# Digital Signal Processing Theory (EC31008) Assignment - 2

***Group Members:*** *A Jaaneshwaran (18EC10071) & Ayan Chakraborty (18EC10075)*

---

## (A) AIM

The entire problem of this assignment can be divided into two parts:

1) **Speaker identification:** From the signal, identify that the speech indeed belongs to an authorised speaker. The main approach to this problem is extraction of temporal frequency characteristics, which will be unique to each speaker. Later, the same temporal frequency characteristics can be extracted from the test signal and compared to get the closest match. There are 2 main ways in which we can extract frequency domain features:
    a) Use the entire signal and pass it through uniform/non uniformly spaced frequency band filters and use the energy of the output of each filter as a feature.
    b) Divide the signal into small windows and extract the pitch and MFCC coefficients for each window. Using pitch and MFCC coefficients as feature vectors for speaker identification has several advantages (discussed later).
2) **Detection of the code spoken:** Even if it is an authorised speaker, the speaker needs to say the correct code, otherwise it is unauthorised access. A deep learning solution is adopted for this problem for digit identification.

## (B) THEORY

### PART 1: KNN based classification of the speaker where the feature vector is the energy of the input waveform after passing through a filter bank

The input sound wave of each speaker was passed through a filter bank. The filter bank was varied throughout the experiment as follows:-

1) 10 uniformly spaced filter banks between 0-4000 Hz
2) 100 uniformly spaced filter banks between 0-4000 Hz
3) 1000 uniformly spaced filter banks between 0-4000 Hz
4) 10 Uniformly spaced filters between 0-1000Hz and 10 Uniformly Spaced filters between 1000-4000Hz
5) 100 Uniformly spaced filters between 0-1000Hz and 100 Uniformly Spaced filters between 1000-4000Hz
6) 1000 Uniformly spaced filters between 0-1000Hz and 1000 Uniformly Spaced filters between 1000-4000Hz
7) 100 Uniformly Spaced Filters between 0-100 Hz, 1000 Uniformly Spaced filters between 100-1000Hz, and 100 Uniformly Spaced filters between 1000-4000Hz

---

(*Uniformly spaced filters mean the difference between the upper cutoff frequency and the lower cutoff frequency is equal between all the filters in the filter bank) These variations were done to better characterize the filter bands that the human ear uses for speech perception.

The filter bank was created by convolving a low pass filter (with its cutoff being the higher band roll-off frequency) and a high pass filter (with its cutoff being the lower band roll-off frequency). The low pass filter was generated by using the sinc function which has been multiplied by the window function (Blackmann, Hanning, Hamming). The high pass filter was generated by the difference between the unit sample function and the sinc function which was later scaled by the window function.

Eight samples were collected from seven speakers (Ayan, Jaanesh, Nivedita, Prerna, Rudrajyoti, Shouvik, Swarnava). There were a total of 56 samples. All the speakers spoke the following combination of numerals - 2 4 3 9 1 7 5

## PART 2: KNN based classification of the speaker where the feature vectors are the MFCC Coefficients and the pitch of the sound

### Pitch:

The Pitch is the fundamental frequency of the vocal cords vibration. It is a characteristic of the speaker identity and the gender of the speaker. It generally ranges from 50 Hz(for males) to 400 Hz (for babies). There are various methods for pitch (or fundamental Frequency) extraction such as measuring the zero crossings or performing autocorrelation with sinusoids of known frequency. (One of the methods is documented later). In this example, directly the pitch() function provided by MATLAB is used to extract the pitch. The pitch function uses a normal correlation function to estimate the pitch over a window of the time signal. This window is then shifted and the process is repeated over the entire length of the time sequence.

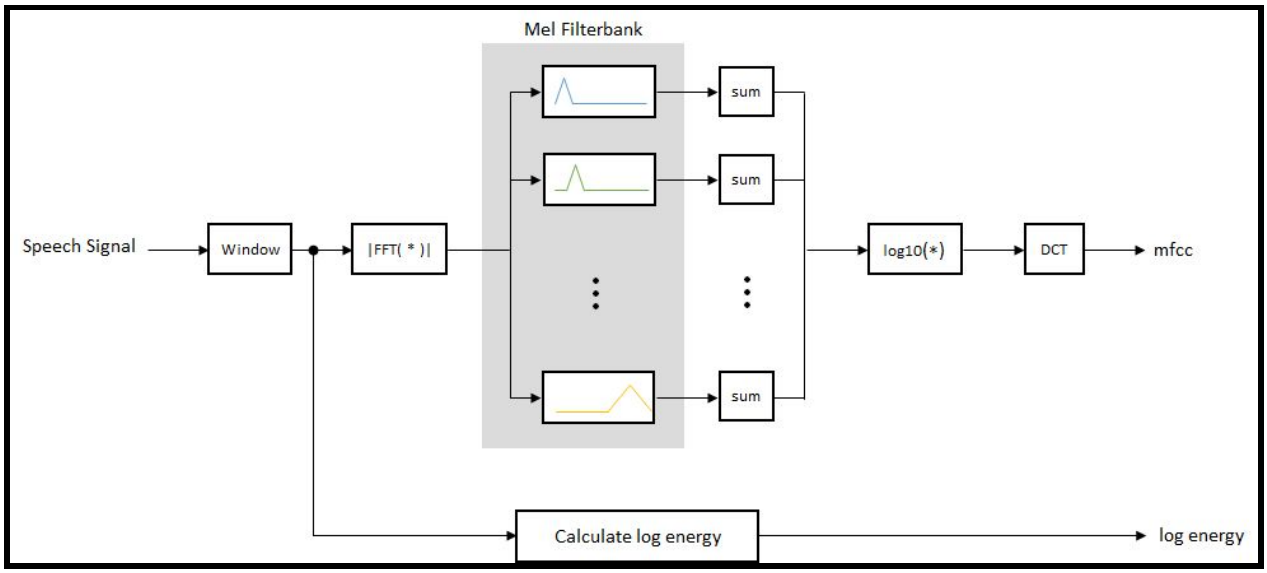
### MFCC:

In sound processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum. This frequency warping can allow for better representation of sound.

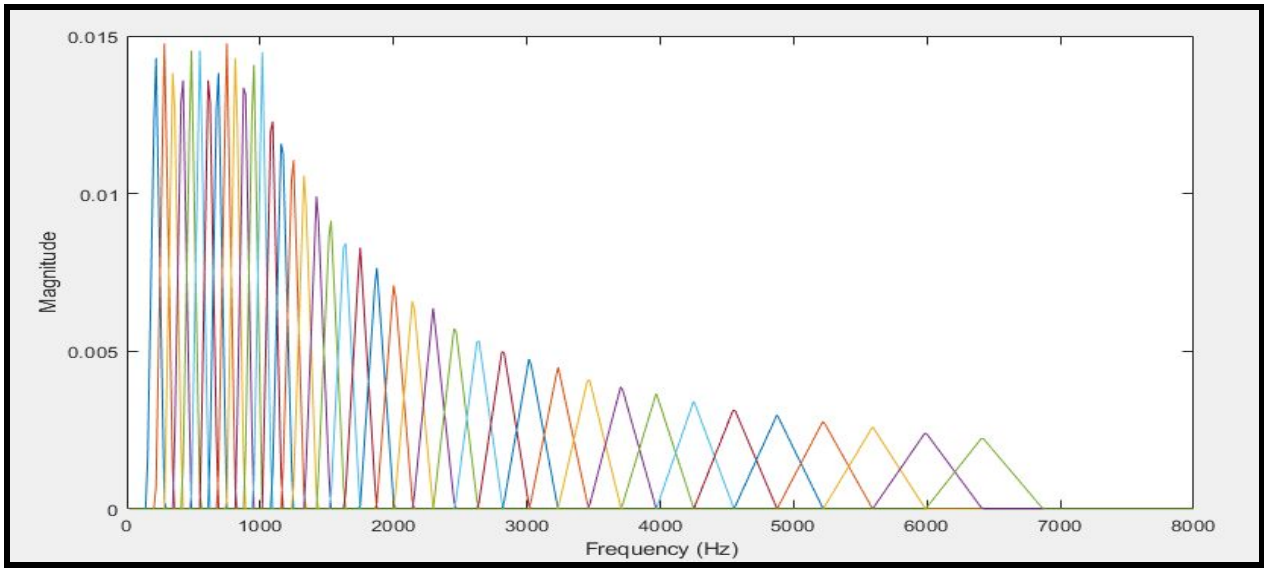
The motivating idea of MFCC is to compress information about the vocal tract (smoothed spectrum) into a small number of coefficients based on an understanding of the cochlea. This accurately models the speech perception mechanism of humans and is expected to achieve the best results.

MFCCs are commonly derived as follows:

1) Take the Fourier transform of (a windowed excerpt of) a signal.
2) Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
3) Take the logs of the powers at each of the mel frequencies.
4) Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5) The MFCCs are the amplitudes of the resulting spectrum.




We have directly used the mfcc() function provided by MATLAB which works as shown above. The mel filterbank linearly spaces the first 10 triangular filters and logarithmically spaces the remaining filters. The individual bands are weighted for even energy. The graph represents a typical mel filterbank:



## K-Nearest Neighbour Classifier

K-Nearest Neighbour is a simple classifier algorithm wherein the feature vectors of the training set are arranged in an L-dimension space where L refers to the length of the feature vector. The k-Nearest Neighbour is based on the premise that the feature vector belonging to a particular label is closer to each other in comparison to the feature vector belonging to a different label.

Later when a test case feature vector is used to verify the behavior of the KNN classifier, it basically calculates the distance of the test feature vector (can be L2 distance or L1 distance or any other distance calculating function) with respect to each training set feature vector and finds the k nearest training set feature vectors.

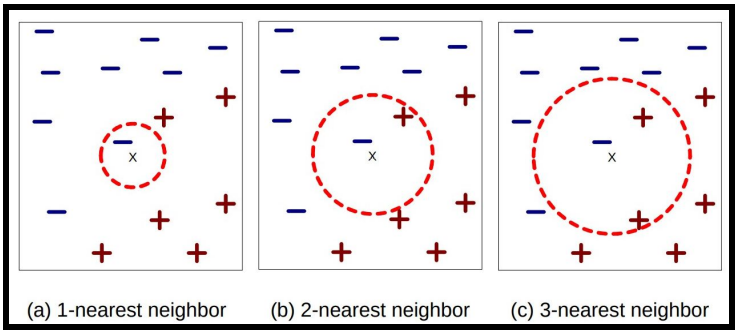
Next, these 'k' nearest feature vectors are weighted by a weighting function. The weighting function gives a higher weight to nearer feature vectors and a lower weight to feature vectors further away. In this experiment, we have chosen the weighting function to be the square of the inverse distance. After weighting the feature vectors, the effective number of feature vectors belonging to each class label is counted, and the class label having the highest score is assigned to the test vector. In case k = 1, it reduces to the nearest neighbour problem.

The mathematical formulation of the KNN is given as:

$$\hat{f}(q) = \arg\max_{v \in V} \sum_{i=1}^{k} \frac{1}{d(x_i, x_q)^2} \delta(v, f(x_i))$$

Here, the delta function gives 1 if the feature vector belongs to that class label, and the class label is chosen that maximises this function

Example of KNN working in 2 dimensions:




Certain pre-processing steps such as Condensing and/or K-D Trees can be adopted to reduce the pre-processing time, but since we are working with a relatively smaller sample set, we have not used them.

The KNN based classifier is available in Matlab as a default function called **fitcknn.** All the 56 feature vectors were used to train the KNN module and later as a test case the average of the 8 samples of a particular speaker was passed as input to the KNN classifier and in each instance, the input test case was rightly matched to the speaker.

To further test the performance of the designed KNN classifier a k fold cross-validation was used as we had a limited number of samples. In the k fold cross-validation, we split the available feature vectors into k groups randomly. Then a KNN model is trained excluding one group and that excluded group is used as a test case. The test case result is stored and the trained model is discarded and the above process is repeated k times such that each group of feature vectors act as a test case a single time. This is achieved by using the **crossval** function in MATLAB. This gives us a better estimate of the validation accuracy. Confusion plots are used. It shows the True Positives, True negatives, False positives, False Negatives for all the classes.

## Pre Processing steps adopted for part 1 (Energy of each Filter Band as FVs)

1) **Feature Normalization:** Since the energy would be vastly different for different speakers it has to be pre-processed to avoid biasing the KNN model. The pre-processing that was carried was to subtract the feature vector from its mean value and divided it by its standard deviation. The mean and standard deviation of the input feature vector was calculated by using the default function **mean()** and **std().**

## Pre Processing steps adopted for part 2 (Pitch and MFCC as FVs)

1) **Detection of voiced and unvoiced regions:** Each of the windows in the time domain is first checked to see if it contains voiced speech or unvoiced speech. There are several methods of differentiating between voiced and unvoiced speech. First, unvoiced regions contain much lesser power compared to voiced regions. Also, most of the power for unvoiced regions is concentrated in the high frequency components whereas voiced regions have power concentrated in the lower frequency components. Secondly, unvoiced regions have much higher rates of zero-crossings. Hence, for detection of unvoiced regions, the energy and number of zero crossings are checked for each window. Windows found to contain unvoiced speech are discarded.

2) **Amplitude scaling:** After the unvoiced regions have been removed, the signal peak amplitude needs to be normalized, so that the feature vectors are standardised across all speakers. This is done by simply dividing the signal by the maximum absolute value present in the signal.

3) **Feature Normalization:** Pitch and MFCC are not on the same scale. This will bias the classifier. The features are normalized by subtracting the mean and dividing the standard deviation.

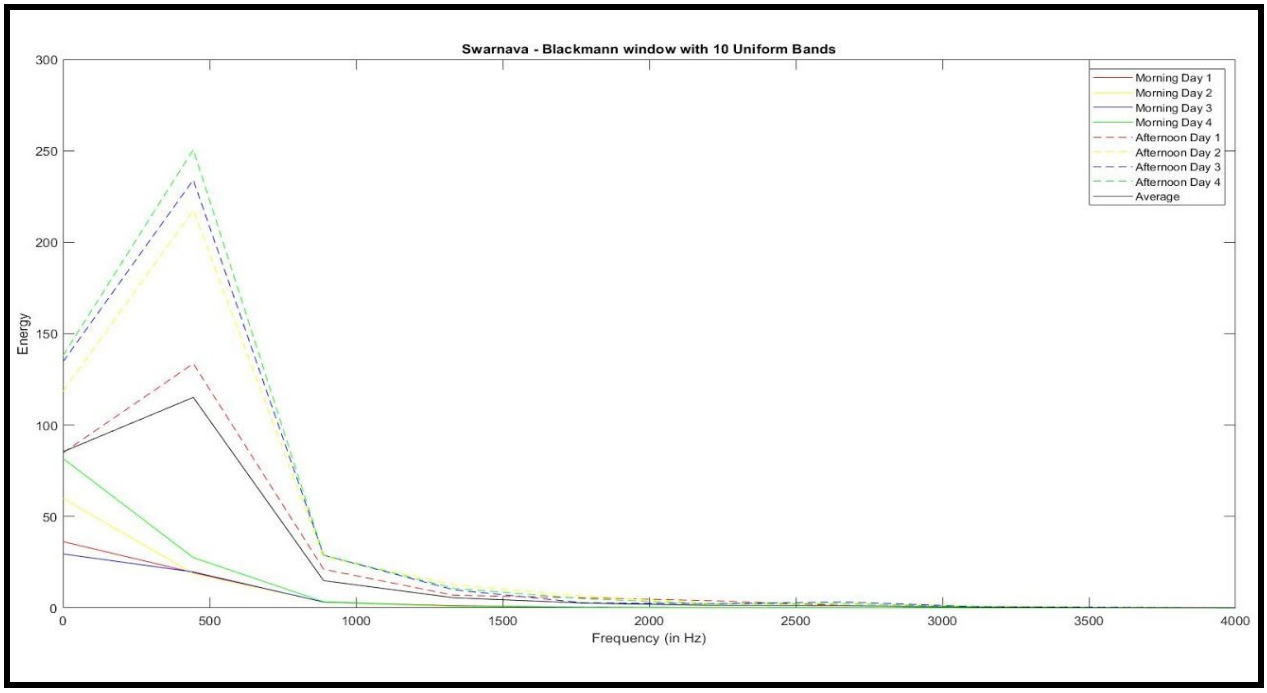
**Testing this KNN Classifier:**

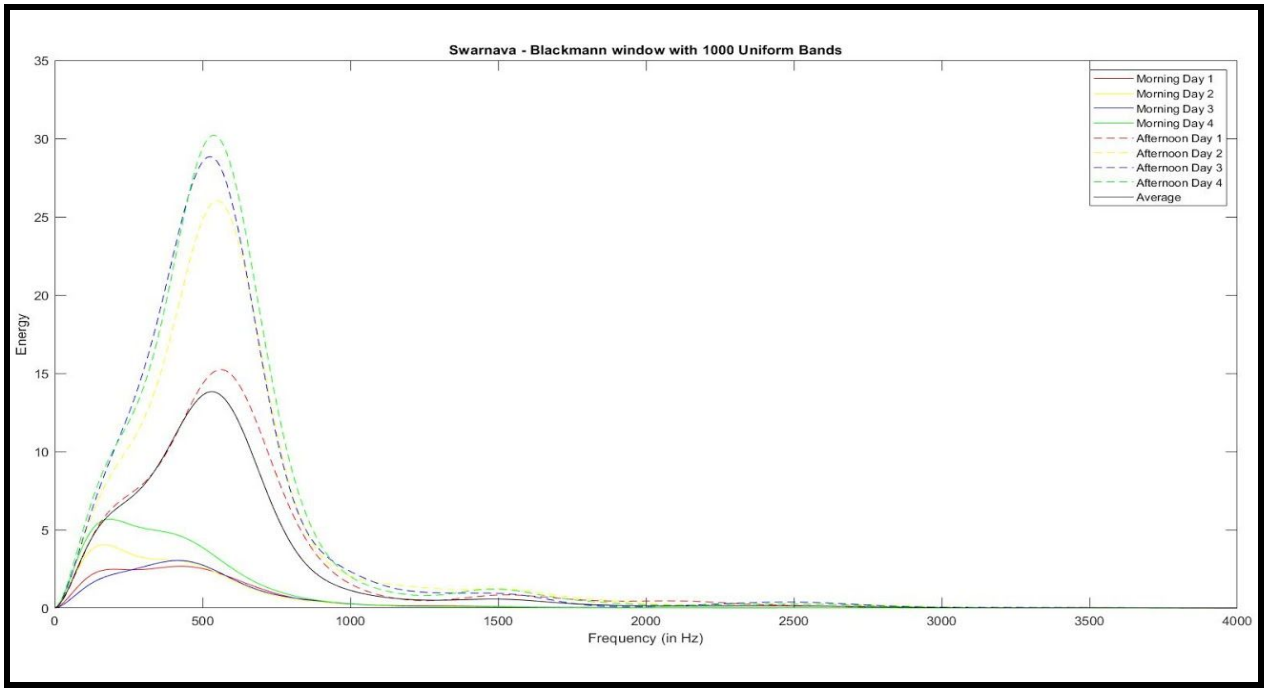
The entire signal is divided into windows, and each window is classified by the KNN. Then the number of classifications made to each speaker is calculated, and the fraction is calculated. If none of the authorised speakers have a fraction more than 0.5 or 50%, then the speech is classified as unauthorised. Else, it is assigned to the speaker having the highest fraction

