**Introduction**

We present here a simple method/strategy for bird’s song and call identification. It builds on known and efficient Technologies. The method presented here relies particularly on the fact that training and test input signals are mono label i.e. only one species may be heard at one time. Currently, capturing and monitoring of bird species through marking individuals with radio monitoring devices or visual tags is sometimes necessary for bio-surveys. The survival and successful reproduction of these captured individuals may be affected. Using bioacoustics may be one way in which to expand conservation efforts and shun this type of handling. Using sound to identify certain species is not a new idea. T. A. Parker recorded the dawn choruses of bird in the Bolivian Amazon, and within 7 days he captured 85% of the regional species on tape. In that same region, seven experienced ornithologists took 54 days to inventory the birds using a capture and release technique. The basis of the project is to study and use bio-acoustical recordings from birds to create a machine learning algorithm that, taking an audio stream as input, will be used as an automated species recognition tool. Automated recognition of bioacoustics signals has been reported with encouraging results in a variety of animals including bats, birds, frogs and Orthoptera (grasshoppers, crickets and locusts).Birds species recognition in particular has been studied using support vector machines, sinusoidal modeling, hidden Markov models, and dynamic time wrapping.

We are training our machine using some provided audio file (.wav) i.e. by fetching some specific features from audio, so that machine can be trained on that basis and then we can test it by providing data or audios through test data.

**Motivation**

Actually the work done in this field is quite less and also machine learning has to be implemented for the voice recognition. So this unique idea i.e. implementing machine learning using octave for voice recognition gave us an immense energy and motivation to work on this idea.

Also bird call recognition helps in finding number of bird species in an area and types of bird species without accessing each and every bird that is quite an irrelevant approach.

**Problem statement**

Given an audio wave file (.wav) as input, predict the bird species (which are available in training set) this audio belongs to.

**Analysis of Previous Similar Works**

1. **Acoustic Monitoring of Night-Migrating Birds [1]:** This paper discusses an emerging methodology that uses electronic technology to monitor vocalizations of night-migrating birds. On a good migration night in eastern North America, thousands of call notes may be recorded from a single ground-based, audio-recording station, and an array of recording stations across a region may serve as a "recording net" to monitor a broad front of migration. Night-flight calls of 35 species of migrant land birds have been identified by spectrographic matching with diurnal calls recorded from known-identity individuals; call types of another 31 species are known, but are not yet distinguishable from other similar calls in several species complexes. Efforts to use signal-processing technology to automate the recording, detection, and identification of night-flight calls are currently under way at the Cornell Lab of Ornithology. Most species of North America's migrant land birds make their transcontinental flights at night, and many species give short vocalizations while they fly. By aiming microphones at the night sky, a volume of sky-with dimensions dependent on microphone design-can be monitored for calls (Graber and Cochran 1959). A variety of microphone and recording station designs have been used for this purpose, depending on the specific monitoring goals and the recording environment (Graber and Cochran 1959, Dierschke 1989, Evans 1994). In many regions of North America, a recording station may detect thousands of calls during a single migration night (Graber and Cochran1960, Evans 1994). Species known to give night-flight calls include the warblers (Parulinae), sparrows (Emberizinae), cuckoos (Cuculidae), rails (Rallidae), herons (Ardeidae), and Catharus thrushes. Groups not known to give regular vocalizations in night migration are the vireos (Vireonidae), flycatchers (Tyrannidae), and orioles (Icterinae). If a monitoring protocol is consistently maintained, an array of microphone stations can provide information on how the species composition and number of vocal migrants vary across time and space. Such data have application for monitoring avian populations and identifying their migration routes. In addition, detection and classification of distinctive call-types is possible with computers (Mills 1995, Taylor 1995), thus information on bird populations might be gained automatically. In this paper they summarize the current state of knowledge for identifying night-flight calls to species; present Selected results from four ongoing studies that are monitoring night-flight calls; and discuss the implications of this research for conservation of migratory land birds.

