions. We use 3 variants of Wavelet transforms for extracting features from wavelet domain, as presented below.

i) Morlet Wavelet Features [9]: A Morlet wavelet is a cosine signal that decays exponentially on both sides. Mathematically it is defined as:

$$y(t) = e^{\frac{-\beta^2 t^2}{2}} cos(\pi t) \tag{1}$$

$$y_{a,b}(t) = e^{\frac{-\beta^2 (t-b)^2}{a^2}} cos\left(\frac{\pi(t-b)}{a}\right) \tag{2}$$

where eqn. (1) represents the Morlet (mother) wavelet and eqn. (2) represents the son wavelet that is obtained by dilating and translating the Morlet wavelet by factors 'a' and 'b' respectively. The time domain signal is then transformed into the time-frequency domain by convolving it with the son wavelet at one scale. From the transformed signal, Standard Deviation, Wavelet Entropy, Kurtosis factor, Skewness, Variance, Zero crossing rate and Sum of peaks are calculated. These 7 statistical parameters are considered to be features obtained from Morlet Transform.

ii) Discrete Wavelet Transform (DWT) [10]: In terms of computational time and implementation on computers, DWT is more efficient than other Wavelet Transforms like Continuous Wavelet Transform (CWT). In case of CWT, the signals are analyzed using a set of basis functions or mother wavelets like Morlet Wavelet, Mexican Hat Function etc. The basis function at each scale is shifted throughout the signal to compute the transformed signal. For implementing CWT on computers, the wavelet series is obtained by sampling the time scale plane. Therefore its computation may consume significant amount of time, depending on the resolution required. DWT, on the other hand is based on sub band coding which is found to yield faster computation of Wavelet Transform because the signal at each level is passed through filters of different cut-off frequencies. Hence the size of the signal reduces at each level; thus decreasing the number of computations. There are several families of filters like Haar, Daubechies [11], and Coifflets etc. At each level, the signal is convoluted with low pass and high pass filters to obtain two sets of coefficients. The convolution expression is given by:

$$h[n]=\sum_{m} f[n-m]*g[m] \qquad (3)$$

Where, *h[n]* is convolved output of functions *f* and *g*. The coefficients obtained from low pass filter are known as approximation coefficients and those obtained from high pass filter are known as detail coefficients.

The transform follows a one sided tree like structure as shown in Figure 1. In the figure, X refers to the pre-processed signal; cA_i and cD_i are the approximation and detail coefficients respectively at i^{th} level. Only the low pass filtered coefficients are decomposed further to get new set of approximation and detail coefficients. Daubechies-4 (db4) decomposition filter has been used for our model. Db4 low pass and high pass filter consists of 8 coefficients each as shown in Table II.

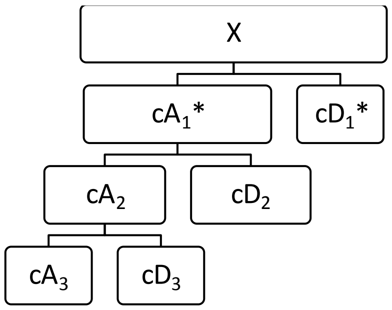
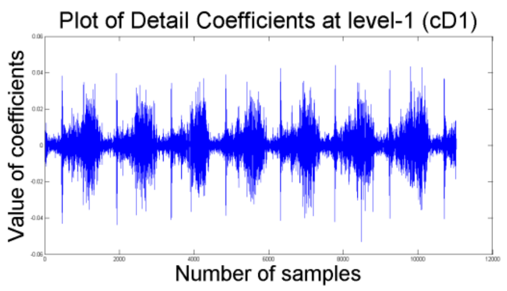
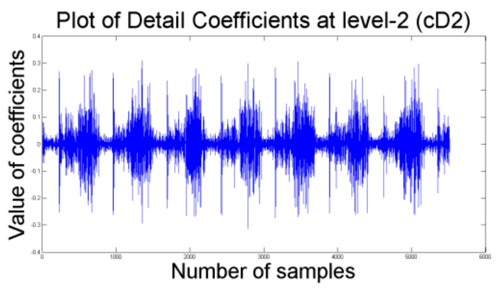