# (C) RESULTS for Part 1

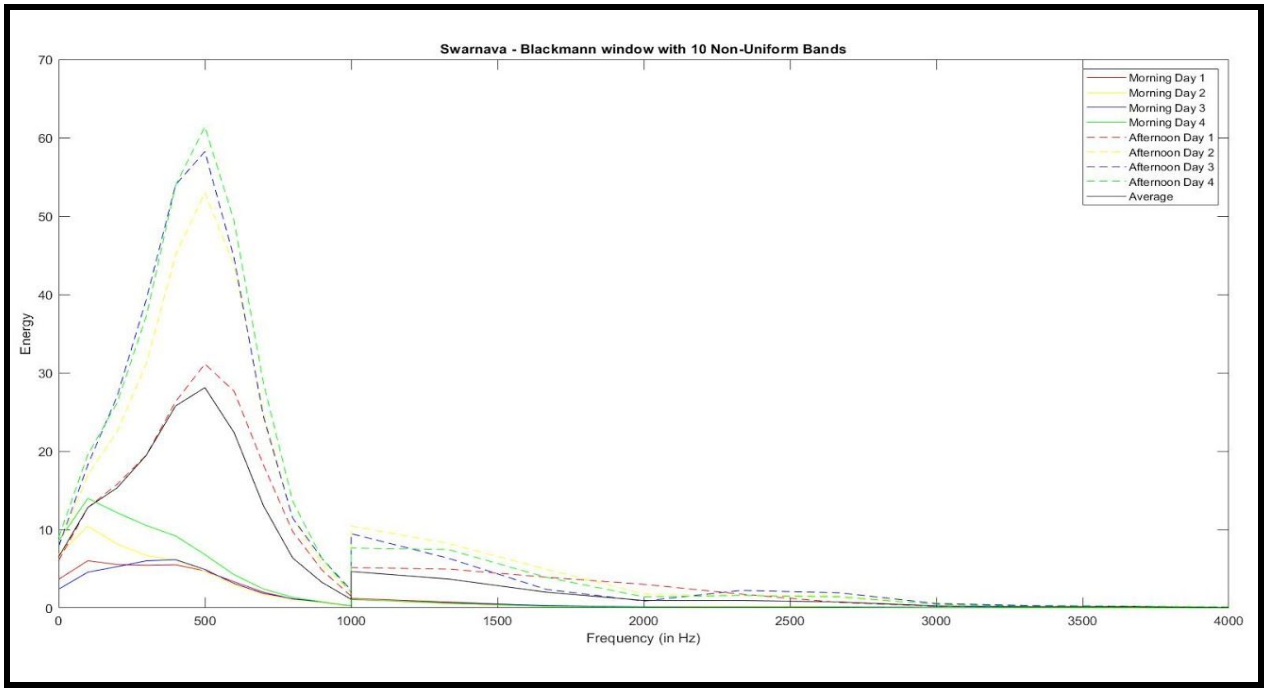
**The graphs have been plotted as follows:** The x-axis is the lower edge of the frequency bin (say if there is a bandpass filter with cutoffs 100-200Hz then the energy of the input waveform after passing through the filter is plotted at the lower frequency value here 100Hz).

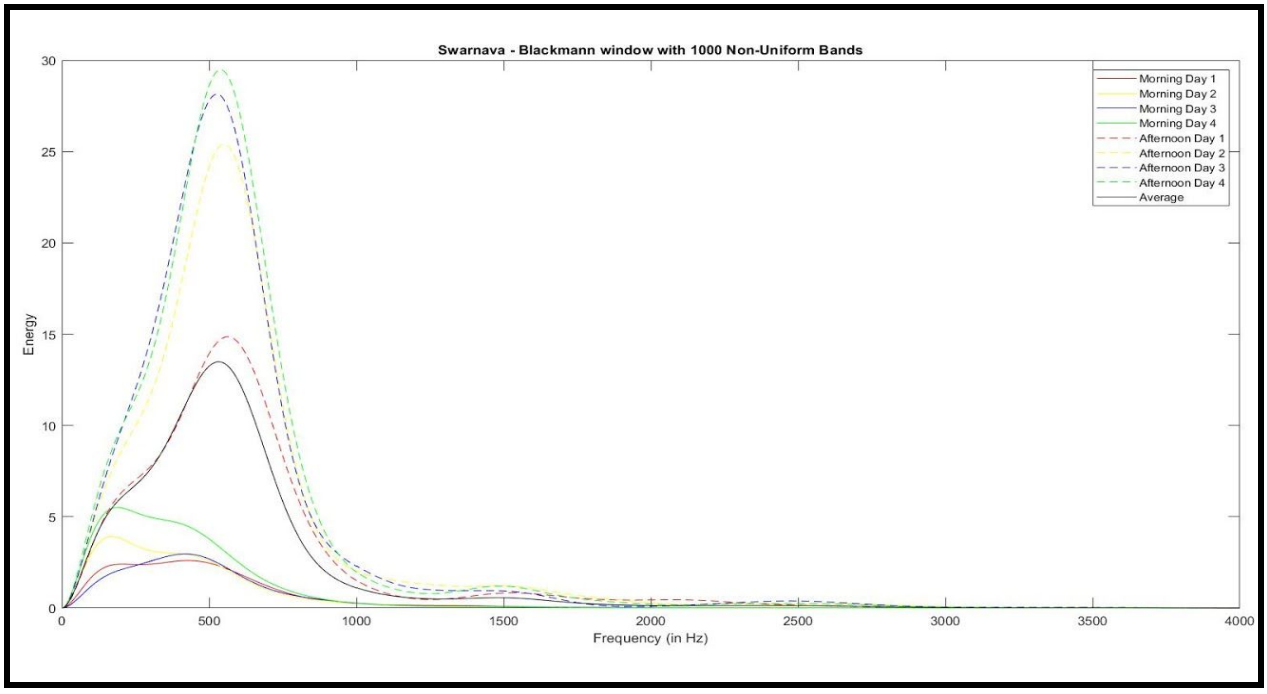
**Energy - Frequency plots for Swarnava using Uniform Frequency Bands**



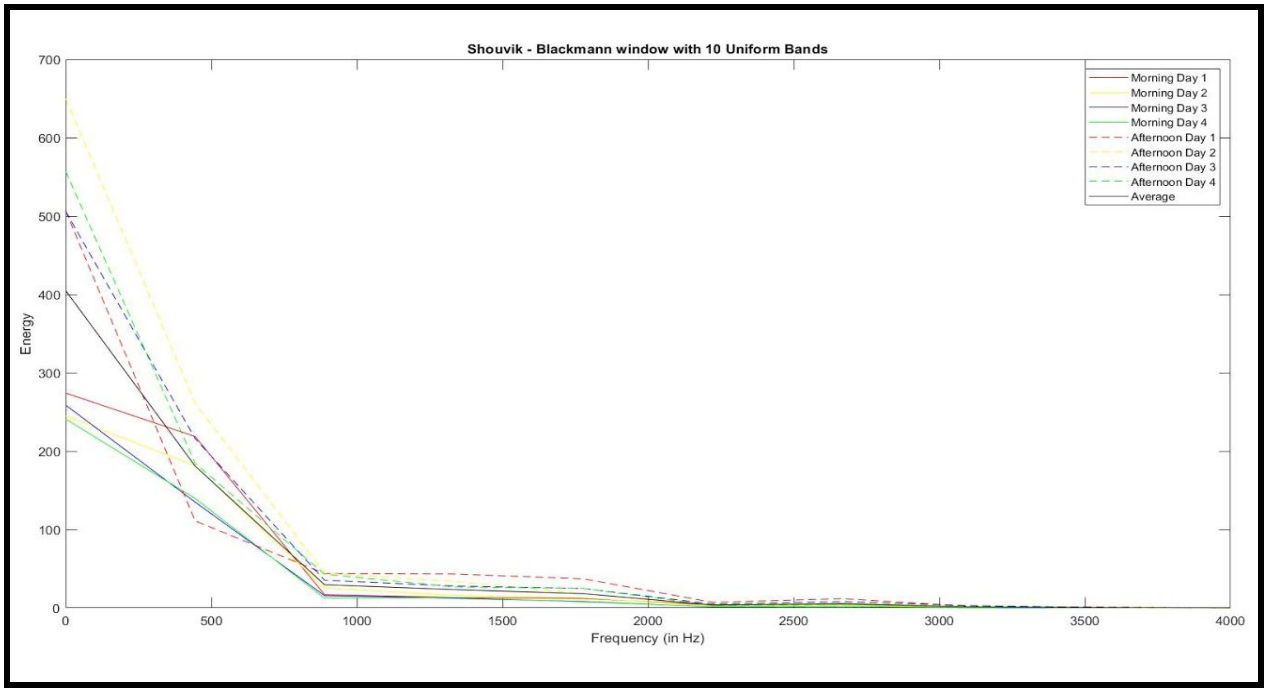



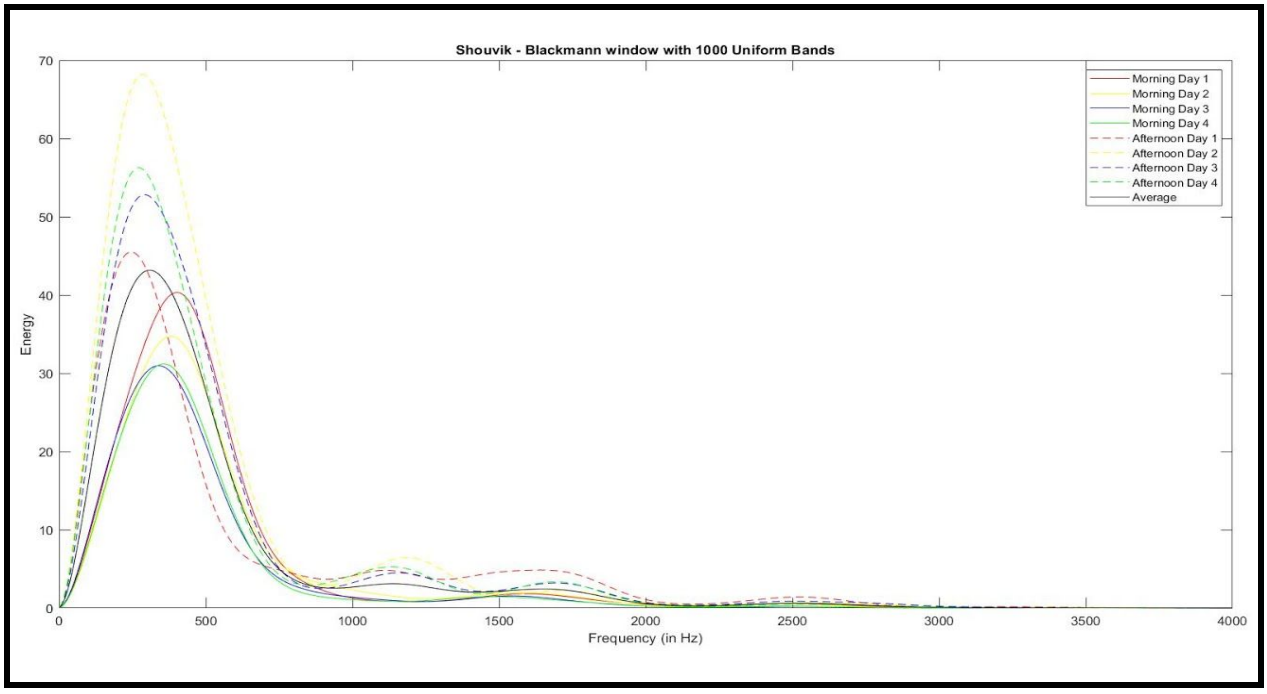
## Energy - Frequency plots for Swarnava using Non-Uniform Frequency Bands



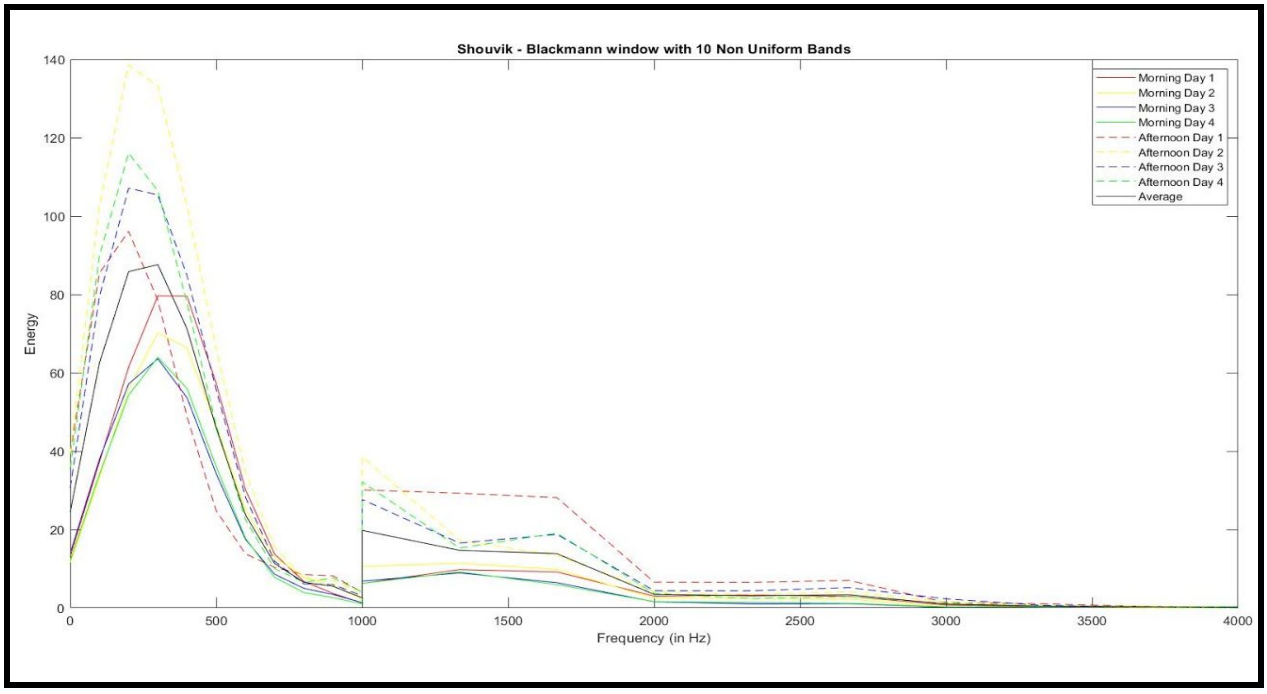



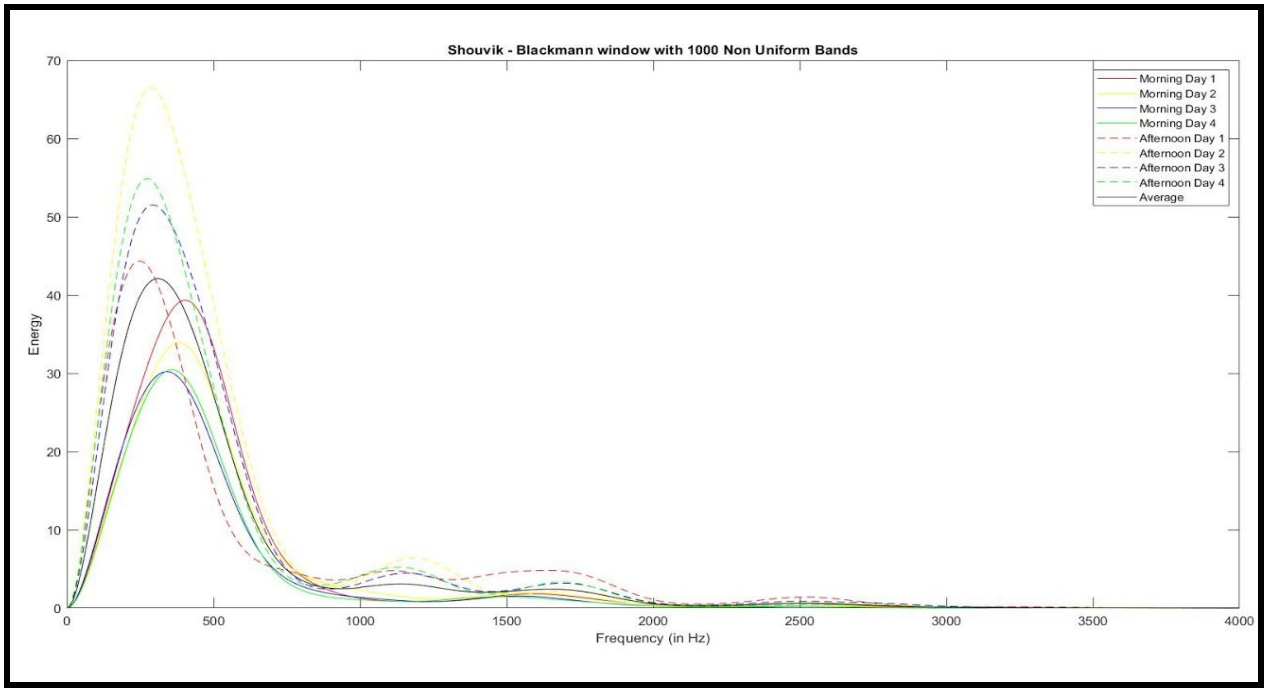
## Energy-Frequency Plots for Shouvik using Uniform Frequency Bands