1. **Automatic identification of bird calls [2]**: Automatic identification of bird calls without manual intervention has been a challenging task for meaningful research on the taxonomy and monitoring of bird migrations in ornithology. In this paper they have applied several techniques used in speech recognition to the automatic identification of bird calls. A new technique which computes the ensemble average on the FFT spectrum is proposed for identification of bird calls. This ensemble average is computed on the FFT spectrum of each bird and is called the Spectral Ensemble Average Voice Print (SEAV) of that particular bird. The SEAV of various birds are computed and are found to be different when compared to each other. A database of bird calls is created from the available recordings of fifteen bird species. The SEAV is then used for the identification of bird calls from this database. The results of identification using SEAV are then compared against the results derived from common classifiers used in speech recognition like dynamic time warping (DTW), Gaussian mixture modeling (GMM). A one level and two level classifier combination is also tried by combining SEAV classifier with the DTW classifier. The SEAV is computationally less expensive when compared to DTW or the GMM based classifiers while performing better than the DTW technique. Several new possibilities in automatic bird call identification using SEAV are also listed.

**Pre-Requisites**

1. **Background**
2. **Bioacoustics**

Bioacoustics pertain to the sounds that animals make and can often provide insight to their behavior. Bioacoustics research and tools can aid in monitoring and managing species, which is vital to the conservation and preservation of diversity. Currently, capturing and monitoring of bird species through marking individuals with radio monitoring devices or visual tags is sometimes necessary for bio-surveys. The survival and successful reproduction of these captured individuals may be affected. Using bioacoustics may be one way in which to expand conservation efforts and shun this type of handling. Using sound to identify certain species is not a new idea. T. A. Parker recorded the dawn choruses of bird in the Bolivian Amazon, and within 7 days he found he had captured 85% of the regional species on tape. In that same region, seven experienced ornithologists took 54 days to inventory the birds using a capture and release technique. The basis of the project is to study and use bioacoustical recordings from birds to create a machine learning algorithm that, taking in an audio stream, will be used as an automated species recognition tool. Automated recognition of bioacoustics signals has been reported with encouraging results in a variety of animals including bats, birds, frogs and Orthoptera (grasshoppers, crickets and locusts). Birds species recognition in particular has been studied using support vector machines, sinusoidal modeling, hidden Markov models, and dynamic time warping .

1. **Spectrograms**

Spectrograms are graphical representations of audio files, with time on the horizontal axis and frequency (going from low frequency at the top of the image and high at the bottom) on the vertical-axis. For each of the audio files, we divide the sound of the signal into frames and compute the spectrum for each of the frames. A spectrum represents the intensity of the signal as a function of frequency. The spectrogram is then created as a graph of the spectra of each frame in the sound. Then for each of the frames, the fast-Fourier transform (FFT) complex coefficients c for I = 1,...,m is taken where m = frame-size/2 = 512. One way in which we remove the noise in the lower frequency range is by using triangular band pass filter (TBF).

1. **Mel-Frequency Cepstral Coefficients (MFCCs)**

For speech/Speaker recognition, the most commonly used acoustics are mel-scale frequency cepstral coefficient (MFCC for short). MFCC takes human perception sensitive with respect to frequencies into consideration, and therefore are best for speech/speaker recognition.

Step by step computation of MFCCs is:

**Pre-emphasis:**

The speech signal s(n) is sent to a high-pass filter:

S2(n) = s(n) – a\*s(n-1)

Where s2(n) is the output signal and the value of a is usually between .9 and 1.0.Value of a used by us is **.95**.

The goal of pre-emphasis is to compensate the high-frequency part that was suppressed during sound production mechanism. It can also amplify the importance of high-frequency formats.