Figure 1. DWT Tree (Level 3 Decomposition)
*cA$_i$: i^th Approximation coefficient *cD$_i$: i^th Detail coefficient

Table II. Daubechies Coefficients [11]

S.No.	**Daubechies 4 Low Pass Filter Coefficients**	**Daubechies 4 High Pass Filter Coefficients**
0	-0.0105974018	-0.2303778133
1	0.0328830117	0.7148465706
2	0.0308413818	-0.6308807679
3	-0.1870348117	-0.0279837694
4	-0.0279837694	0.1870348117
5	0.6308807679	0.0308413818
6	0.7148465706	-0.0328830117
7	0.2303778133	-0.0105974018





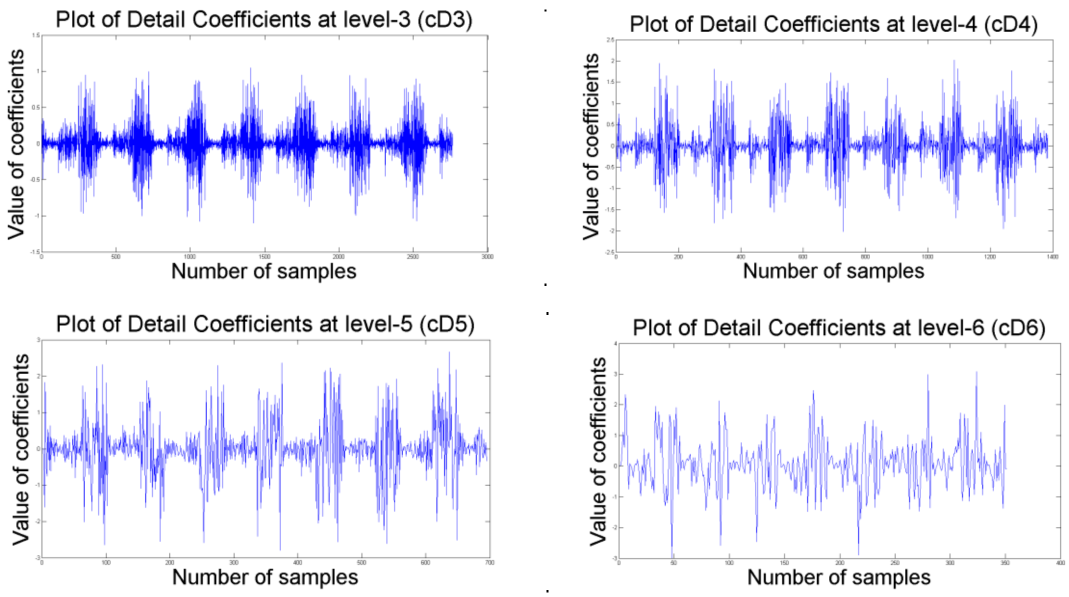


Figure 2: Detail Coefficients at each level

In our model, the signal is decomposed till 6th level which results in six detail coefficients and one approximation coefficient. The initial levels are found to have more abrupt variations as compared to higher levels. The detail coefficients at level 1, 2 and 3 have higher frequency content as compared to the detail coefficients at level 4, 5 and 6.

These observations are useful in the extraction of features from DWT. A total of 9 features are extracted from DWT, of which, 3 are variances of the detail coefficients at level 1, 2 and 3. For finding next 3 features, firstly auto correlation of detail coefficients at level 4, 5 and 6 is performed and then variances of the auto correlation coefficients are calculated. Auto-correlation is a mathematical tool for finding repeated patterns, thus capturing similarity within observations.

$$R_{xx}(l) = \sum_{n=i}^{N-|k|-l} x[n] * x[n-l] \qquad (4)$$

Where, l is lag and R_{xx} is the autocorrelation.