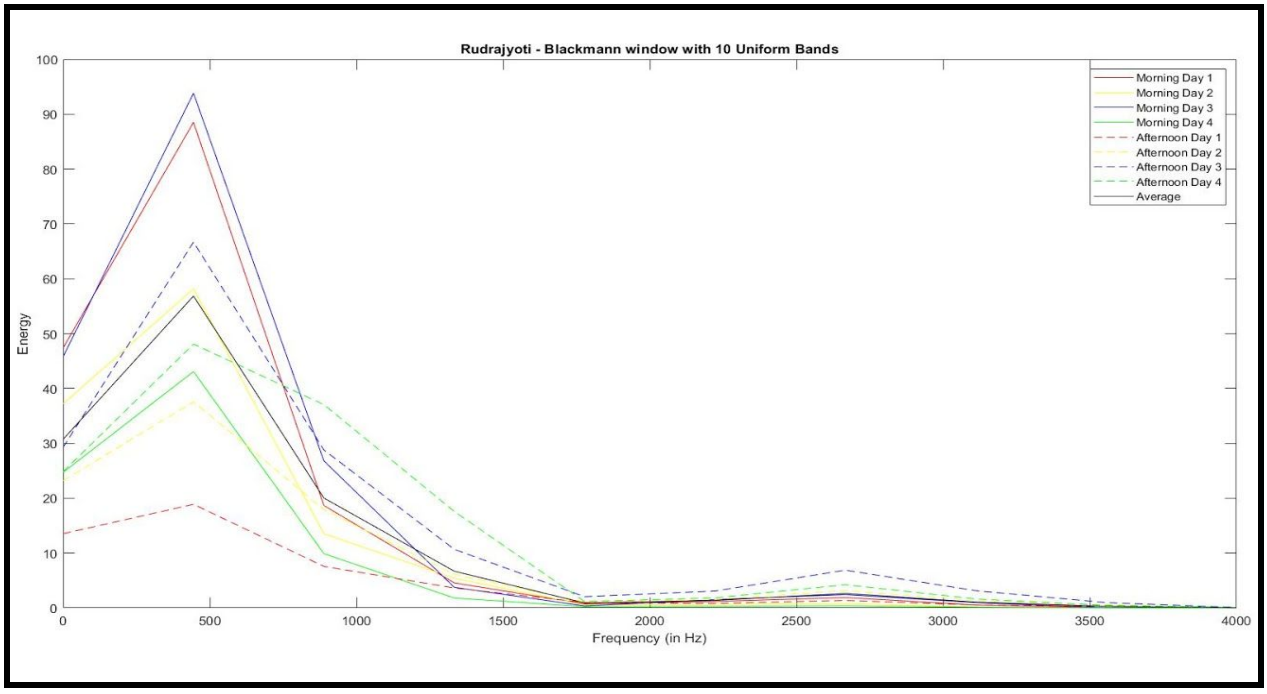



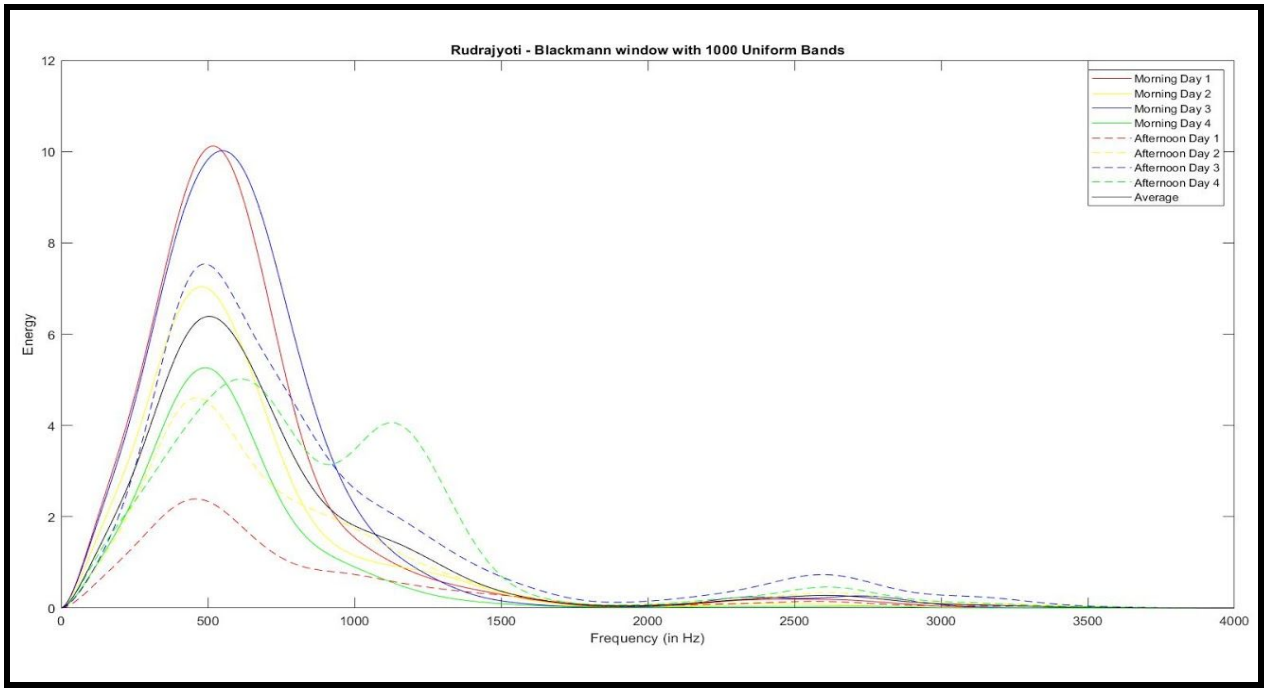
## Energy-Frequency Plots for Shouvik using Non-Uniform Frequency Bands



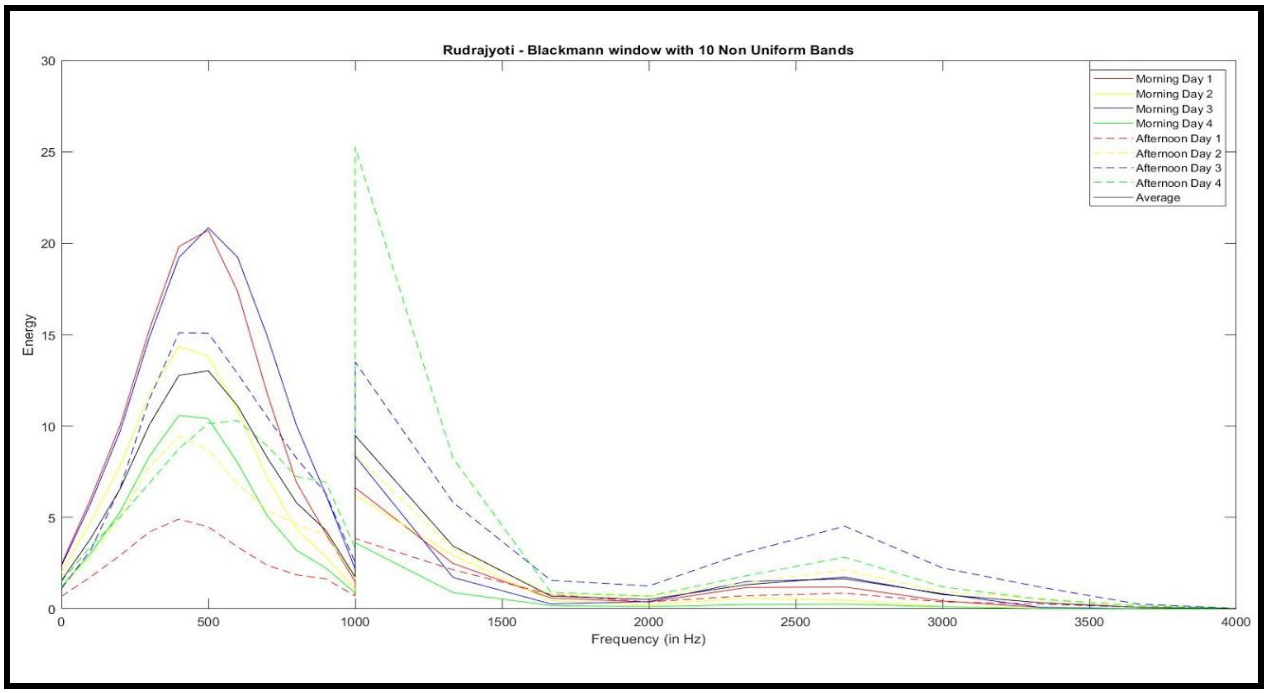



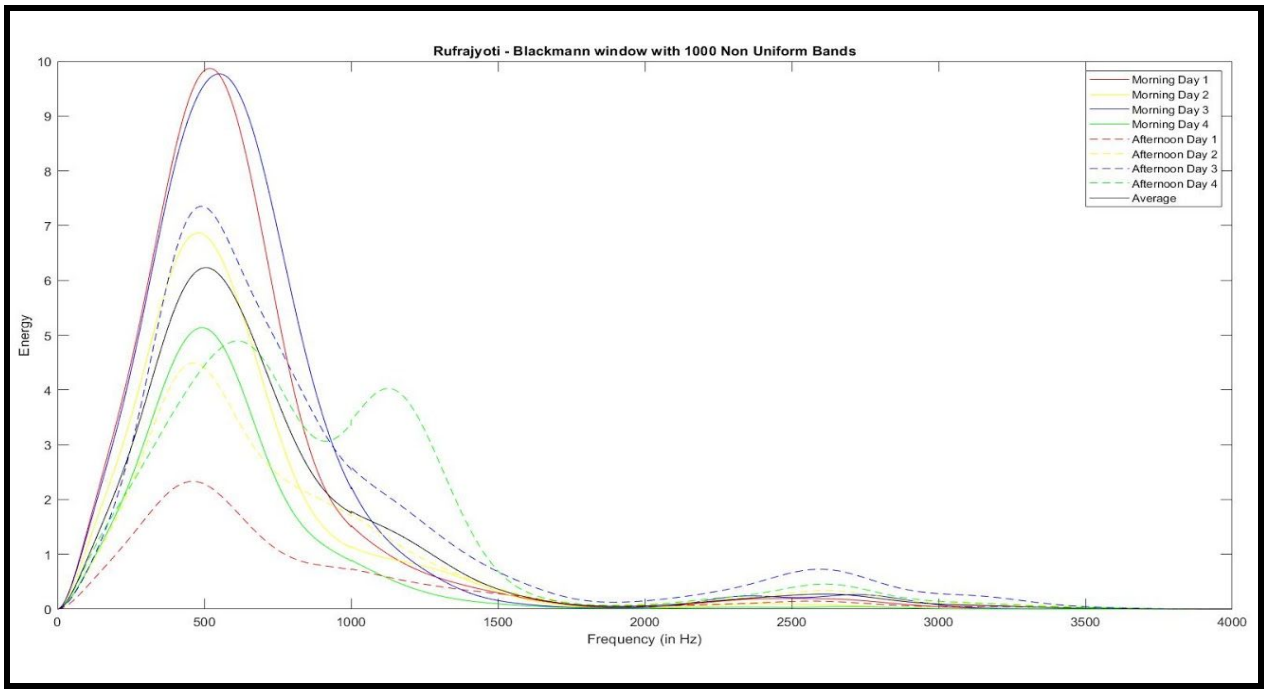
## Energy-Frequency Plots for Rudrajyoti using Uniform Frequency Bands



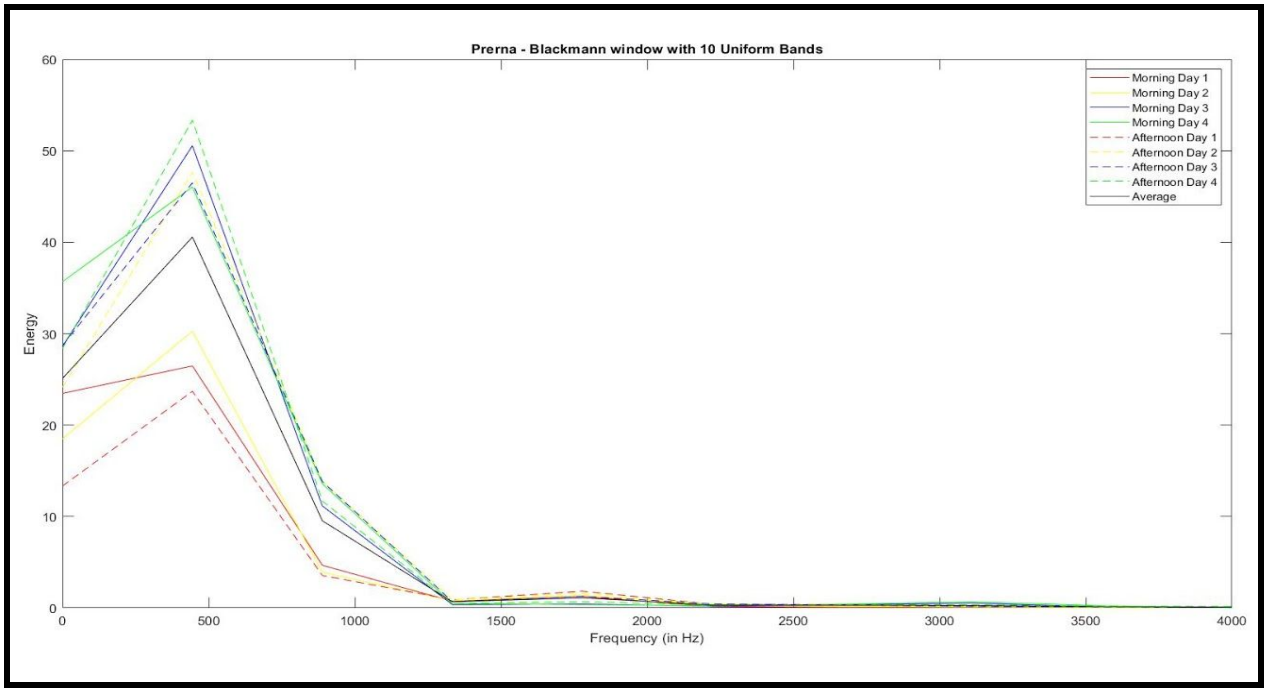



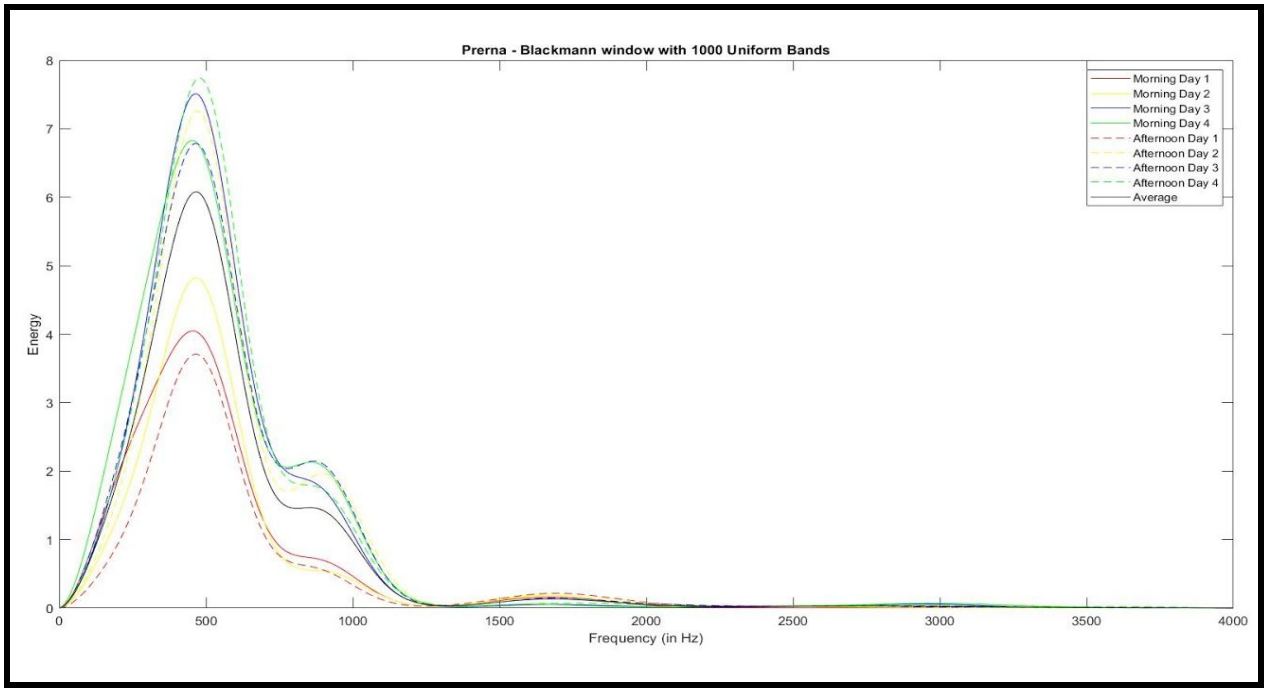
## Energy-Frequency Plots for Rudrajyoti using Non-Uniform Frequency Bands