**Frame Blocking:**

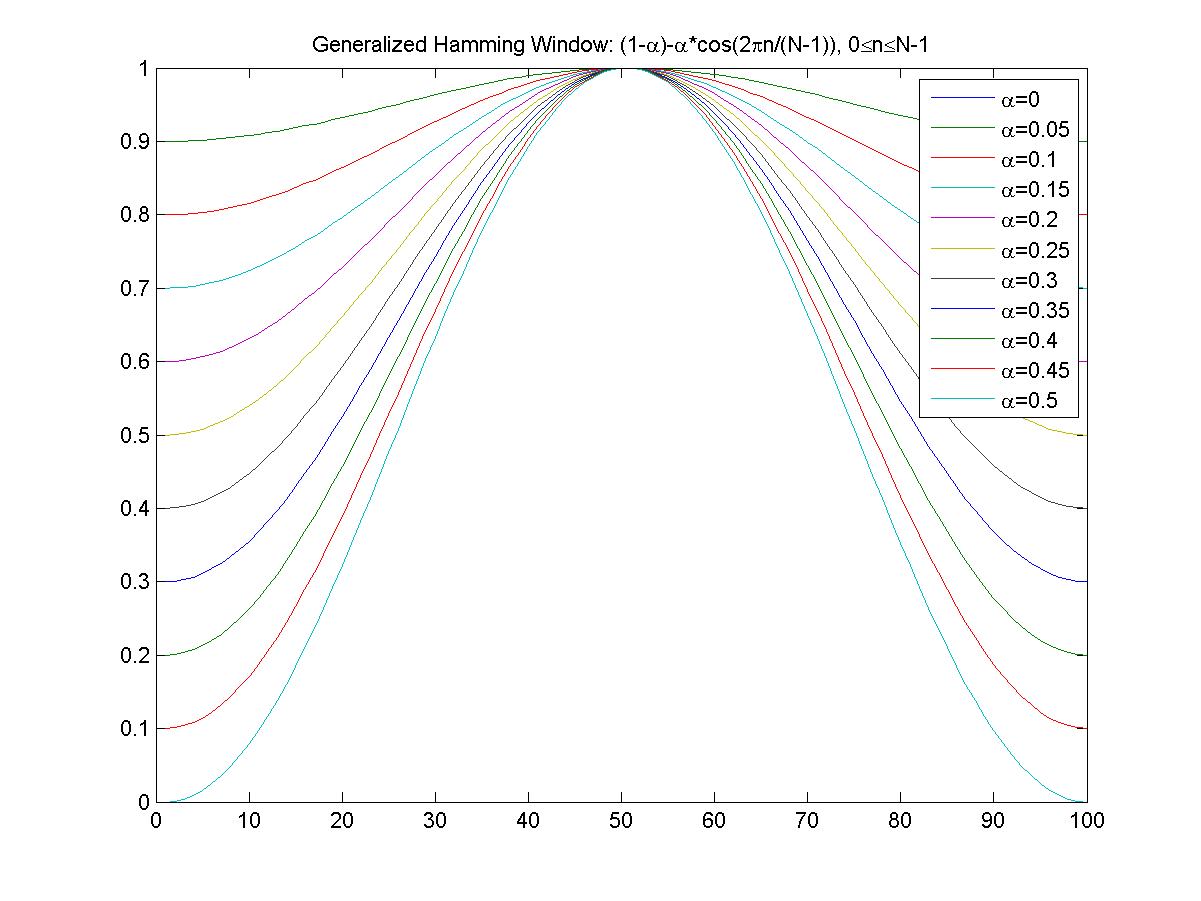
The input speech signal is segmented into frames of 20~30 ms with optional overlap of 1/3~1/2 of the frame size. Usually the frame size (in terms of sample points) is equal to power of two in order to facilitate the use of FFT. If this is not the case, we need to do zero padding to the nearest length of power of two. If the sample rate is 16 kHz and the frame size is 320 sample points, then the frame duration is 320/16000 = 0.02 sec = 20 ms. Additional, if the overlap is 160 points, then the frame rate is 16000/(320-160) = 100 frames per second.

**Hamming Window:**

Each frame has to be multiplied with a hamming window in order to keep the continuity of the first and the last points in the frame. If the signal in a frame is denoted by s(n), n = 0,…N-1, then the signal after Hamming windowing is s(n)\*w(n), where w(n) is the Hamming window defined by:

w(n, a) = (1 - a) - a cos(2pn/(N-1))，0≦n≦N-1

Different values of a corresponds to different curves for the Hamming windows.