Three more features were found by taking the absolute mean of smoothened versions of detail coefficients at level 1, 2 and 3. The moving averaging filter is used for smoothing the rapid fluctuations of the detail coefficients:

$$y(k) = \frac{\sum_{l=k-m}^{k+m} |x(l)|}{2m} \qquad (5)$$

Where, m decides the degree of smoothness which was taken as 5 in our case.

iii) Wavelet Packet Transform (WPT): One of the drawbacks of Discrete Wavelet Transform is its poor frequency resolution in high frequency region making it difficult to discriminate between signals having high-frequency content, using DWT. WPT gives alternate bases that are formed by taking linear combination of the usual wavelet functions [8]. Unlike DWT, WPT decomposes both low pass and high pass results. Therefore WPT follows a balanced binary tree like structure as compared to the one-sided tree like structure in DWT. The binary tree structure for WPT is shown in Figure 2.

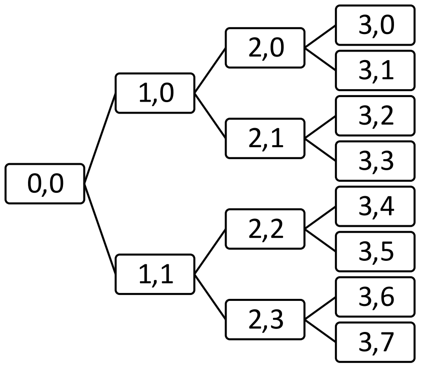


Figure 3: WPT Tree (Level 3 Decomposition)

The level of decomposition in case of WPT was 7. Therefore total numbers of nodes obtained are 254 and the features were the individual node energy. Hence 254 features were extracted from WPT.

So if we combine all the above features, 8 from time domain, 8 from frequency domain and 270 from time-frequency domain; we will have 286 features in total.

B. Feature Selection:

Feature selection is the process of selecting most relevant features from the given set of features. This process is commonly used in data mining applications to improve model performance by alleviating the curse of dimensionality and speeding up the learning process. We used Principal Component Analysis (PCA) [12][13] for selecting the best set of features from the given features.

PCA is a simple and effective means of feature selection. It basically converts the feature space to an uncorrelated basis functions space. These basis functions are found from the eigenvectors of the co-variance matrix of input dataset. These eigenvectors form the basis functions of the new space. These are also known as principal components, which are uncorrelated. The eigenvalues indicate the variance in the direction of the eigenvectors and so the principal components are ranked on the basis of these eigenvalues. This means that greatest variance of the data lies in the direction of first eigenvectors (called the first principal component), similarly the second greatest variance lies in the direction of second eigenvector, and so on. The input data is thus transformed to the new space based on the required number of features. Our goal is that given the number of features required, we try to get a space that is uncorrelated and which maintains most of the variance. The number of top ranked features (in the new space) is generally decided according to the percentage of cumulative energy required.

C. Classification with Support Vector Machines

Support Vector Machines (SVM) is a very popular supervised learning method used for classification. SVM is a binary classifier. The fundamental idea behind this method is to find an optimal hyperplane which maximizes the minimum margin between the two classes, which is only dependent on the border samples known as support vectors. Once we have the support vectors, rest of the feature set is discarded, since the vectors contain all the necessary information for the classifier. The boundary is expressed as,

$$(\boldsymbol{w}.\boldsymbol{x}) + b = 0, \qquad \boldsymbol{w} \in R^N, \qquad b \in R, \tag{6}$$

where the vector $\boldsymbol{w}$ defines the boundary, $\boldsymbol{x}$ is the input vector of dimension N and b is a scalar threshold. We have the decision function as given by equation (7),

$$f(x) = sign\big((\boldsymbol{w}.\boldsymbol{x}) + b\big) \tag{7}$$

Finding of the optimal hyper plane corresponds to solving of the optimization problem given in equation (8).