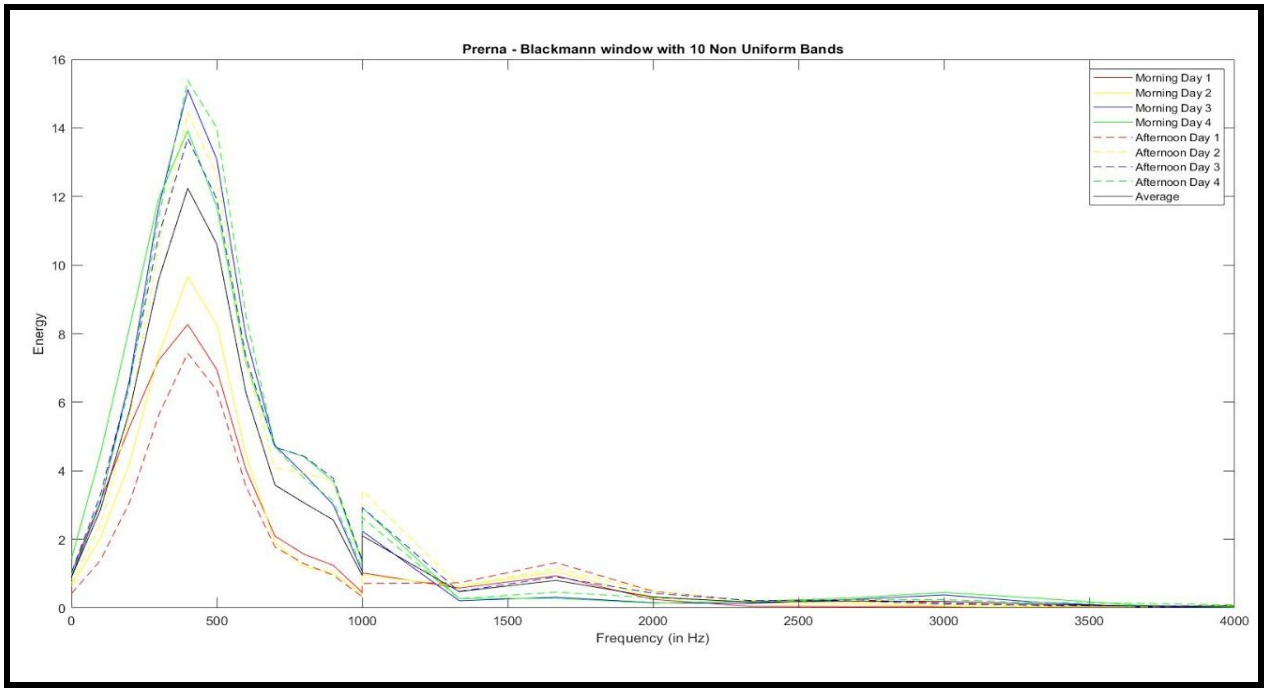



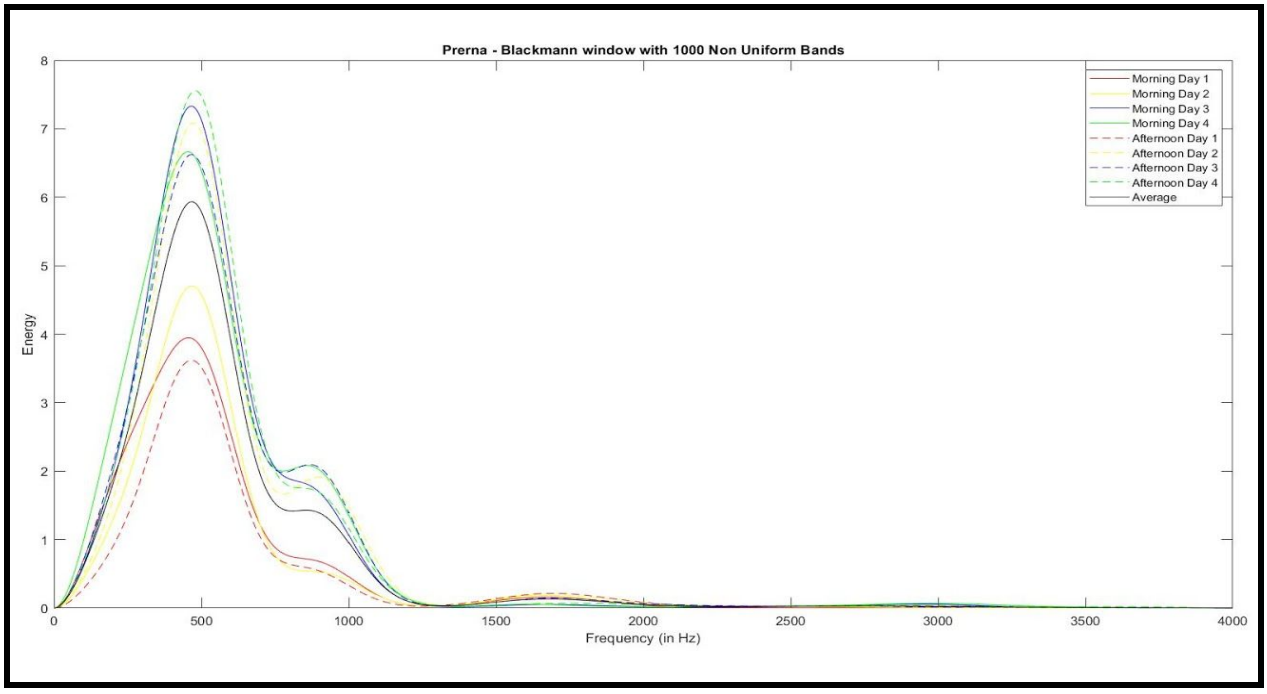
## Energy-Frequency Plots for Prerna using Uniform Frequency Bands



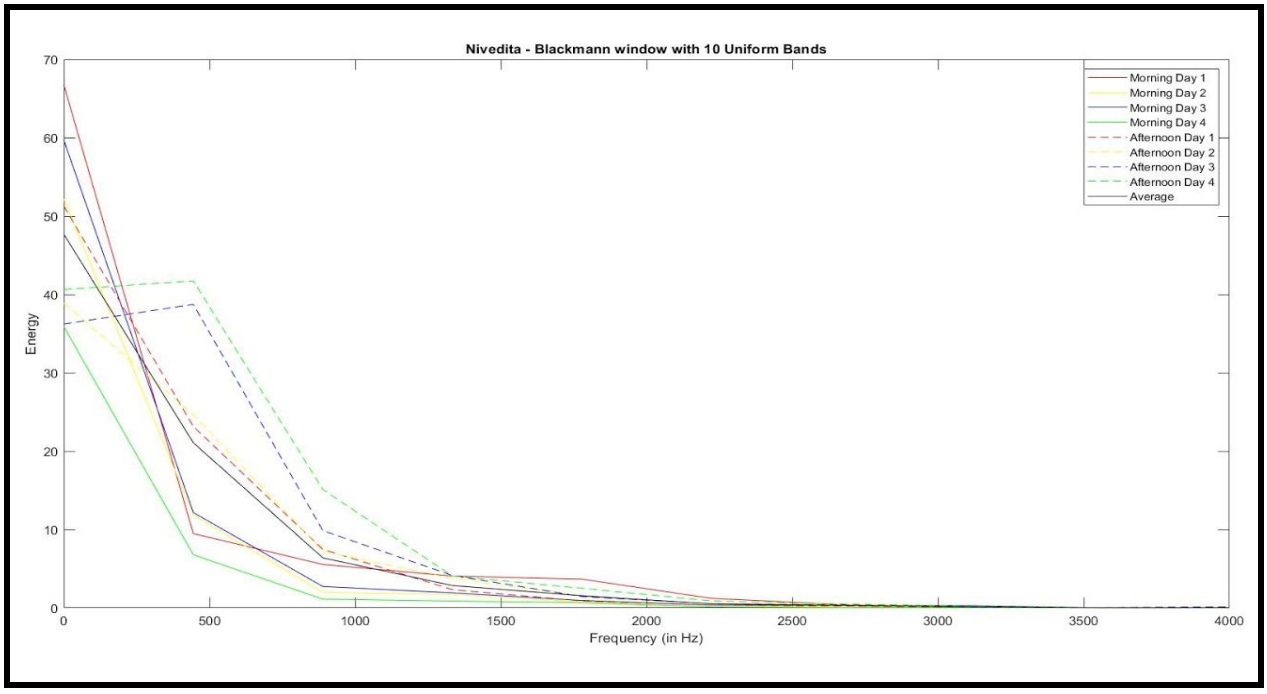



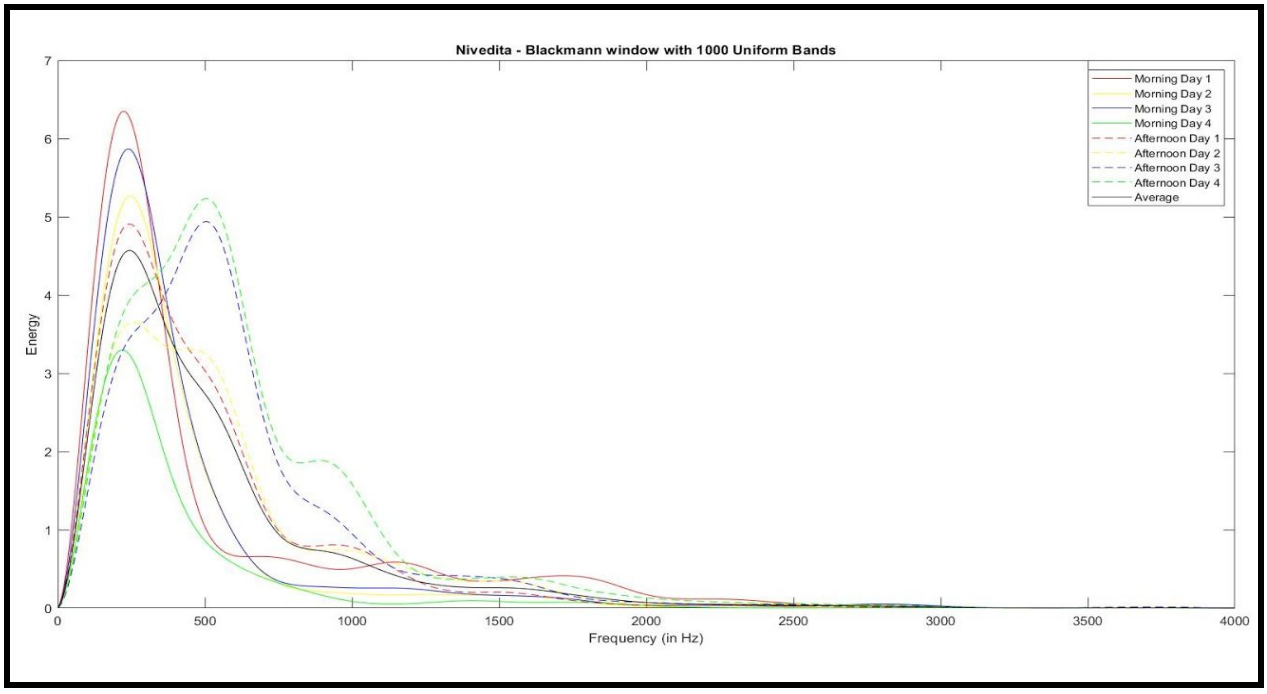
## Energy-Frequency Plots for Prerna using Non-Uniform Frequency Bands



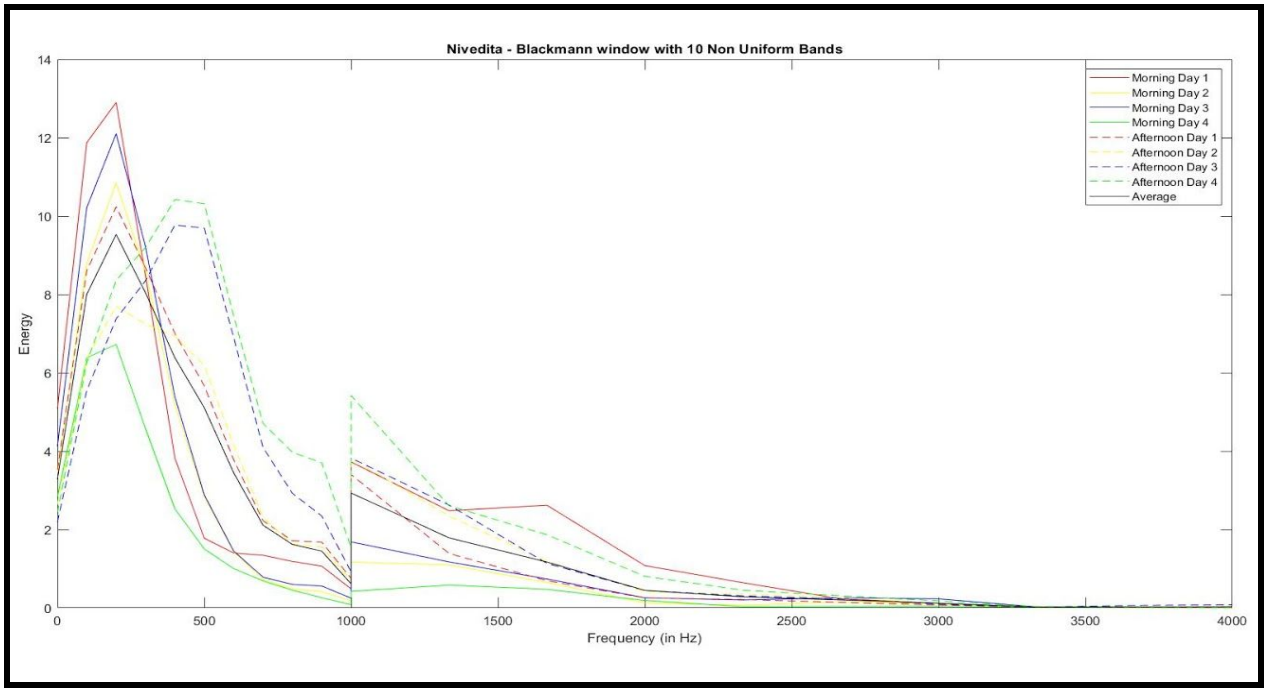



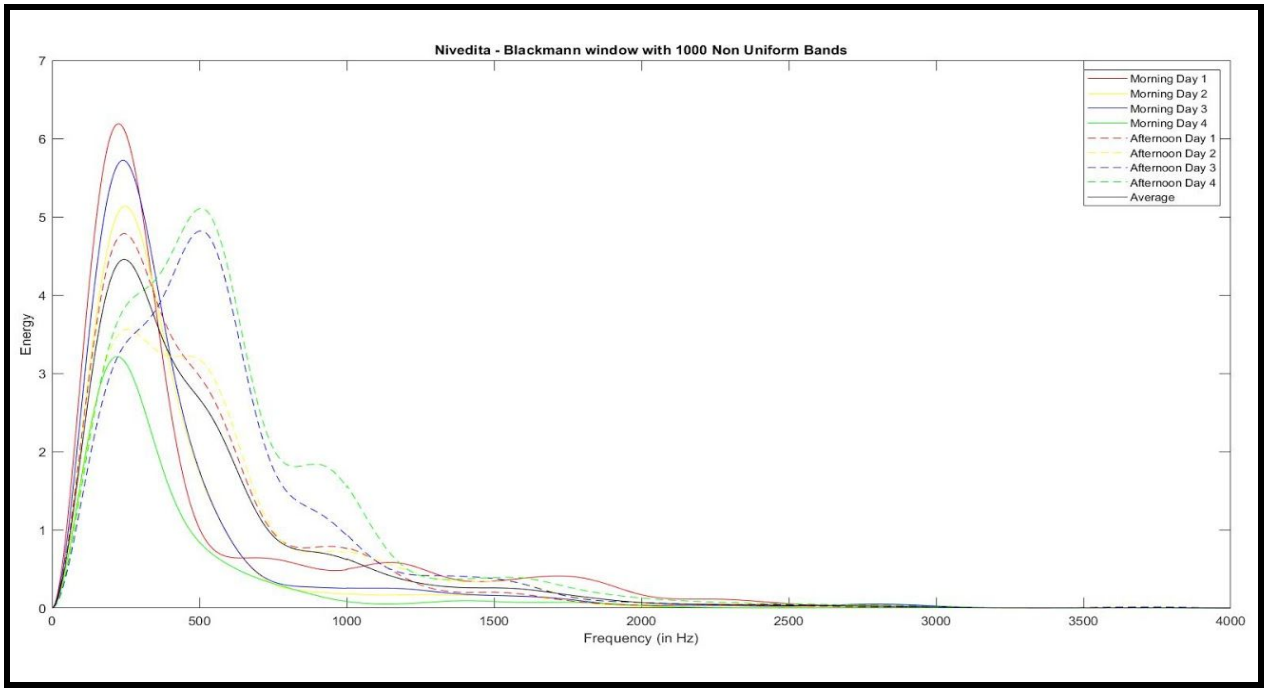
## Energy-Frequency Plots for Nivedita using Uniform Frequency Bands