**Fast Fourier Transform (FFT):**

Spectral analysis shows that different timbres in speech signals corresponds to different energy distribution over frequencies. Therefore we usually perform FFT to obtain the magnitude frequency response of each frame.

When we perform FFT on a frame, we assume that the signal within a frame is periodic, and continuous when wrapping around. If this is not the case, we can still perform FFT but the incontinuity at the frame's first and last points is likely to introduce undesirable effects in the frequency response. To deal with this problem, we have two strategies:

Multiply each frame by a Hamming window to increase its continuity at the first and last points.

Take a frame of a variable size such that it always contains a integer multiple number of the fundamental periods of the speech signal.

The second strategy encounters difficulty in practice since the identification of the fundamental period is not a trivial problem. Moreover, unvoiced sounds do not have a fundamental period at all. Consequently, we usually adopt the first strategy to mutiply the frame by a Hamming window before performing FFT.

**Triangular Band Pass Filter (TBF):**

We multiple the magnitude frequency response by a set of 20 triangular bandpass filters to get the log energy of each triangular bandpass filter. The positions of these filters are equally spaced along the Mel frequency, which is related to the common linear frequency f by the following equation:

mel(f)=1125\*ln(1+f/700)

Mel-frequency is proportional to the logarithm of the linear frequency, reflecting similar effects in the human's subjective aural perception.

The reasons for using triangular band pass filters are twofold:

Smooth the magnitude spectrum such that the harmonics are flattened in order to obtain the envelop of the spectrum with harmonics. This indicates that the pitch of a speech signal is generally not presented in MFCC. As a result, a speech recognition system will behave more or less the same when the input utterances are of the same timbre but with different tones/pitch.

Reduce the size of the features involved.

**Discrete Cosine Transform (DCT):**

In this step, we apply DCT on the 20 log energy Ek obtained from the triangular band pass filters to have L Mel-scale cepstral coefficients. The formula for DCT is:

Cm=Sk=1Ncos[m\*(k-0.5)\*p/N]\*Ek, m=1,2, ..., L

Where N is the number of triangular band pass filters, L is the number of Mel-scale cepstral coefficients. Usually we set N=20 and L=12. Since we have performed FFT, DCT transforms the frequency domain into a time-like domain called frequency domain. The obtained features are similar to cepstrum, thus it is referred to as the Mel-scale cepstral coefficients, or MFCC. MFCC alone can be used as the feature for speech recognition. For better performance, we can add the log energy and perform delta operation.

1. **Support Vector Machine (SVM):**

In machine learning, support vector machines (SVMs, also support vector network) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples into one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

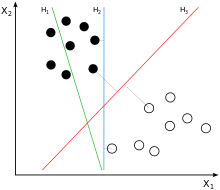
In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature space.

More formally, a support vector machine constructs a hyperplane  or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.

Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function k(x,y) selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters ‘αi’ of images of feature vectors that occur in the data base. With this choice of a hyperplane, the points ‘x’ in the feature space that are mapped into the hyperplane are defined by the relation: ∑iαi k(xi,x)=constant. Note that if k(x,y)  becomes small as ‘y’  grows further away from ‘x’ , each term in the sum measures the degree of closeness of the test point ‘x’ to the corresponding data base point xi . In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points ‘x’ mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets which are not convex at all in the original space.

**SVM Motivation:**

Classifying data is a common task in machine learning. Suppose some given data points each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support vector machines, a data point is viewed as a p-dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a (p − 1)-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum margin classifier; or equivalently, the perceptron of optimal stability.