$$\left.\begin{array}{l} \min_{\xi,w,b} < w.w > + C\sum_{i=1}^{l}\xi_i \\ \textit{subject to} \quad y_i(< w.x_i > +b) \geq 1-\xi_i, \quad i=1,.....,l \end{array}\right\} \quad (8)$$

As it is a convex problem, it gives global and unique solution. By applying kernel tricks, it can be extended to non-linear classification. We have used radial basis function as kernel function with appropriate gamma parameter value. The standard SVM was designed for binary classification, but it can be applied for multi-class classification as well. This is made possible by using decomposition algorithms like one against one, one against all techniques where the multi class classification problem is divided into multiple binary classification problems [14] [15][16].

SVM claims to have good generalization properties as compared to conventional classifiers. This is because maximal margin assures lower VC dimension which further assures better generalization. As compared to the traditional classifiers which are usually trained to minimize the empirical risk, SVM training focuses on minimizing the structural risk [14][17]. SVM is therefore popular for giving a good generalized classifier.

III. CASE STUDY

The entire fault recognition model as presented in Section II, which includes data acquisition, feature extraction, feature selection and classification was implemented as an Android based application. The application was tested on a smart-phone with 830MHz ARMv6 processor, 290MB RAM and Android OS v2.3 as the operating system. Better results w.r.t. speed should be expected on the higher end smart-phones. The air compressor present in the Department of Electrical Engineering, IIT Kanpur, was tested with the application for its running condition - whether they were healthy or faulty. The air compressor used had the following specifications:

- Air pressure range-0-500 lb/m2, 0-35 Kg/cm2
- Induction Motor: 5 HP, 415V, 5Amp, 3ph, 50 Hz, 1440 rpm
- Pressure Switch: Type PR-15, Range: 100-213 PSI, 7-15 Kg/cm2

Audio recordings were done by the smartphone at 44.1 kHz sampling rate for 4 different running conditions of air compressors. The dataset consisted of 4 classes associated with 4 conditions with 200 instances per class, amounting to a total of 800 instances. The four different conditions of compressor were:

a) Healthy condition: As the name suggests, the compressor is healthy.
b) LIV faulty condition: It occurs due to damaged inlet valve of the reciprocating air compressor. Strong pressure is generated as air is compressed by piston action of the reciprocating air compressor. Damaged valve will partially allow the compressed air to leak from inlet valve, which then results in pressure oscillations.
c) LOV faulty condition: It occurs when outlet valve is damaged. This results in increased time consumption for the filling of air tank at required pressure level.
d) NRV faulty condition: It occurs when the non-return valve is damaged.

For ease of implementation of required mathematical functions in Java, Commons Math Package by Apache Foundation [18] is used. This java library consists of various methods for numerical analysis, used in engineering and math.

A. Data Acquisition:

The goal of data acquisition is to get raw sampled data of the acoustic signal. The acoustic signal is acquired from the most sensitive position on the compressor [19][20] in `*wav*` format. Here sensitive positions refers to the set of sensor positions on the machine which exhibit appropriate fault characteristics in form of acoustics in a much better way than other positions. The '*wav*' format was chosen for its simplicity, as it is an uncompressed format that stores the data in 16 bit linear PCM format in 2's compliment. With appropriate code using Android classes namely '*Audio Recorder class*` and `*WavFile class*`, we could get the sampled data from the '*wav*` file for further processing.

B. Pre-processing:

Data pre-processing is important for improving the reliability of data. We used a pre-existing module built at our lab, which worked on the sampled data with the main aim of removal of noise and effect of outliers. Following steps were followed in the module:

1. Down sampling: The sampling rate of the data was reduced by a factor of 2. This reduced the data size to more manageable value. This was done because high sampling rate is not required as much of the information content was found in frequencies below 12 kHz.

2. Clipping: This was performed to get a more stable signal from the 5 second signal. This was done by finding and selecting the 1 second window with minimum standard deviation. An added advantage of the same is that the data used for further computation is also reduced.