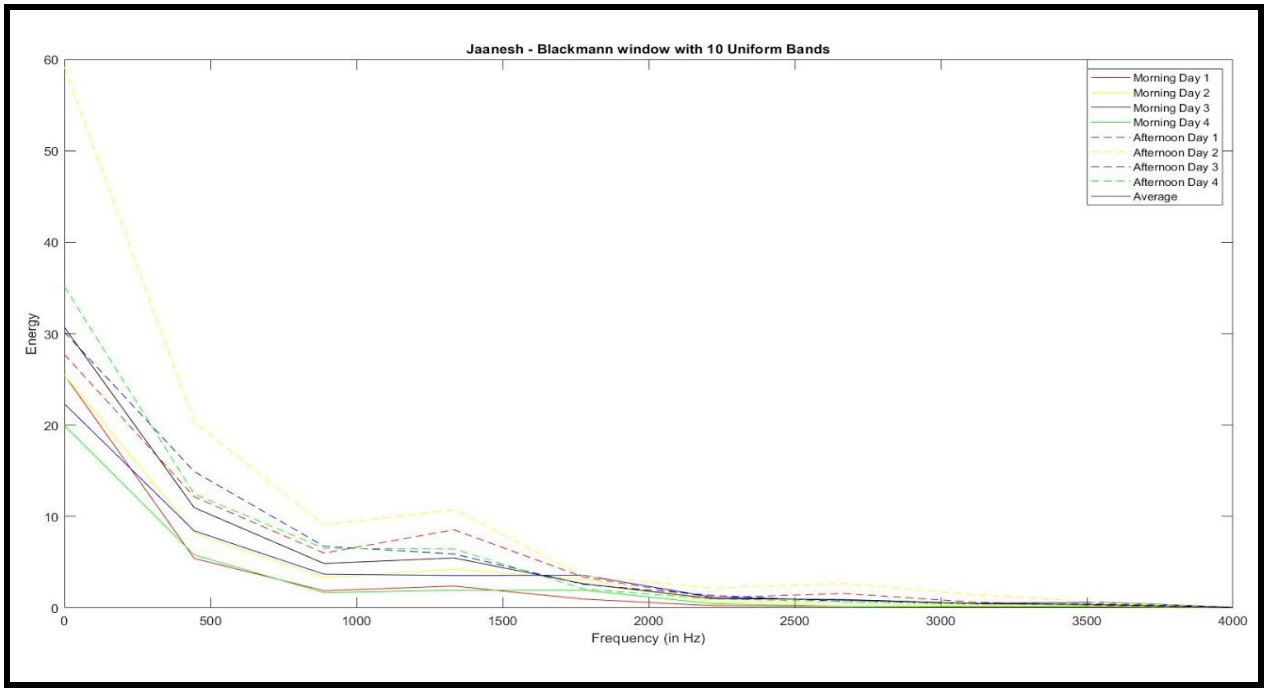



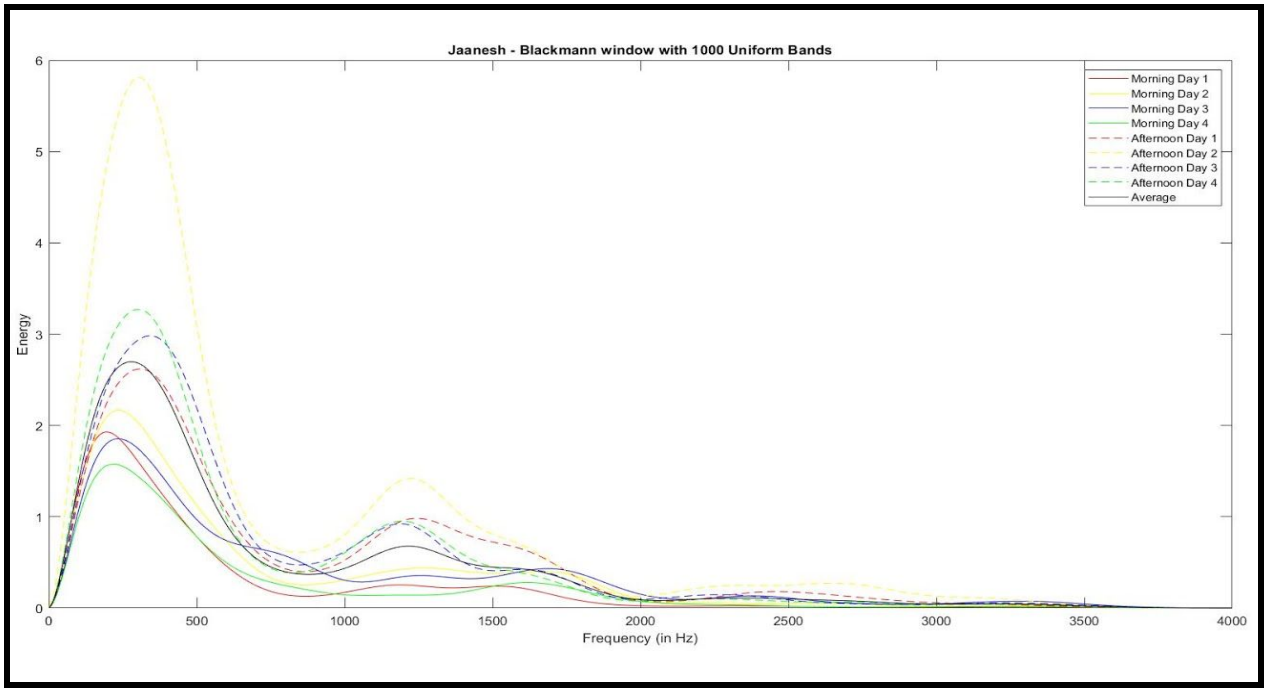
## Energy-Frequency Plots for Nivedita using Non-Uniform Frequency Bands



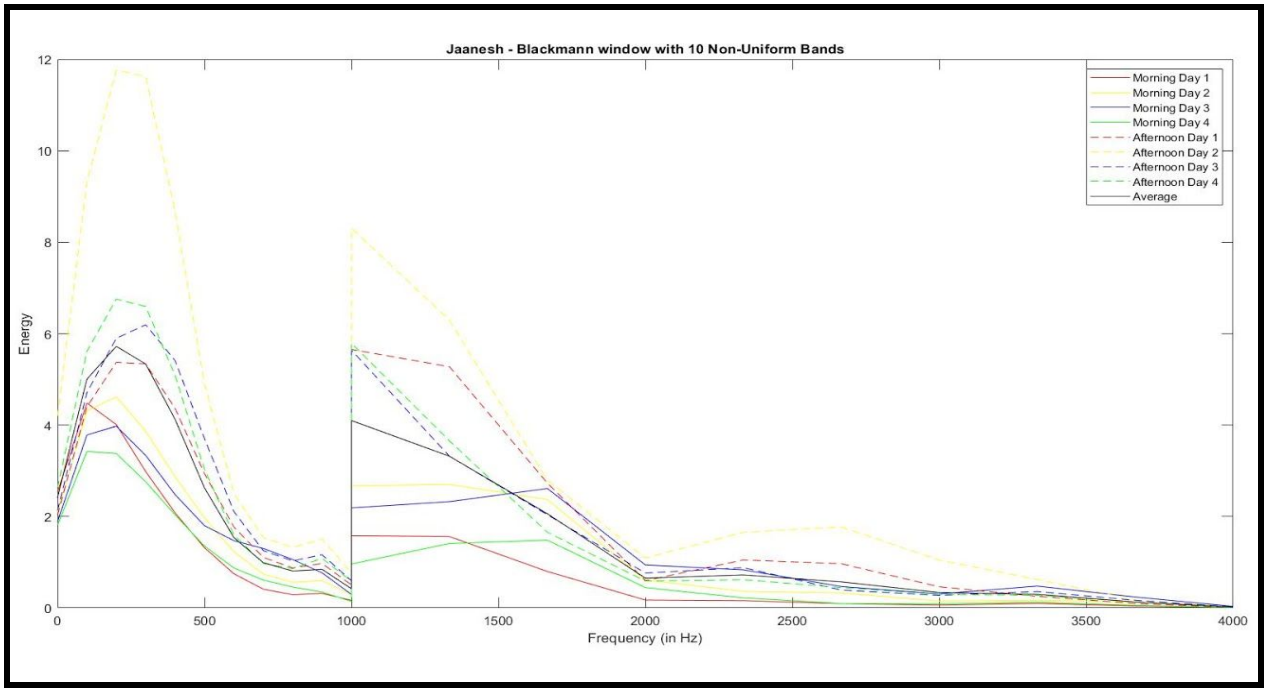



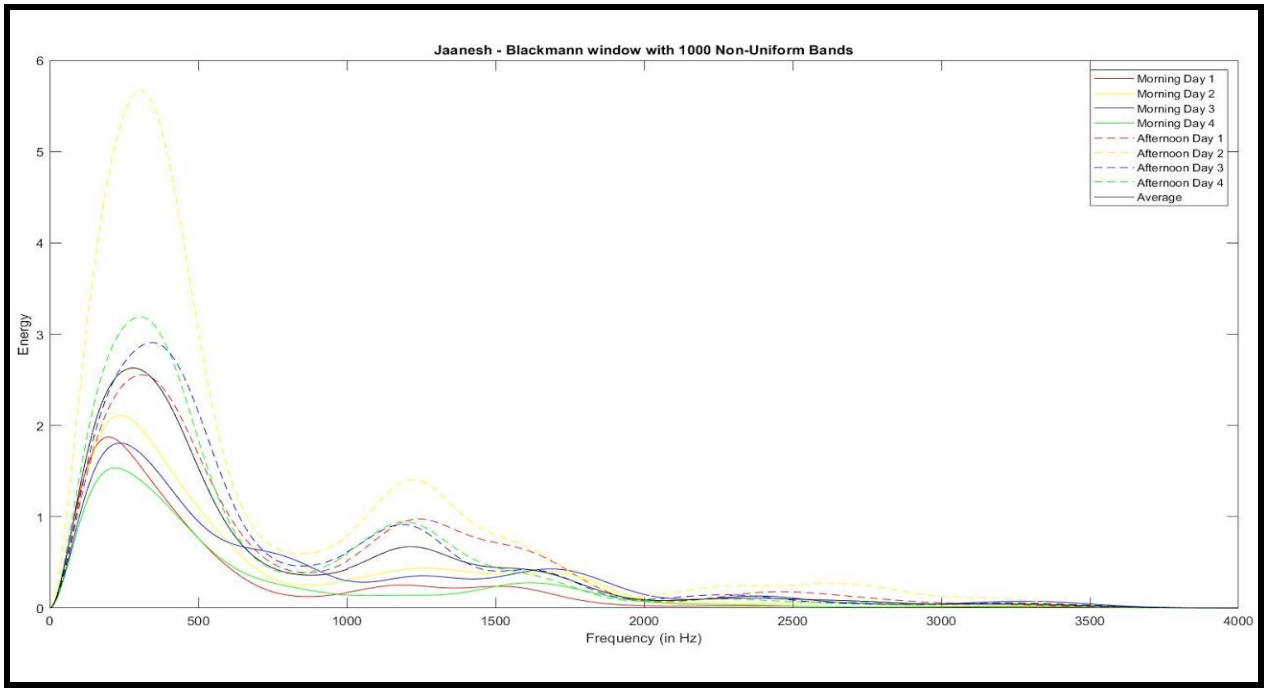
## Energy-Frequency Plots for Jaanesh using Uniform Frequency Bands



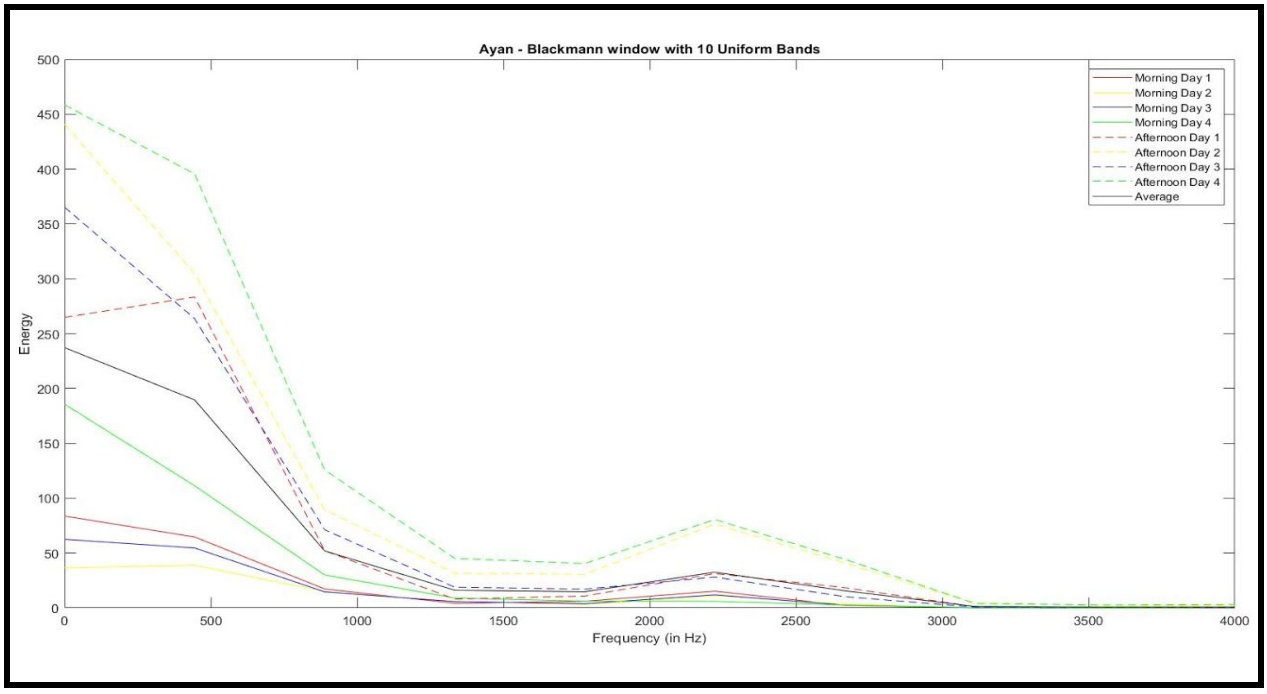



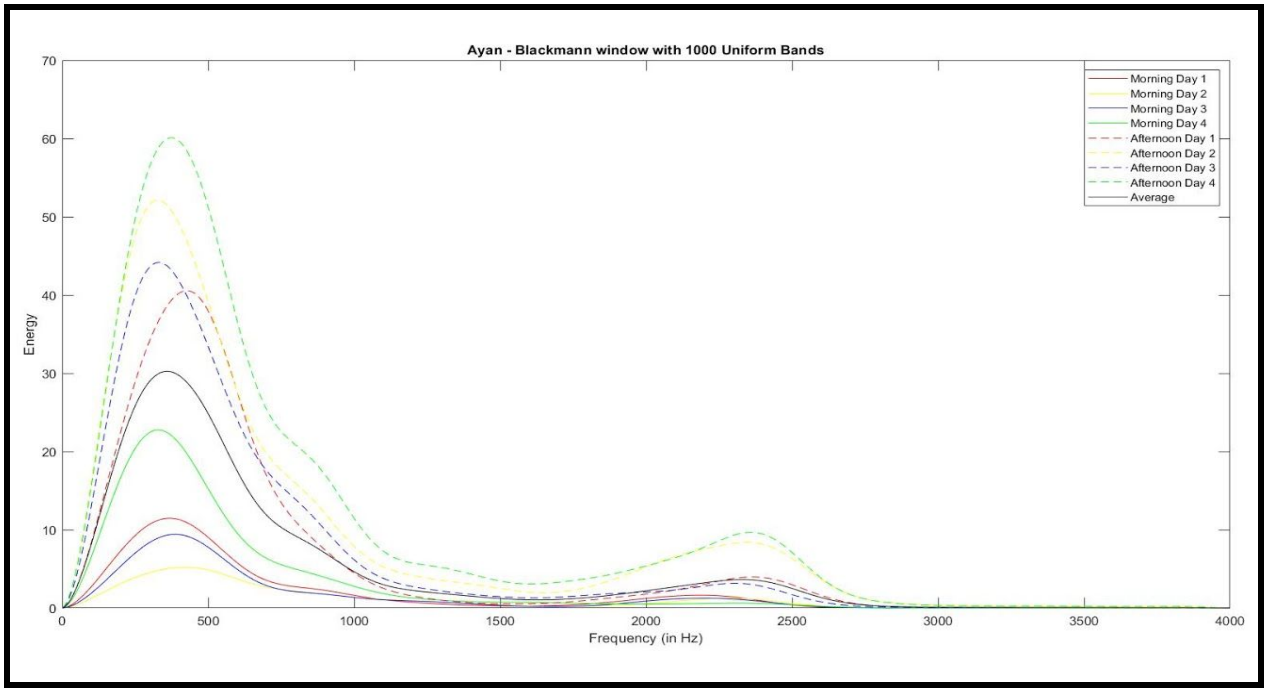
## Energy-Frequency Plots for Jaanesh using Non-Uniform Frequency Bands