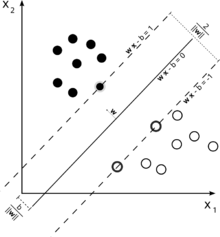


**Linear SVM**

Given some training data \mathcal{D}, a set of n points of the form

\mathcal{D} = \left\{ (\mathbf{x}_i, y_i)\mid\mathbf{x}_i \in \mathbb{R}^p,\, y_i \in \{-1,1\}\right\}_{i=1}^n

where the yi is either 1 or −1, indicating the class to which the point \mathbf{x}_i  belongs. Each  \mathbf{x}_i  is a p-dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having y_i=1 from those having y_i=-1. Any hyperplane can be written as the set of points \mathbf{x} satisfying.

[](http://en.wikipedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png)

Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

\mathbf{w}\cdot\mathbf{x} - b=0,\,

where \cdot denotes the dot product and {\mathbf{w}} the (not necessarily normalized) normal vector to the hyperplane. The parameter \tfrac{b}{\|\mathbf{w}\|} determines the offset of the hyperplane from the origin along the normal vector {\mathbf{w}}.

If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. The region bounded by them is called "the margin". These hyperplanes can be described by the equations

\mathbf{w}\cdot\mathbf{x} - b=1\,

and

\mathbf{w}\cdot\mathbf{x} - b=-1.\,

By using geometry, we find the distance between these two hyperplanes is 2/\|\mathbf{w}\|, so margin, we add the following constraint: for each i either

\mathbf{w}\cdot\mathbf{x}_i - b \ge 1\qquad\text{ for }\mathbf{x}_i  of the first class

or

\mathbf{w}\cdot\mathbf{x}_i - b \le -1\qquad\text{ for }\mathbf{x}_i  of the second.

This can be rewritten as:

y_i(\mathbf{w}\cdot\mathbf{x}_i - b) \ge 1, \quad \text{ for all } 1 \le i \le n.\qquad\qquad(1)

We can put this together to get the optimization problem:

Minimize (in {\mathbf{w},b})

\|\mathbf{w}\|

subject to (for any i = 1, \dots, n)

y_i(\mathbf{w}\cdot\mathbf{x_i} - b) \ge 1. \, 

1. **GNU Octave**

**GNU Octave** is a high-level programming language, primarily intended for numerical computations. It provides a command-line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with MATLAB. It may also be used as a batch-oriented language.

The Octave language is an interpreted programming language. It is a structured programming language (similar to C) and supports many common C standard library functions, and also certain UNIX system calls and functions. However, it does not support passing arguments by reference.

Octave programs consist of a list of function calls or a script. The syntax is matrix-based and provides various functions for matrix operations. It supports various data structures and allows object-oriented programming.

Its syntax is very similar to MATLAB, and careful programming of a script will allow it to run on both Octave and MATLAB.

1. **C (Programming language)**

C is an imperative (procedural) language. It was designed to be compiled using a relatively straightforward compiler, to provide low-level access to memory, to provide language constructs that map efficiently to machine instructions, and to require minimal run-time support. C was therefore useful for many applications that had formerly been coded in assembly language, such as in system programming.

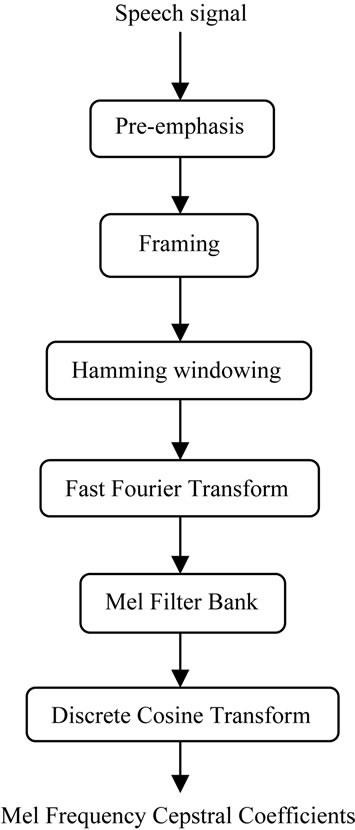
Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A standards-compliant and portably written C program can be compiled for a very wide variety of computer platforms and operating systems with few changes to its source code. The language has become available on a very wide range of platforms, from embedded microcontrollers to supercomputers.