3. Filtering: The data was then filtered using a Butterworth low pass filter of 12 kHz cut off frequency to remove high frequency noise. Finally the data was scaled between 0 and 1.

C. Feature Extraction

In the time domain, the aforementioned eight statistical parameters were calculated. The algorithms used for calculation of these variables were compatible with the algorithms used in MATLAB. This allowed us to quickly check and compare our results with the corresponding built-in MATLAB functions.

For frequency domain features, Discrete Fourier Transform (DFT) is calculated using the 'Fast Fourier Transform' class in Apache Commons Math Library. The DFT is implemented in the library using Cooley-Tukey FFT (Fast Fourier Transform) algorithm. It is a divide and conquer algorithm that recursively breaks down DFT problem into smaller DFT problems. 256 point DFT is calculated and the transformed signal is divided into 8 bins of 32 samples each. Finally, ratio of individual bin energy and total energy were taken to be frequency domain features.

For wavelet domain, we extracted seven features from the Morlet wavelet analysis as mentioned in Section III, nine features from the DWT analysis and 254 features from WPT.

For DWT features, autocorrelation function is required to calculate certain features. The algorithm used for implementing autocorrelation is based on Wiener-Khinchin theorem because it is faster than the usual brute force method based on the definition of autocorrelation (i.e. going by the expression of autocorrelation). Time complexity for brute force method is $O(n^2)$ and Wiener-Khinchin theorem improves it to $O(n\,log(n))$. The steps involved in Wiener-Khinchin theorem are as follows:

1. FFT of the given data.

$$F[k] = \sum_{n=0}^{N-1} x[n] \times e^{-i2\pi\frac{k}{N}n} \qquad (9)$$

2. Multiply the result obtained in Step 1 with its conjugate.

$$S[n] = F[n] * \overline{\overline{F}[n]} \qquad (10)$$

3. Take Inverse FFT of the result obtained in Step 2.

$$R[n] = \frac{1}{N}\sum_{k=0}^{N-1} S[k] \times e^{i2\pi\frac{k}{N}n} \qquad (11)$$

At each level of decomposition, convolution of the signal with the filters is followed by dyadic decimation (or down-sampling). In our implementation, down sampling was done by taking every even indexed elements like x[0], x[2], x[4] etc. and leaving odd indexed elements. The basic purpose of down sampling in DWT is to reduce the signal size at each level. The size of data is a critical issue as it would cause the heap size memory error.

In case of WPT, we decompose signal up to level 7 because decomposition after 7th level would cause redundancy. Moreover higher decomposition level would have created java heap size error. We used the Daubechies-4 filter for analysis which is same as the one used for DWT. Decomposition level of 7 gave 127 packets each of approximation and detail coefficients. Here packets refer to nodes in the binary tree structure for WPT. The features extracted from WPT were the individual energies of 254 packets. Expression for energy is as follows:

$$E_i = \sum_{n_i} x[n]^2 \qquad (12)$$

Where, E_i is energy at ith node and n_i is number of points belonging to the ith node.

D. Feature Selection

As mentioned earlier, PCA was applied to the 286 features extracted from the recorded wav files. 640 samples out of the total 800 were taken in the training set where each sample corresponds to one wav-file recording. Though the application can be programmed to select any number of features required, we decided to work on selecting 23 feature set from the data set of 286 features. Previous experiments showed that the 23 features selected contained more than 99.9% energy

[19]. The final selected features are written in comma separated format along with the eigen matrix, scaling factors and mean values to their respective files on SD-card. This was done because same parameters will be used while testing. The results are shown in Table IV.

E. Feature Classification:

For SVM classification, the LIB-SVM java library was used, which implements an SMO type algorithm for solving the SVM's dual optimization problem [21].

The SVM module works in the following modes:

- Parameter Search: In this mode, the application cross validates for finding the best possible pair of cost and gamma parameters. The application also gives an option where user can manually enter the parameter values.
- Training: In this mode, the application builds an SVM model for later prediction.
- Classification: In this mode, the application uses the stored classifier model for predicting the classes of unknown test data and writes the result to a file.