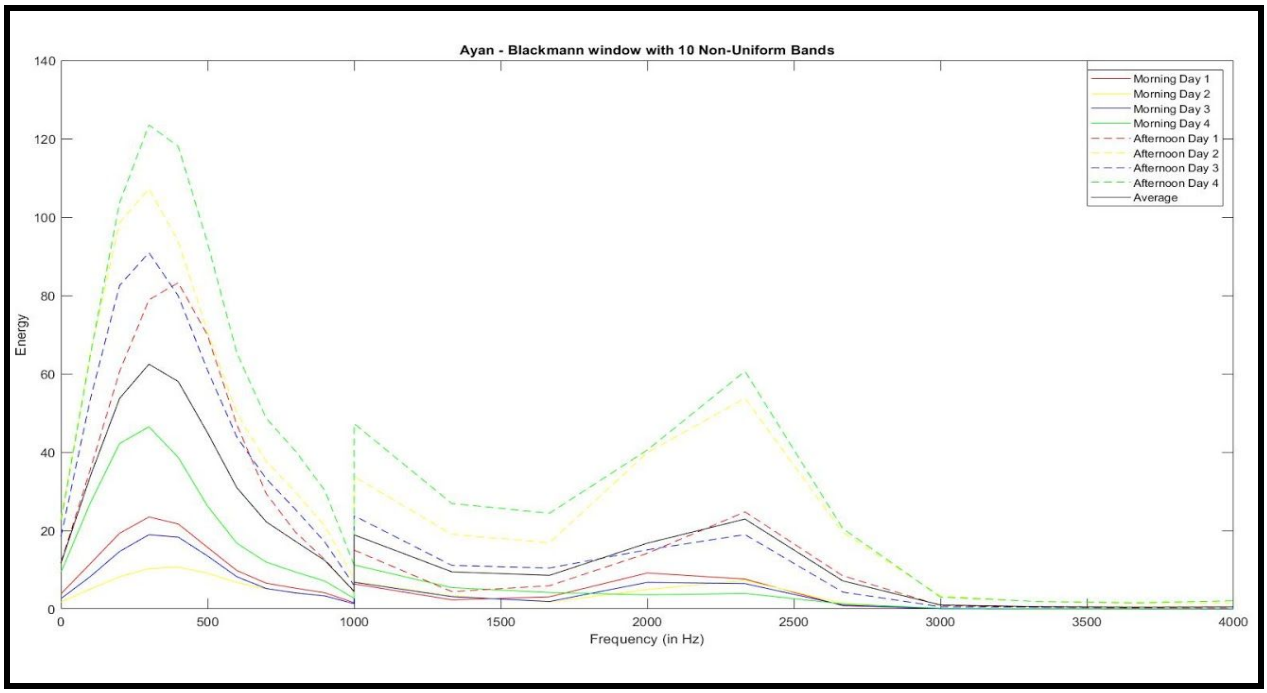



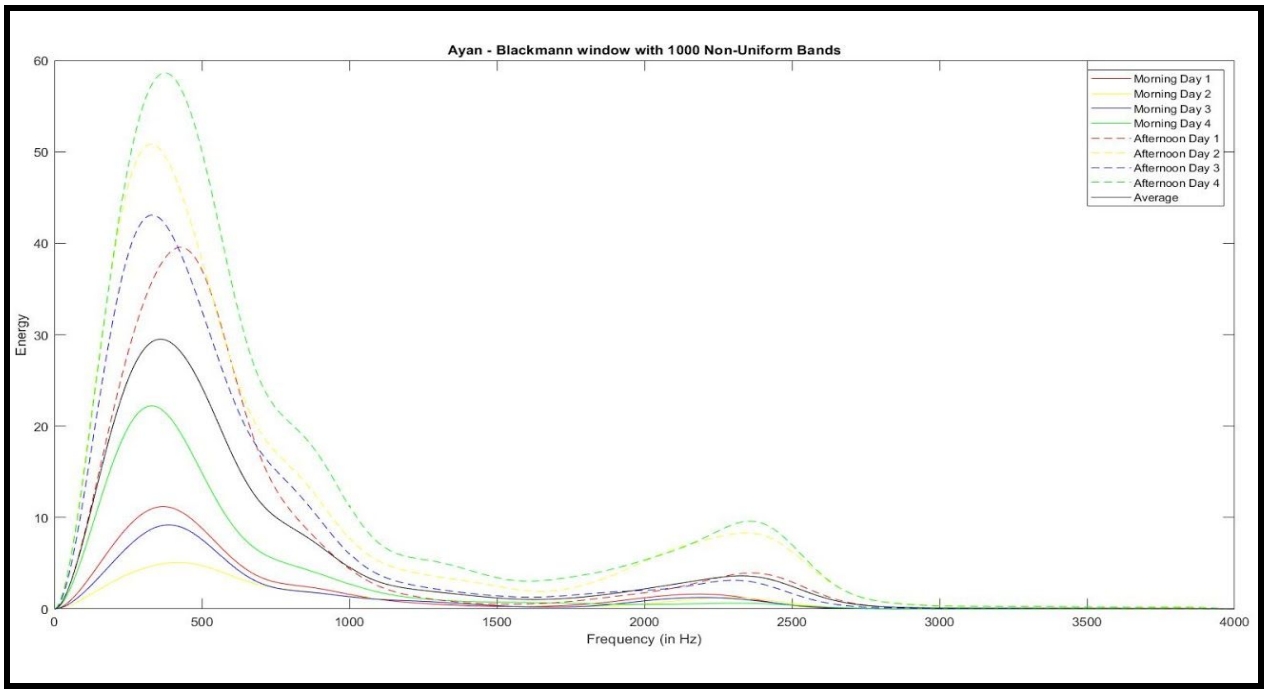
## Energy-Frequency Plots for Ayan using Uniform Frequency Bands



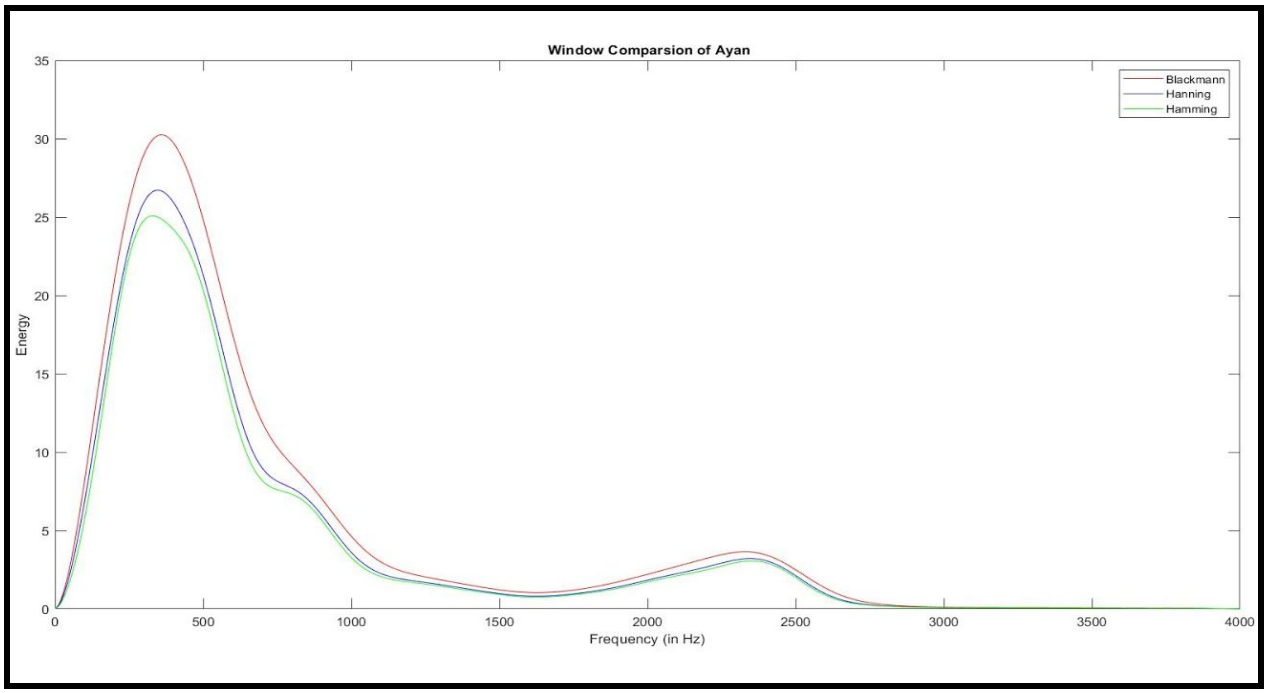
## Energy-Frequency Plots for Ayan using Non - Uniform Frequency Bands



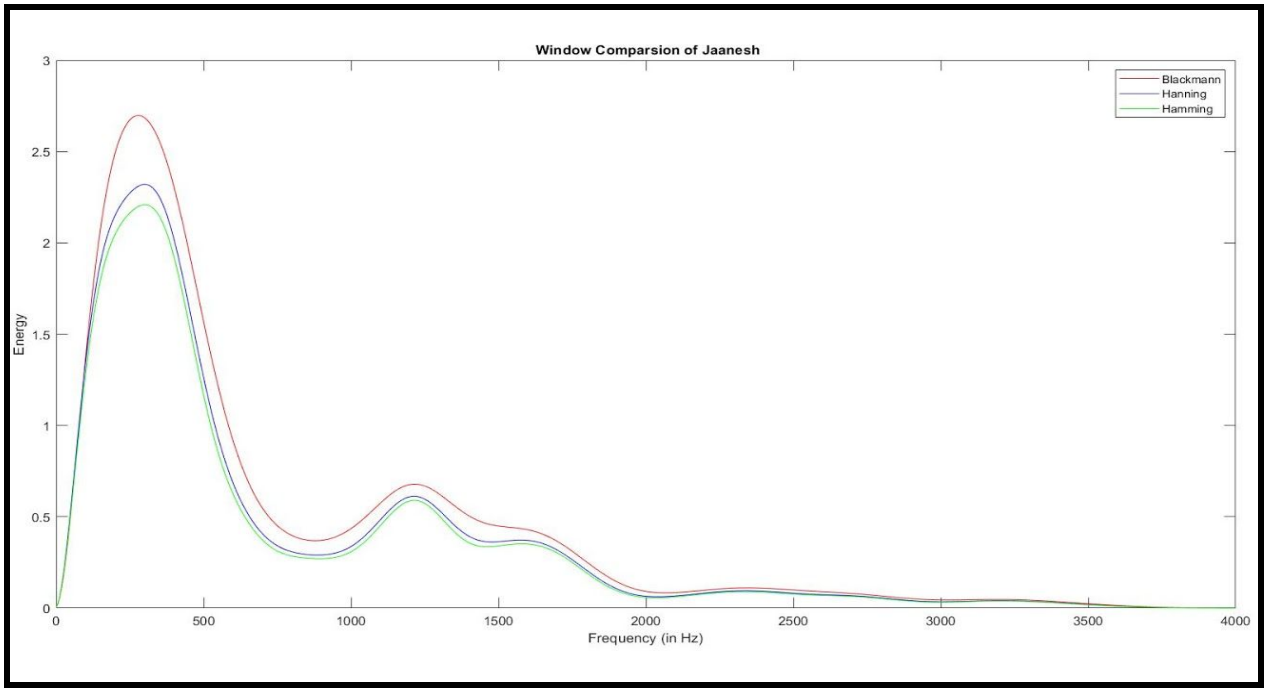




All these graphs show us that the effect of each filter bin is highly different and even the 8 samples of the same speaker are highly different and thus it is important to clearly design a classifier that takes into account the variance introduced by each speaker and yet clearly differentiate it from the subsequent speakers.

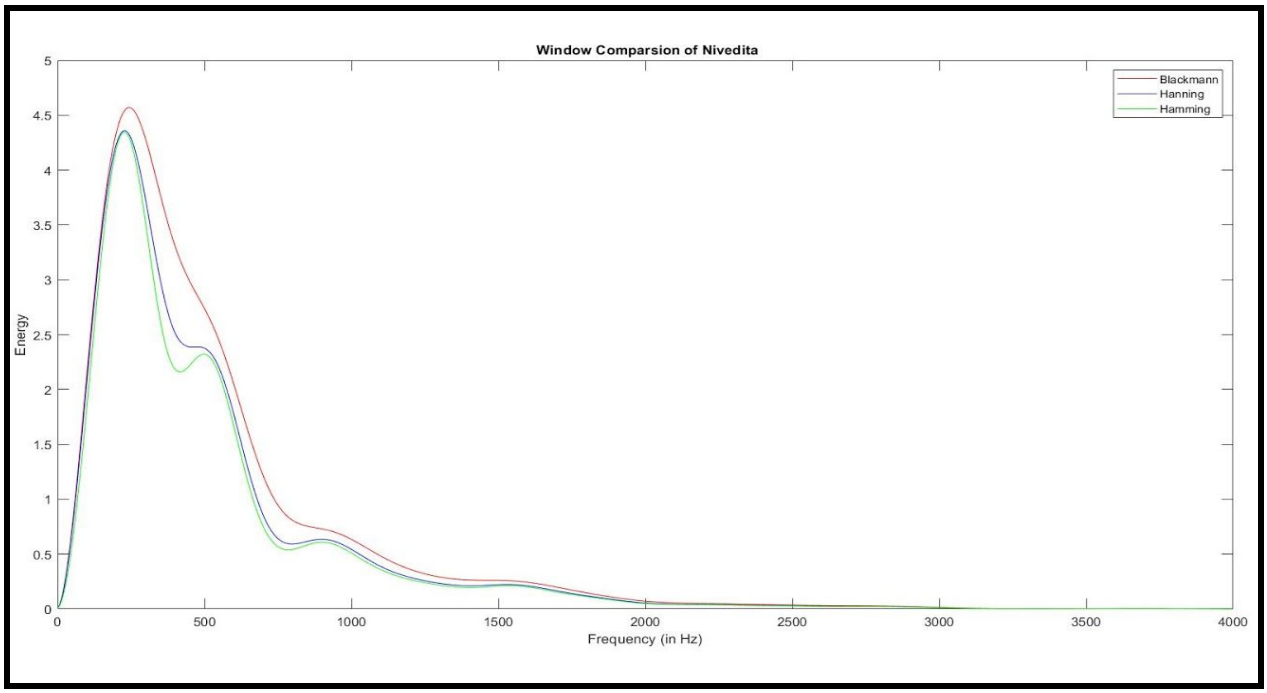
**Effect of window functions after filtering on different speaker samples:**