**Proposed Methodology of Implementation**

We want to learn a multi label classifier from a set of N mono labeled training samples where each input is an audio recording. The system should be able to learn and eventually be able to predict the bird species in an audio.

**1. Pre-processing:**

Our preprocessing is based on MFCC cepstral coefficients, which have been proved useful for speech recognition. A signal is first transformed into series of frames where each frame consists of 13 MFCC (mel-frequency cepstral coefficients) feature vectors, including energy. Each frame represents a short duration of around 20ms.



**MFCCs extraction steps**

1. **Pre-emphasis :**

The speech signal s(n) is sent to a high-pass filter:

S2(n) = s(n) – a\*s(n-1)

Where s2(n) is the output signal and the value of a is usually between .9 and

1. Value of a used by us is **.95**.

*Code:*

*waveFile='bird.wav';*

*[y, fs]=wavread(waveFile);*

*a=0.95;*

*y2 = filter([1, -a], 1, y);*

1. **Frame blocking:**

The input speech signal is segmented into frames of 20~30 ms with optional overlap of 1/3~1/2 of the frame size. Usually the frame size (in terms of sample points) is equal to power of two in order to facilitate the use of FFT.

1. **Hamming Window:**

Each frame has to be multiplied with a hamming window in order to keep the continuity of the first and the last points in the frame (to be detailed in the next step). If the signal in a frame is denoted by s(n), n = 0,…N-1, then the signal after Hamming windowing is s(n)\*w(n).

w(n, a) = (1 - a) - a cos(2pn/(N-1))，0<=n<=N-10

*Code:*

*N=100;*

*n=(0:N-1)';*

*alpha=linspace(0,0.5,11)';*

*h=[];*

*for i=1:length(alpha),*

*h = [h, (1-alpha(i))-alpha(i)\*cos(2\*pi\*n/(N-1))];*

*end*

1. **Fast Fourier Transform:**

Spectral analysis shows that different timbres in speech signals corresponds to different energy distribution over frequencies. Therefore we usually perform FFT to obtain the magnitude frequency response of each frame.

*Code:*

*fs=8000;*

*t=(1:512)'/fs;*

*f=306.396;*

*original=sin(2\*pi\*f\*t)+0.2\*randn(length(t),1);*

*windowed=original.\*hamming(length(t));*

*fftmag = fft(windowed);*

1. **Triangular Band Pass Filter:**

We multiple the magnitude frequency response by a set of 20 triangular bandpass filters to get the log energy of each triangular bandpass filter. The positions of these filters are equally spaced along the Mel frequency, which is related to the common linear frequency f by the following equation:

mel(f)=1125\*ln(1+f/700)

1. **Discrete Cosine Transform (DCT):**

In this step, we apply DCT on the 20 log energy Ek obtained from the triangular bandpass filters to have L mel-scale cepstral coefficients. The formula for DCT is shown next.

Cm=Sk=1Ncos[m\*(k-0.5)\*p/N]\*Ek, m=1,2, ..., L

* 1. **Feature extraction**

We have used rms (root-mean square) values for each frame and took 100 frames with max values. Then this 2-D array of 100 frames each with 13 features was use for training.

*Code:*

*for i = 1:n,*

*vector(i) = power(sum(mfcc(:,i).^2)/m,1/2);*

*end;*

* 1. **Training**

Based on feature extraction step we describe above the simplest strategy to train a classifier on the feature vector of MFCCs is using SVM (Support Vector Machine) to decide which species is present in test signal.

Yet we have a better strategy which is not implemented yet by us. First is to perform a clustering in order to split all samples corresponding to species into 2 different classes. The reason behind this process is that calls of a particular species are completely different so corresponding feature vectors may lie in entire different feature space.

After using this, the problem of recognizing k species changes to classification problem with 2 X k species.

Second one is to change the MFCC features into some better features by using concept of windowing and removing silence frames and after that computing some new feature vector values on basis of selected windows of frames.

The final step is to learn multi-class svm through one-versus-all fashion, i.e learning one svm to classify by distinguishing the samples of one class from the samples of all other classes.