IV. RESULTS

The user interface for our application consists of four buttons as shown in Figure 3. The buttons "Extract Features", "Select Features", "Search Best Parameters" and "Train" when clicked, performs Feature Extraction, Feature Selection, search for the best Cost and Gamma parameters and building of the trained model respectively.

The processing time for reading a single '*.wav*' recording and extracting features are given in Table III. The processing times for building models for PCA (Feature Selection), Parameter Search and SVM classifier (Classifier model) are given in Table IV. Though the timings given in table are quite large, this training needs to be done only once. While performing acoustic recognition on real time, the testing time for a single recording was 58 seconds. The training times can be improved by working on smart phones with better processing capabilities. All the results were cross-validated with the corresponding pre-built modules made in our lab on MATLAB and results were found to be identical.

Table III. Feature Extraction Processing Time

File	Size	Time
Healthy/Reading1.wav	1.12 MB	12 s
Lov/Reading1.wav	1.14 MB	12 s
Liv/Reading1.wav	1.14 MB	12 s
Nrv/Reading1.wav	1.14 MB	12 s
Healthy/Reading2.wav	1.14 MB	12 s
Lov/Reading1.wav	1.14 MB	12 s
Liv/Reading2.wav	1.14 MB	12 s
Nrv/Reading2.wav	1.14 MB	12 s

Table IV. Model Building Details

Operation Performed	Number of Samples	Number of Features	Time (approx..)
Feature Selection	800	286	355 s
Parameter Search (RBF kernel)	800	23	8140 s
Classifier model building	800	23	53 s

For classification purposes, we worked on data after feature selection i.e. samples having 23 features. Further details of the classification module are as follows:

1. Parameter Search: Using the LIBSVM's train function, we built a cross validation module which checked for best Cost and Gamma parameters (for RBF kernel) for training. Initially the range for the parameter values was taken to be 2^{-9} to 2^{9}. Finding the best parameter values for RBF kernel takes very long time. Therefore it is recommended to find best parameter values externally on a desktop or else one can find the same on smartphone for a smaller range of parameter values. A better option could be to use other kernel functions like linear, quadratic and polynomial, as in such cases, finding best parameter value is easily manageable on smartphone. Still considering the fact that we are having smartphones with much better processors and will have even better in the future, finding best parameter values for RBF kernel is quite feasible.

2. Training: After selecting 23 features by PCA, classifier model was built using the LIB-SVM's algorithm. This model was stored in SD card as a model file for prediction.

3. Testing: For a test wav file, the file is read, pre-processed and features are extracted to a feature vector. The feature vector is mean-centered using the mean values calculated during training. After mean-centering, it is multiplied by the transform matrix generated from PCA. The transformed matrix, which has 23 features, is passed to classifier (LIBSVM) module which performs classification using the already stored model.

We performed cross validation on the 800 sample database with 640 samples for training and the remaining 160 samples for testing. The classification process was checked for linear, polynomial and rbf kernels. The accuracy results for different kernels are shown in Table V, VI and VII. The results were obtained by comparing one class with the remaining classes. For example, the accuracy results for healthy state would be obtained by checking how accurately our system is able to detect if the state is healthy or not. The same is done for LOV, LIV and NRV states as well. We do this, because multi-class SVM classifier would classify a given test sample to any one of the classes only. In real world scenario, it is quite possible that at a given time there might be more than one fault in the system. By performing binary classification i.e. checking presence of individual state, we can have the test sample belonging to more than one state. As can be seen from the results tables, the accuracy results for recognizing the different machine states are very encouraging.