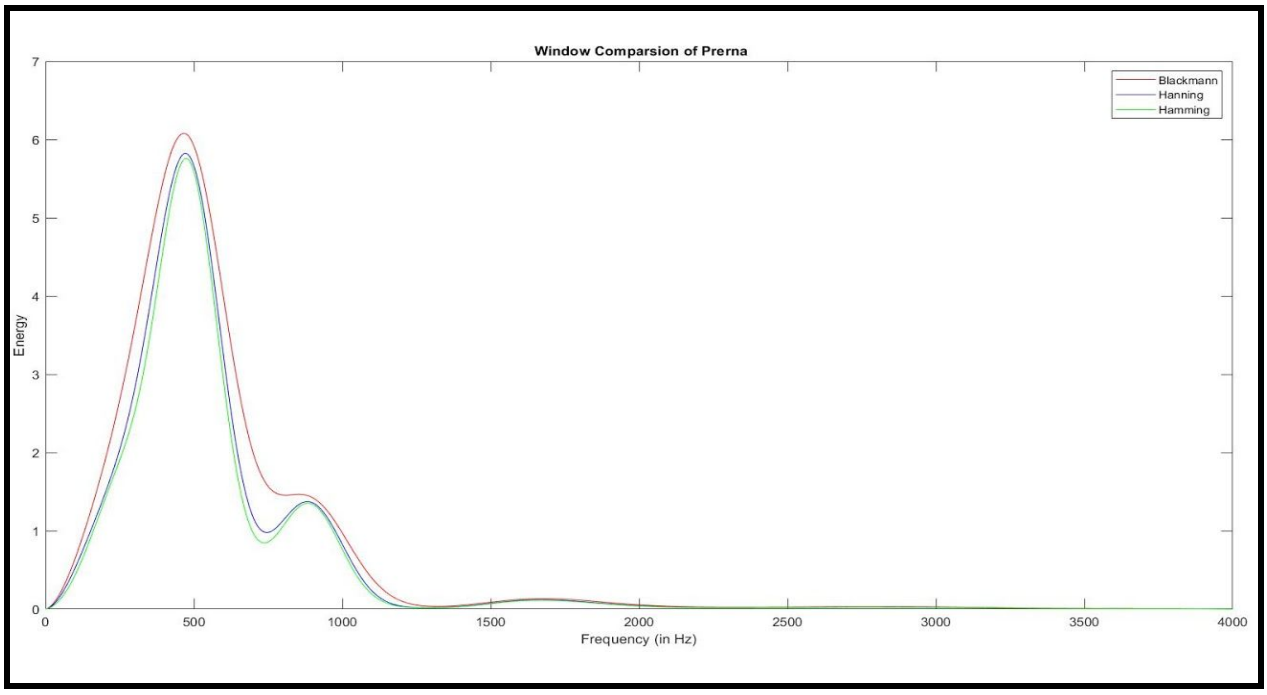

**Window Comparison of Ayan**



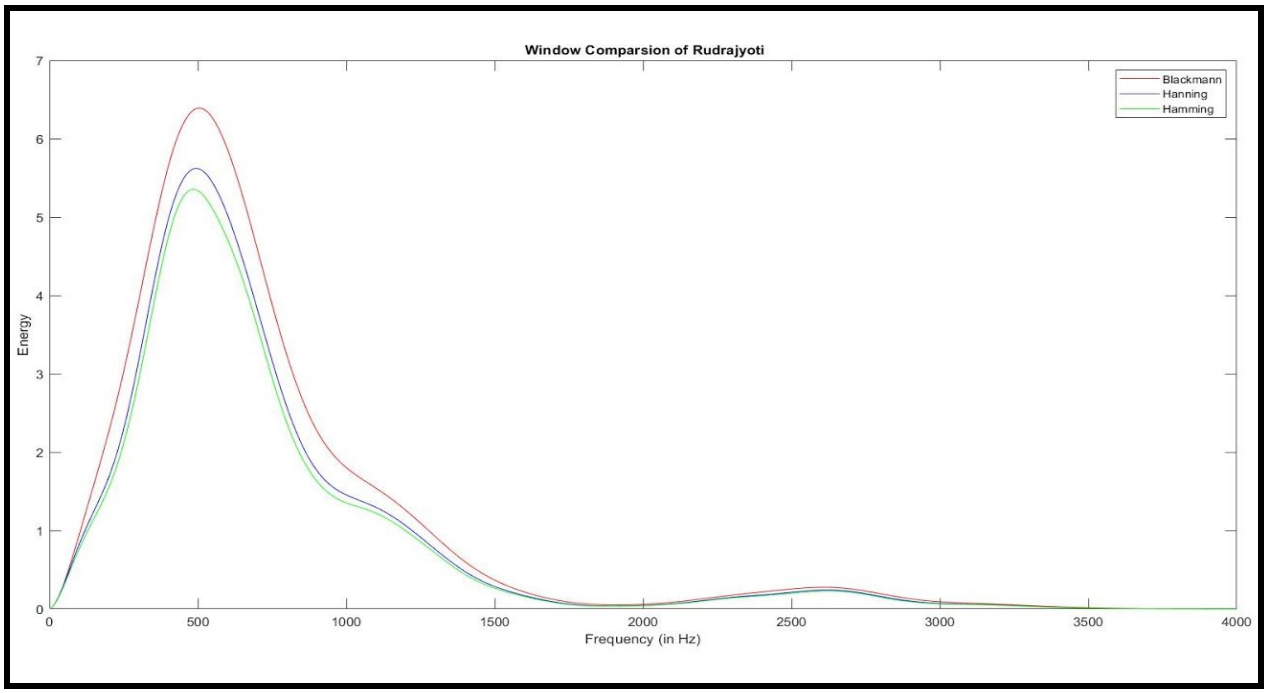

**Window Comparison of Jaanesh**



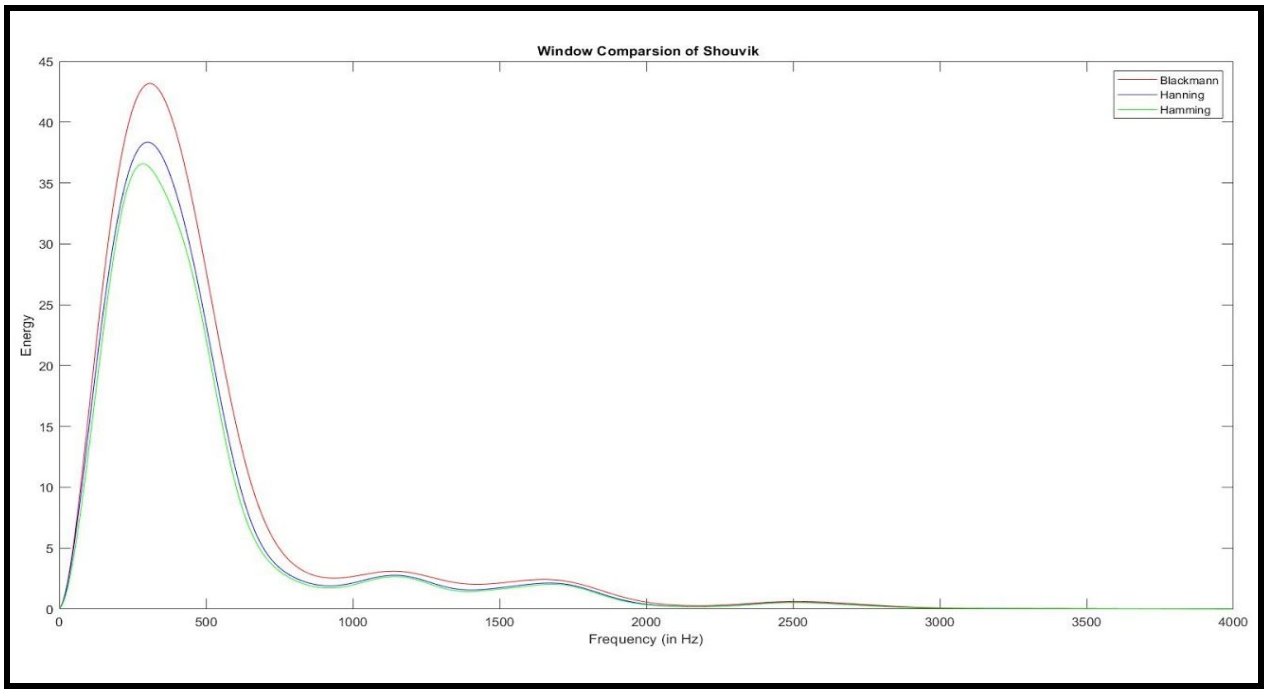

**Window Comparison of Nivedita**



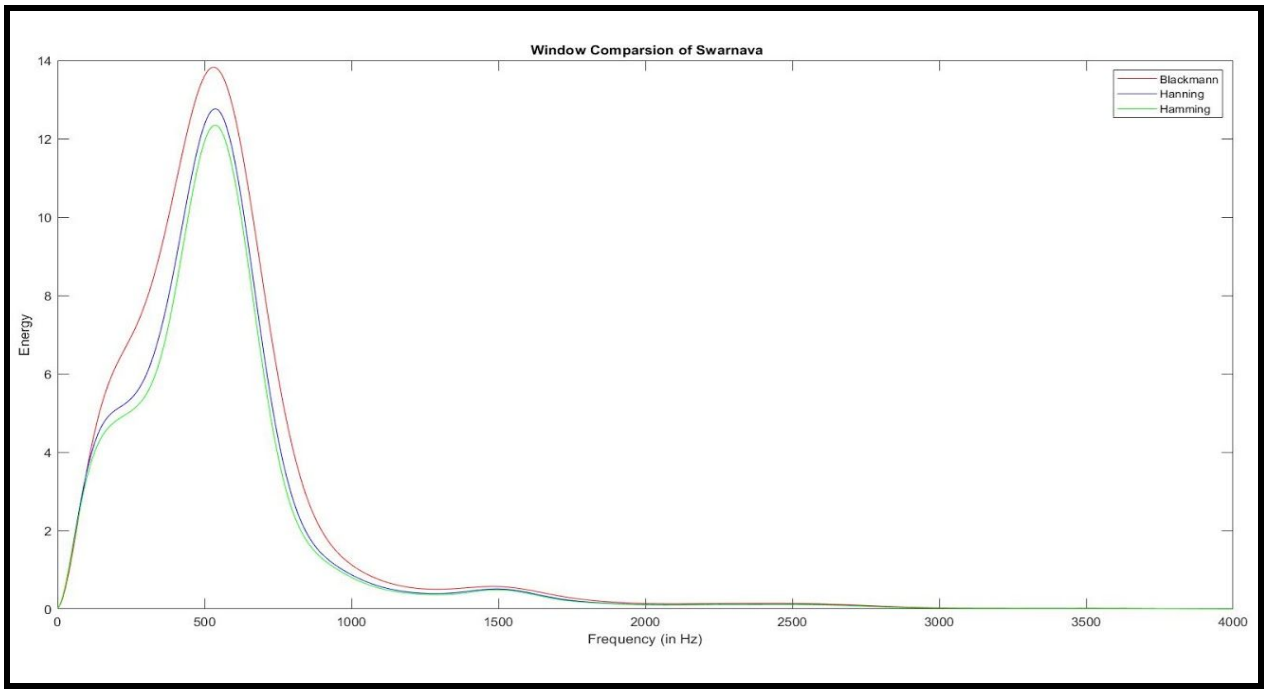

**Window Comparison of Prerna**




**Window Comparison of Rudrajyoti**




**Window Comparison of Shouvik**



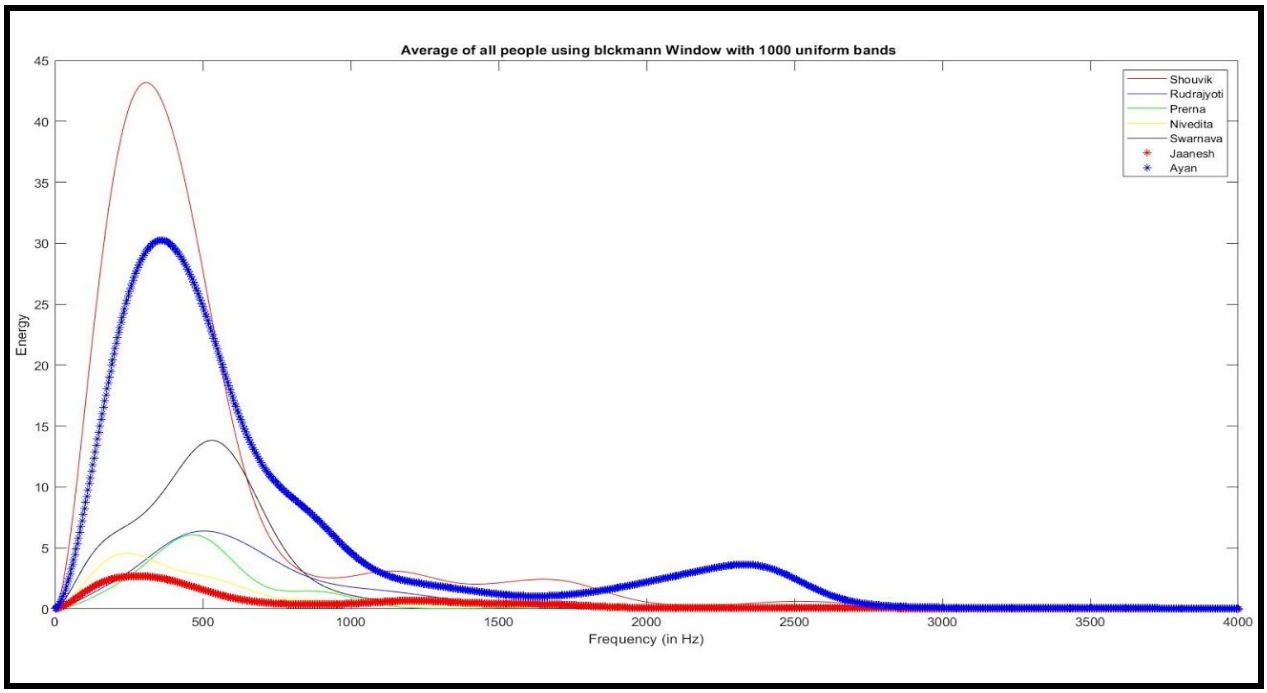

**Window Comparison of Swarnava**

From the above graphs, it is quite clear that different windows don't differ much in their effect. The blackmann window is not able to capture very quick troughs. Other than that all the windows are quite similar and thus Blackmann Window has been used as the default window in the subsequent sections.

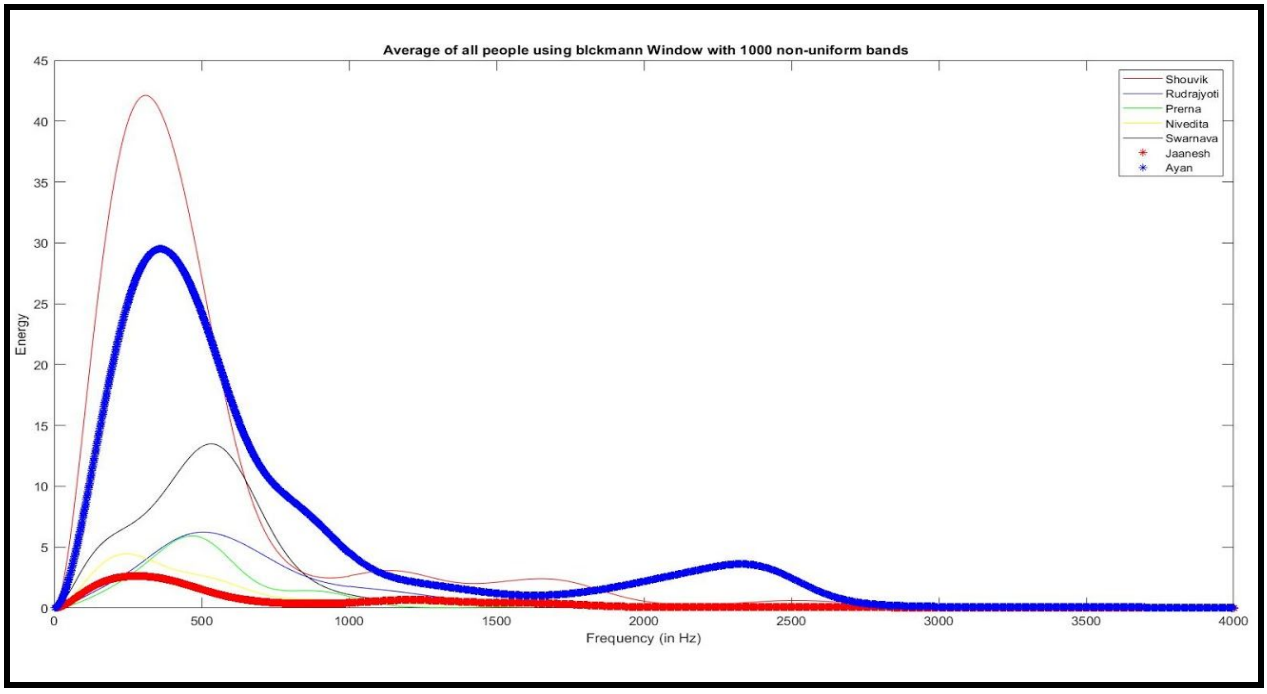
Next, we plot the average energy of each filter band for each speaker to get a comprehensive view of how the energy varies corresponding to a speaker through a filter band. The average energy of a filter bank was calculated by taking the average of the energies of the eight samples corresponding to a particular speaker. It can be clearly seen that there are similarities between speakers in terms of the peak energy at a particular frequency and thus just the frequency content representing the maximum energy can not be used to identify the speaker and it requires to incorporate much more detail into building the classifier.

The plots are shown in the next pages.

## Plots showing comparison between all speakers




*Average Energy - Frequency plots for each speaker using 1000 uniformly spaced filters*

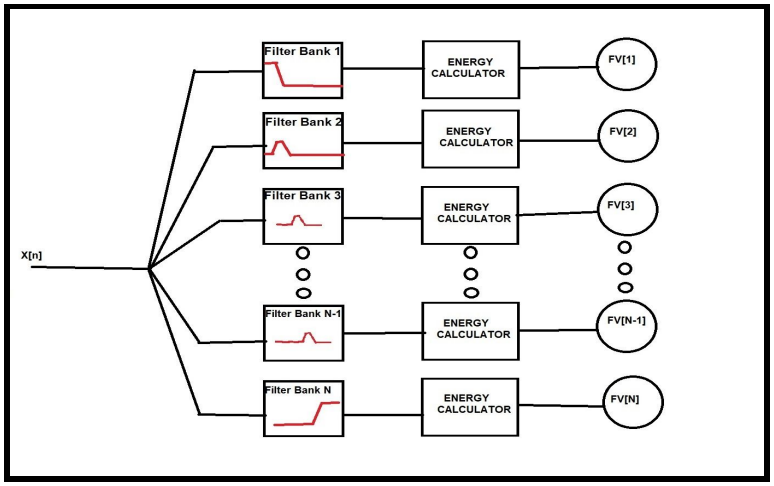



*Average Energy - Frequency plots for each speaker using 1000 non - uniformly spaced filters*

# C) RESULTS for Part 1(Using Energy of Filter Bank as FVs)

**Procedure / Pseudo Code:**

1) Read the entire speech Signal.
2) Pass it through the designed filter bank
3) The energy of each filter bank is calculated and is used as the feature vector for that sample
4) A label is associated with the sample
5) Repeat the process until all the speech signals are processed
6) Normalize the feature vectors by subtracting the mean and dividing by standard deviation




The above step was repeated for all the speakers and thus we had 56 feature vectors collected ( 8 samples per speaker and 7 speakers). This was later used to train a KNN based classifier.

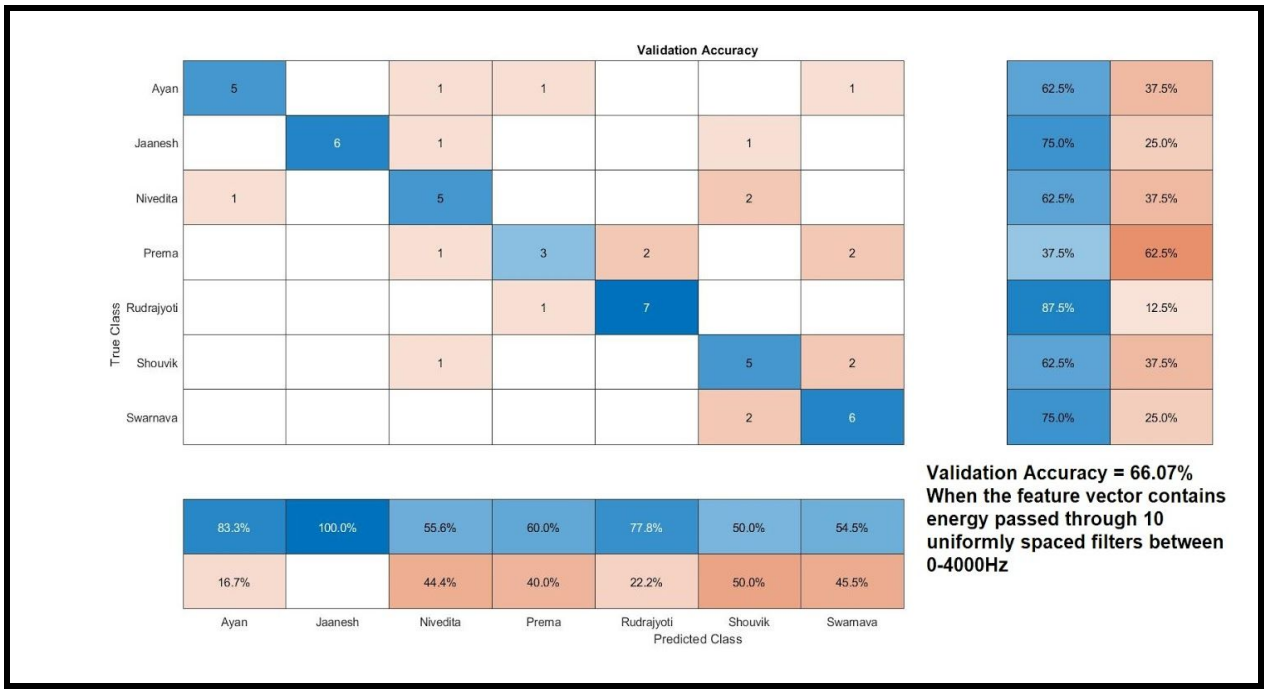
**The hyperparameters we have in this classifier are:-**

1. L = Number of Filter Banks
2. Distance function in fitcknn function => euclidean
3. Distance Weighting Function in fitcknn function => squaredinverse
4. Nearest Neighbours in fitcknn function => 1/3/5

Here the Filter Bank Configuration is varied, and for each specific configuration, k is varied as 1/3/5, and the results for the best k value for that specific filter bank configuration is shown.