**Various Functions used in our project:**

* 1. **Call Recognition**

Predicts the output of all the audios in Test Set and according to expected to results tells the accuracy of the model.

*Code:*

*features = load(‘test.dat').features;*

*result = load('test\_output.dat').result;*

*result1 = GetBirdIndex(result);*

*model=load('model.dat').model;*

*[predicted\_label1,accuracy,decision\_values]=svmpredict(result1\*1.0, features, model);*

*predicted\_label = GetBirdName(predicted\_label1);*

* 1. **Features Extract**

Extracts the desired MFCC features from the input audio files.

*Code:*

*waveFile = BirdsWaveFilePath;*

*[y fs opt] = wavread(waveFile);*

*opt = mfccOptSet(fs);*

*[mfcc yPreEmp] = wave2mfcc(y,fs,opt);*

*temp(1,:,:) = highestEnergyFrame(mfcc);*

*Function* ***highestEnergyFrame***

*for i = 1:n,*

*vector(i,1) = power(sum(mfcc(:,i).^2)/m,1/2);*

*end;*

* 1. **Predict Bird**

Predicts the output of the input audio of a bird.

*Code:*

*[y fs opt] = wavread(BirdWaveFile);*

*opt = mfccOptSet(fs);*

*[mfcc yPreEmp] = wave2mfcc(y,fs,opt);*

*Data(1,:,:) = highestEnergyFrame(mfcc);*

*[predicted\_label, accuracy, decision\_values] = svmpredict(0, Data, model);*

*predicted\_label = GetBirdName(predicted\_label);*

* 1. **Save Data**

Extracts the MFCC features from Test and Training Set and saves them for future use. It also trains the SVM model on Training Set and saves the model for future use.

*Code:*

*[features result BirdsDirs] = featuresExtract(‘TestSet');*

*BirdMapping(BirdsDirs);*

*save(‘test.dat','features');*

*save(‘test\_output.dat','result');*

*[features result BirdsDirs] = featuresExtract(‘TrainingSet');*

*BirdMapping(BirdsDirs);*

*save(‘train.dat','features');*

*save(‘train\_output.dat','result');*

*result = GetBirdIndex(result);*

*model = svmtrain(result\*1.0, features , option);*

*save(‘model.dat','model');*

* 1. **Bird Mapping**

Generates the mapping of bird names to integer values and stores them in a .dat file.

*Code:*

*mapping = ['a'];*

*if size(DataFiles,2)==3,*

*mapping=[mapping; load('mapping.dat').mapping];*

*end;*

*for i=3:size(BirdDirs,2),*

*flag = 0;*

*for j=1:size(mapping,2),*

*if mapping(j)==BirdDirs{i},*

*flag = 1;*

*break;*

*end;*

*end;*

*if flag==0,*

*mapping = [mapping;BirdDirs{i}];*

*end;*

*end;*

*mapping = mapping(2:end,:);*

*save(‘mapping.dat','mapping');*

* 1. **Get Bird Name**

Converts the integer values to corresponding bird names.

*Code:*

*mapping=load(‘mapping.dat').mapping;*

*name = ['a'];*

*for i=1:size(result,1),*

*s = mapping(result(i),:);*

*name = [name ; s];*

*end;*

*name = name(2:end,:);*

* 1. **Get Bird Index**

Converts the Birds names to corresponding Integer Values.

*Code:*

*mapping=load(‘mapping.dat').mapping;*

*index = zeros(size(result,1),1);*

*for i=1:size(result,1),*

*s = result(i,:);*

*x = 0;*

*for j=1:size(mapping,1),*

*if mapping(j,:)==s,*

*x = j;*

*break;*

*end;*

*end;*

*index(i) = x;*

*end;*

* 1. **Add To Test Set**

Add more audios to Test Set.