Table V. Accuracy Results using Linear Kernel

Cross-validation Data Set	**Healthy Accuracy**	**LOV Accuracy**	**LIV Accuracy**	**NRV Accuracy**
1	83.12%	95%	100%	91.88%
2	75%	75%	100%	85.62%
3	93.12%	75%	100%	83.12%
4	88.75%%	95.625%	99.37%	75%
5	75%	75%	75%	75%
Average Accuracy :	**83%**	**83.12%**	**94.87%**	**82.12%**

Table VI. Accuracy Results using Polynomial Kernel

Cross-validation Data Set	**Healthy Accuracy**	**LOV Accuracy**	**LIV Accuracy**	**NRV Accuracy**
1	91.25%	93.75%	100%	89.38%
2	92.5%	96.88%	100%	90%
3	90%	93.12%	100%	93.75%
4	90.62%%	93.75%	100%	91.88%
5	90%	93.12%	99.37%	89.38%
Average Accuracy :	**90.88%**	**94.12%**	**99.88%**	**90.88%**

Table VII. Accuracy Results of Application (RBF Kernel)

Cross-validation Data Set	**Healthy Accuracy**	**LOV Accuracy**	**LIV Accuracy**	**NRV Accuracy**
1	95%	96%	100%	94%
2	98.25%	99.5%	100%	96%
3	98.33%	99.33%	99.83%	96.33%
4	97%	99.33%	99.67%	96.33%
5	98.25%	99.5%	99.87%	96.62%
Average Accuracy :	**97.37%**	**98.73%**	**99.87%**	**95.86%**

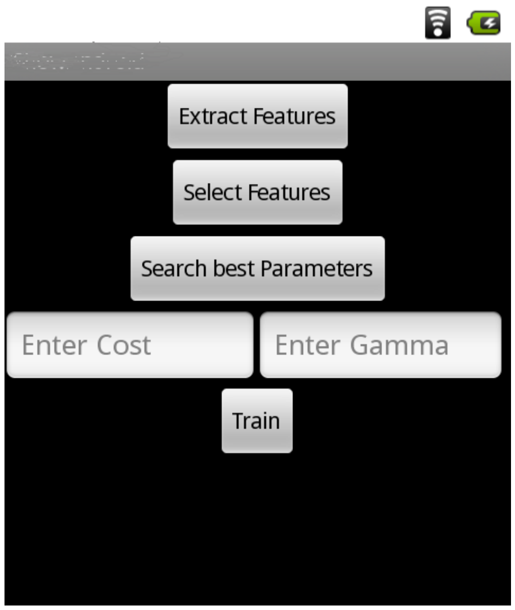


Figure 4. Model Building Interface

V. CONCLUSION

This paper described an Android implementation of a general purpose audio data-mining application using Apache-Commons Math and LIBSVM libraries. The application running on smartphone is complete in itself, as it gives facility to perform all operations i.e. from data acquisition to training of classifier model. By using RBF kernel while training, we are able to get accuracies of 97.37%, 98.73%, 99.87% and 95.86% for Healthy, LOV, LIV and NRV states of air compressor respectively. As RBF kernel gave better results as compared to other kernels, it is used in the final implementation of our application. We feel that this application with some minor changes in specifications can be extended and used for acoustic pattern recognition in many other applications as well. For future work, PCA can be implemented on the native layer of Android to speed up the process of analysis. Also we would like to use check feature selection algorithms like Independent Component Analysis and algorithms based on Mutual Information amongst features, and compare their performance accordingly.

ACKNOWLEDGEMENT

We would like to express or sincere gratitude towards Boeing for their funding support for this research work through research grant (BOEING/EE/20100220), at IIT Kanpur.

REFERENCES

[1] Li, Weilin, Pan Fu, Weiqing Cao, "Study on Feature Selection and Identification Method of Tool Wear States Based on SVM," *International Journal on Smart Sensing and Intelligent Systems*, vol. 6, no. 2, pp. 448-465, April 2013.

[2] Lane, D. Nicholas, et al. "BeWell: A smartphone application to monitor, model and promote wellbeing," *5th International Conference on Pervasive Computing Technologies for Healthcare* (Pervasive Health 2011), 2011.

[3] Frank Sposaro, Justin Danielson, Gary Tyson. "iWander: an Android application for dementia patients," *In Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pp. 3875-3878, 2010.