*Code:*

*for j = 3:size(BirdsWaveFiles,2),*

*waveFile=BirdWaveFile{j};*

*[y fs opt] = wavread(waveFile);*

*opt = mfccOptSet(fs);*

*[mfcc yPreEmp] = wave2mfcc(y,fs,opt);*

*temp(1,:,:) = highestEnergyFrame(mfcc);*

*Data = [Data ; temp];*

*Result = [Result;res];*

*end;*

*features=load('test.dat').features;*

*result=load(‘test\_output.dat').result;*

*features = [features ; Data(2:end,:,:)];*

*result = [result ; Result(2:end)];*

*BirdMapping({res});*

*save(‘test.dat','features');*

*save(‘test\_output.dat','result');*

* 1. **Add To Training Set**

Add more audios to Test Set.

*Code:*

*for j = 3:size(BirdsWaveFiles,2),*

*waveFile=BirdsWaveFiles{j};*

*[y fs opt] = wavread(waveFile);*

*opt = mfccOptSet(fs);*

*[mfcc yPreEmp] = wave2mfcc(y,fs,opt);*

*temp(1,:,:) = highestEnergyFrame(mfcc);*

*Data = [Data ; temp];*

*Result = [Result;res];*

*end;*

*features=load(‘train.dat').features;*

*result=load(‘train\_output.dat').result;*

*features = [features ; Data(2:end,:,:)];*

*result = [result ; Result(2:end)];*

*BirdMapping({res});*

*save(‘train.dat','features');*

*save(‘train\_output.dat','result');*

* 1. **Change Model**

Change the model to on which SVM trained on Training Set.

*Code:*

*features=load(‘train.dat').features;*

*result=load(‘train\_output.dat').result;*

*result1 = GetBirdIndex(result);*

*model = svmtrain(result1\*1.0, features , options);*

*save(‘model.dat','model');*

* 1. **MFCC**

Extracts MFCC from input audio.

*Code:*

*yPreEmp = filter([1, -opt.preEmCoef], 1, y);*

*yPreEmp = floor(yPreEmp);*

*framedY = enframe(yPreEmp, opt.frameSize, opt.overlap);*

*filterBankParam = getTriFilterPrm(opt.frameSize, fs, opt.tbfNum, 0);*

*mfcc = [];*

*for i = 1:size(framedY, 2),*

*Wframe = hamming(opt.frameSize).\*framedY(:,i);*

*fftMag = abs(fft(Wframe));*

*halfIndex = floor((opt.frameSize+1)/2);*

*fftMag = fftMag(1:halfIndex);*

*fftMag=interp1(1:halfIndex,fftMag,1:1/opt.alpha:halfIndex)';*

*tbfCoef=triBandFilter(fftMag,opt.tbfNum, filterBankParam);*

*theMfcc=melCepstrum(opt.cepsNum, opt.tbfNum, tbfCoef);*

*mfcc = [mfcc theMfcc'];*

*end;*

**Result:**

|  |  |
| --- | --- |
| SVM-type Kernel-type | Accuracy |
| C-SVC Linear | 75% |
| C-SVC Polynomial | 69% |
| C-SVC Radial | 46% |
| NU-SVC Linear | 75% |
| NU-SVC Polynomial | 69% |
| NU-SVC Radial | 46% |

So we decided to use C-SVC as svm type and linear kernel.

**Conclusion**

Given a wave file as input, the SVM model will predict the name of the bird to whom input audio belongs to.

**]**

**Future Scope**

**Extension of Bird Call recognition to Human voice recognition [2]**

In the used features (MFCCs) the frequency bands are equally spaced on the mel-scale, which approximates the human auditory systems response more closely than the linearly-spaced frequency bands used in normal spectrum and hence are considered as good features for process of speech recognition and hence bird call recognition can be modified to Human voice recognition.

**References**

1. Graber, R. R. and W. W. Cochran. 1959. An audio technique for the study of nocturnal migration of birds. Wilson Bull. 71:220-236.
2. Hemant Tyagi, Rajesh M. Hegde, Hema A. Murthy and Anil Prabhakar “Automatic Identification of Bird Calls using Spectral Ensemble Average Voice Prints”. IEEE(ISSN-2219-5491),2006