[4] N.K. Verma, S. Singh, J.K. Gupta, R.K. Sevakula, S. Dixit, A. Salour, "Smartphone application for fault recognition," *Sixth International Conference on Sensing Technology (ICST),* 18-21 Dec. 2012

[5] Andrew K.S. Jardine, Daming Lin, Dragan Banjevic. "A review on machinery diagnostics and prognostics implementing condition-based maintenance," *Mechanical Systems and Signal Processing,* vol. 20, no. 7, pp. 1483-1510, 2006.

[6] Qingbo He, Fanrang Kong, Ruqiang Yan. "Subspace-based gearbox condition monitoring by kernel principal component analysis," *Mechanical Systems and Signal Processing,* vol. 21, no. 4, pp. 1755-1772, May 2007.

[7] Qingbo He, Ruqiang Yan, Fanrang Kong, Ruxu Du, "Machine condition monitoring using principal component representations," *Mechanical Systems and Signal Processing,* vol. 23, no. 2, pp. 446-466, February 2009.

[8] B. Liu, S.F. Ling, Q.F. Meng, "Machinery diagnostics based on wavelet packets," *Journal of Vibration and Control*, vol. 3, no. 1, pp. 5–17, 1997.

[9] J. Ming, "Feature Extraction based on Morlet wavelet and its application for mechanical fault diagnosis," *Journal of Sound and Vibration*, vol. 234, no. 1, pp. 135-148, 2000.

[10] S.K. Goumas, M.E. Zervakis, G.S. Stavrakakis, "Classification of washing machines vibration signals using discrete wavelet analysis for feature extraction," *IEEE Transactions on Instrumentation and Measurement,* vol. 51, no. 3, pp. 497-508, 2002.

[11] Ingrid Daubechies, "Ten lectures on wavelets," *Philadelphia: Society for industrial and applied mathematics,* vol. 61, 1992.

[12] I.T. Jolliffe, "Principal component analysis," *Spring-verlag, New York*, 1986.

[13] S. Wold, K. Esbensen, P. Geladi. "Principal component analysis," *Chemometrics and intelligent laboratory systems,* vol. 2, no. 1, pp. 37-52, 1987.

[14] A. Widodo, Bo-Suk Yang, "Support vector machine in machine condition monitoring and fault diagnosis," *Mechanical Systems and Signal Processing,* vol. 21, no. 6, pp. 2560–2574, August 2007.

[15] A. Mathur, G.M. Foody, "Multiclass and Binary SVM Classification: Implications for Training and Classification Users," *Geoscience and Remote Sensing Letters, IEEE,* vol. 5, no. 2, pp. 241-245, April 2008.

[16] N.K. Verma, A. Roy, A. Salour, "An optimized fault diagnosis method for reciprocating air compressors based on SVM," *IEEE International Conference on System Engineering and Technology (ICSET),* pp. 65-69, 27-28 June 2011.

[17] C. Cortes, V. Vapnik, "Support-vector networks," *Machine Learning,* vol. 20, no. 3, pp. 273-297, 1995.

[18] Commons Math Developers, Apache Commons Math, Release 2.2
Library available at: http://commons.apache.org/math

[19] N.K. Verma, K. Jagannatham, A. Bahirat, T. Shukla, "Finding Sensitive Sensor positions under faulty condition of Reciprocating Air Compressors," *IEEE International Conference on Recent Advances in Intelligent Computational Systems*, pp. 242-246, Sep. 22-24, 2011.

[20] N.K. Verma, K. Piyush, R.K. Sevakula, S. Dixit, A. Salour. "Ranking of sensitive positions based on statistical parameters and cross correlation analysis," *Sixth International Conference on Sensing Technology (ICST),* pp. 815-821, 18-21 Dec. 2012.

[21] C. Chang, C. Lin, "LIBSVM : a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology,* vol. 2, no. 3, pp. 27:1-27:27, 2011.
Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.