## Best results obtained when k=1 for this specific filter bank configuration

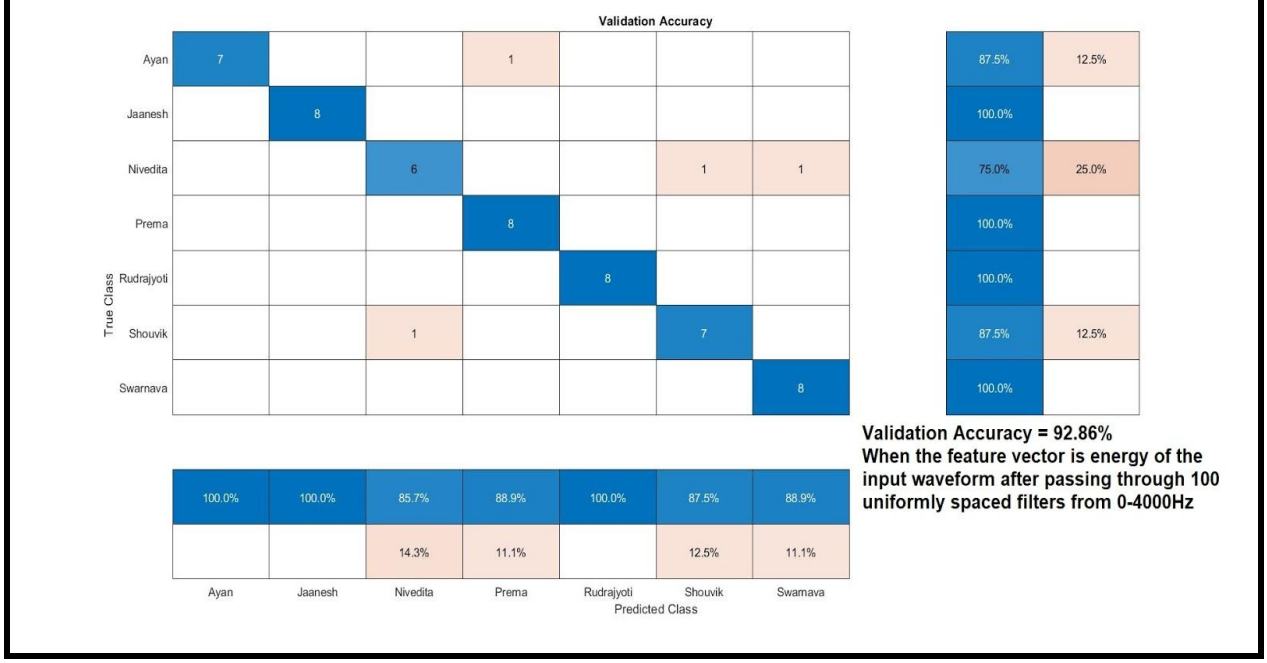
*Confusion plot*




**Validation Accuracy = 66.07%**
**When the feature vector contains energy passed through 10 uniformly spaced filters between 0-4000Hz**

## Best results obtained when k=5 for this specific filter bank configuration

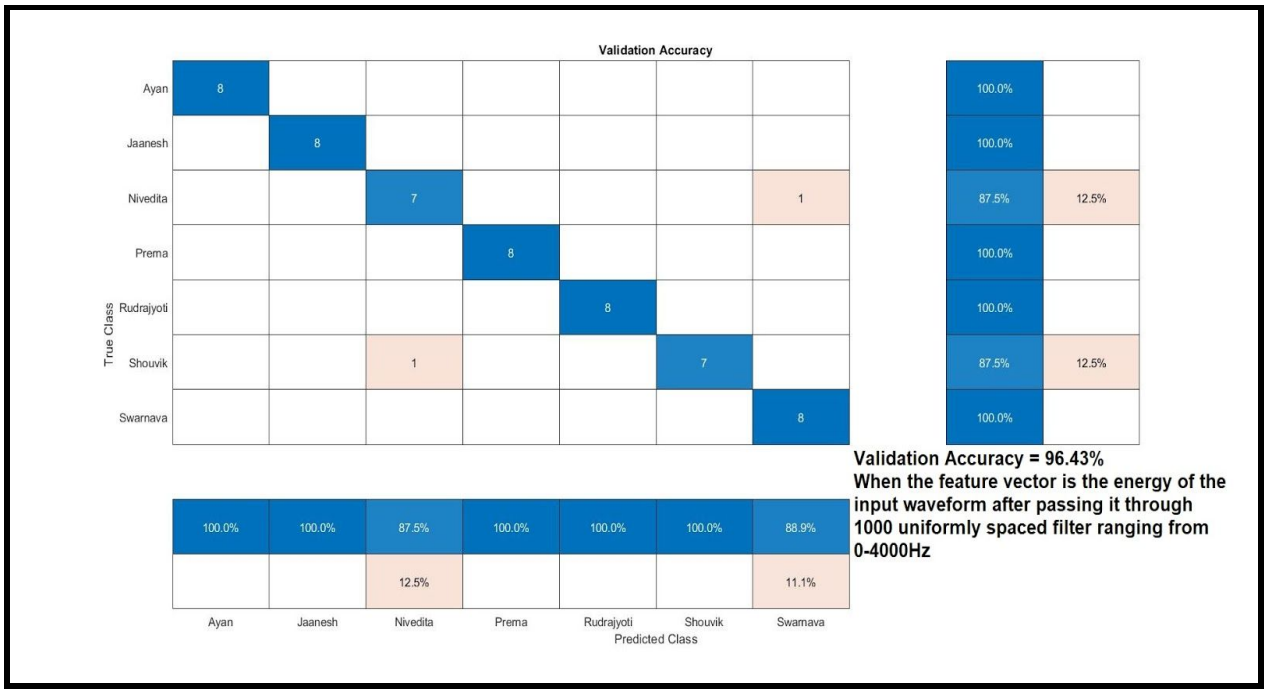
*Confusion plot*




**Validation Accuracy = 92.86%**
**When the feature vector is energy of the input waveform after passing through 100 uniformly spaced filters from 0-4000Hz**

## Best results obtained when k=1 for this specific filter bank configuration

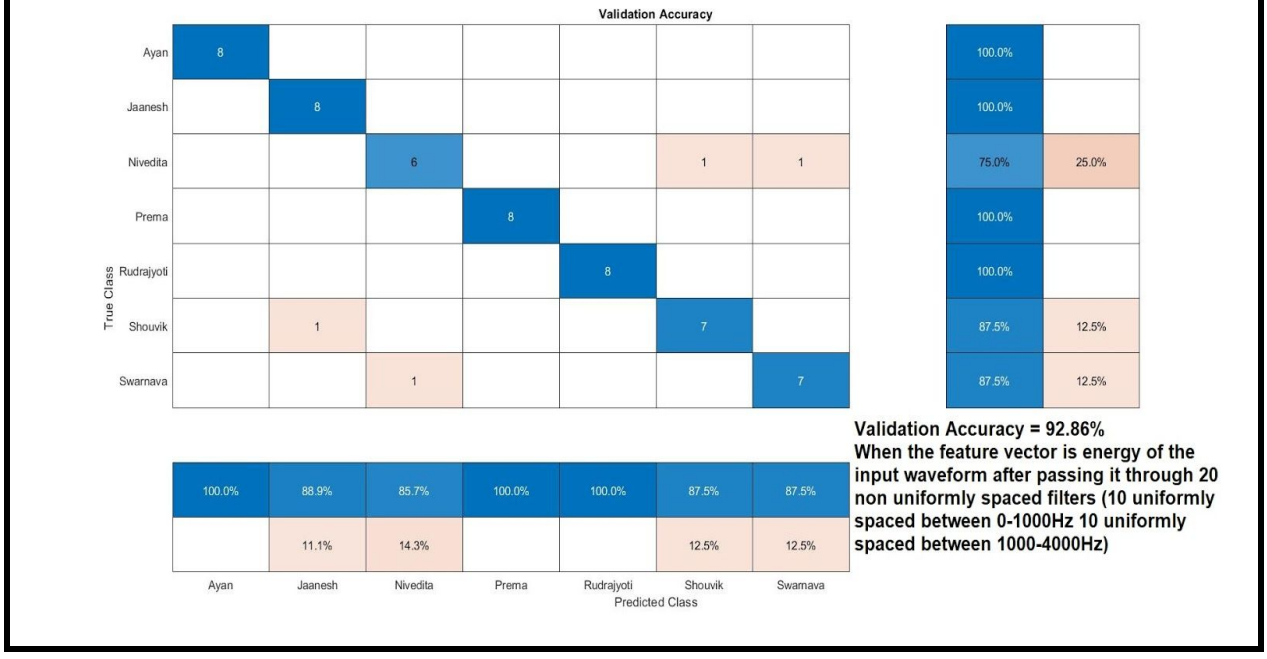
*Confusion plot*




**Validation Accuracy = 96.43%**
**When the feature vector is the energy of the input waveform after passing it through 1000 uniformly spaced filter ranging from 0-4000Hz**

## Best results obtained when k=1 for this specific filter bank configuration

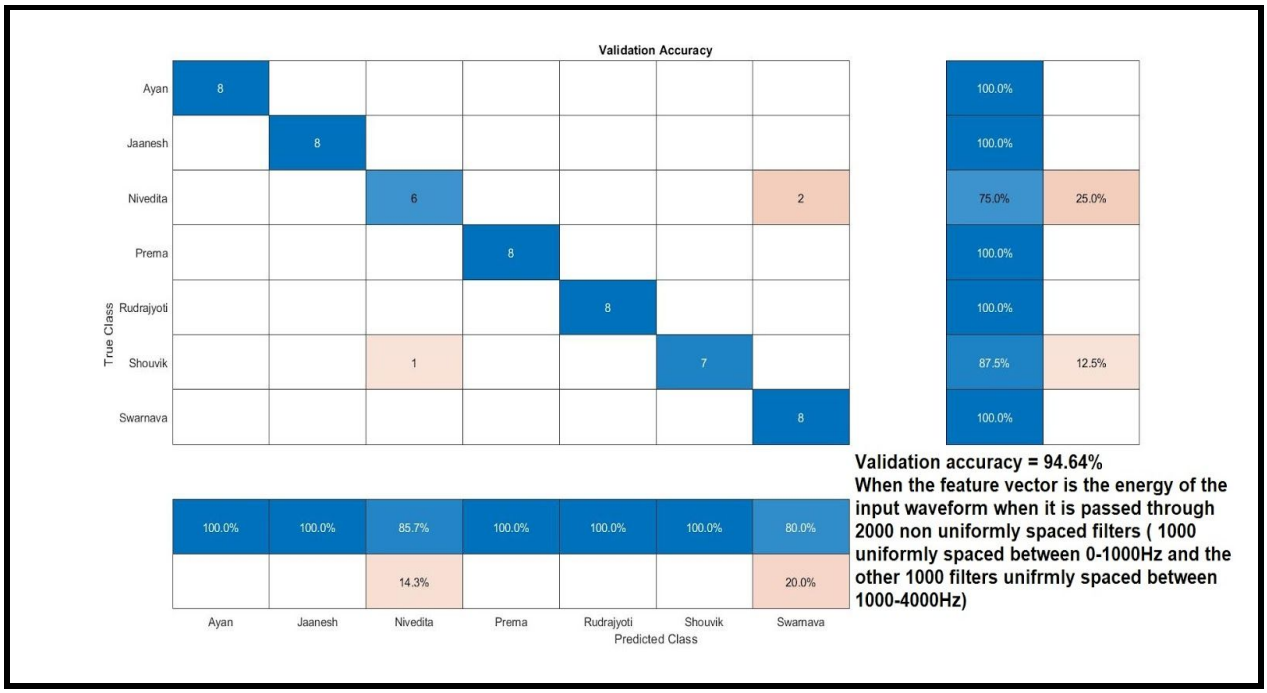
*Confusion plot*




**Validation Accuracy = 92.86%**
**When the feature vector is energy of the input waveform after passing it through 20 non uniformly spaced filters (10 uniformly spaced between 0-1000Hz 10 uniformly spaced between 1000-4000Hz)**

## Best results obtained when k=1 for this specific filter bank configuration

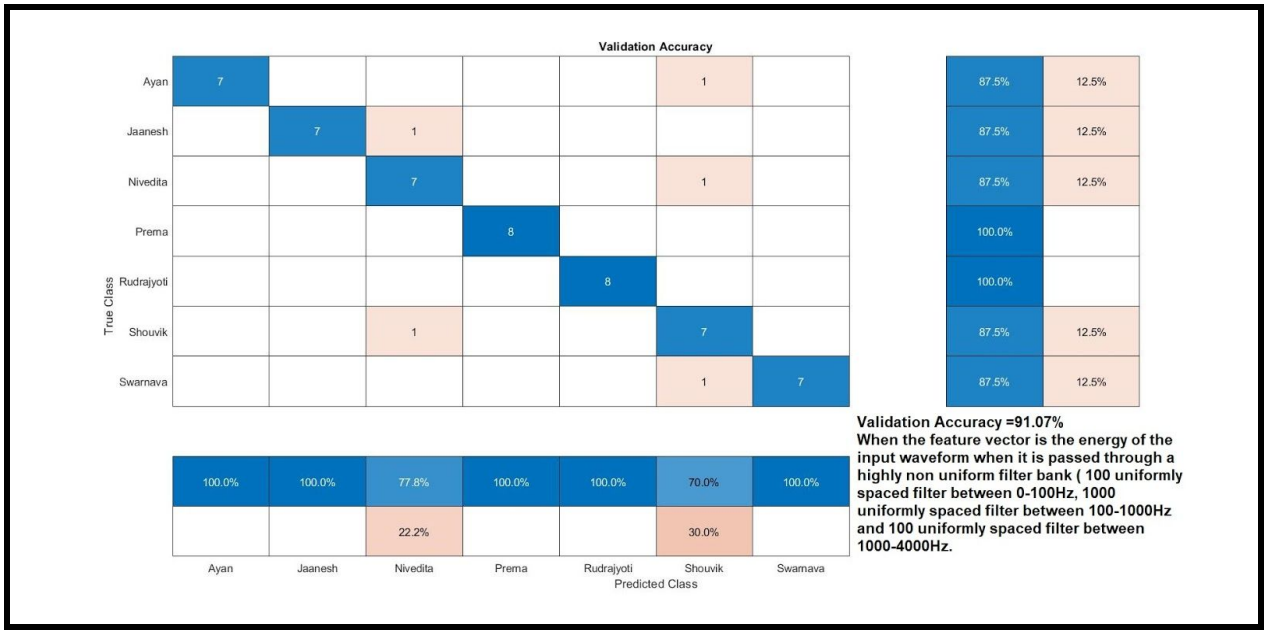
*Confusion plot*




**Validation accuracy = 94.64%**
**When the feature vector is the energy of the input waveform when it is passed through 2000 non uniformly spaced filters ( 1000 uniformly spaced between 0-1000Hz and the other 1000 filters unifrmly spaced between 1000-4000Hz)**

## Best results obtained when k=1 for this specific filter bank configuration

*Confusion plot*




**Validation Accuracy =91.07%**
**When the feature vector is the energy of the input waveform when it is passed through a highly non uniform filter bank ( 100 uniformly spaced filter between 0-100Hz, 1000 uniformly spaced filter between 100-1000Hz and 100 uniformly spaced filter between 1000-4000Hz.**

# (C) RESULTS for part 2 (Using MFCC and Pitch as FVs)

**Procedure / Pseudo Code:**

1) Read the entire speech signal
2) Detect voiced and unvoiced regions
3) Perform Amplitude Normalization
4) Divide the signal into small overlapping windows
5) Extract MFCC and Pitch for each of the windows
6) Combine MFCC and pitch to form a feature vector for each window.
7) Since, all of the windows belong to the same speaker, all of the feature vectors are labelled with the same speaker.
8) Repeat the process until all the speech files are processed
9) Normalize the feature vectors by subtracting the mean and dividing by standard deviation

**Hyper parameters used:**

1) Window Length: $0.03 * F_S = 0.03 * 8000 = 240$ samples
2) Overlap Length between Windows: $0.025 * F_S = 200$ samples
3) Power threshold used for Unvoiced region detection = -30dB
4) Number of zero crossings threshold = 1000
5) Distance metric used: Euclidean / City Block
6) Weighting method used: Inverse distance
7) Value of ‘k’ : k is chosen as 1 and 3 and the corresponding accuracies are observed

## Test Results:

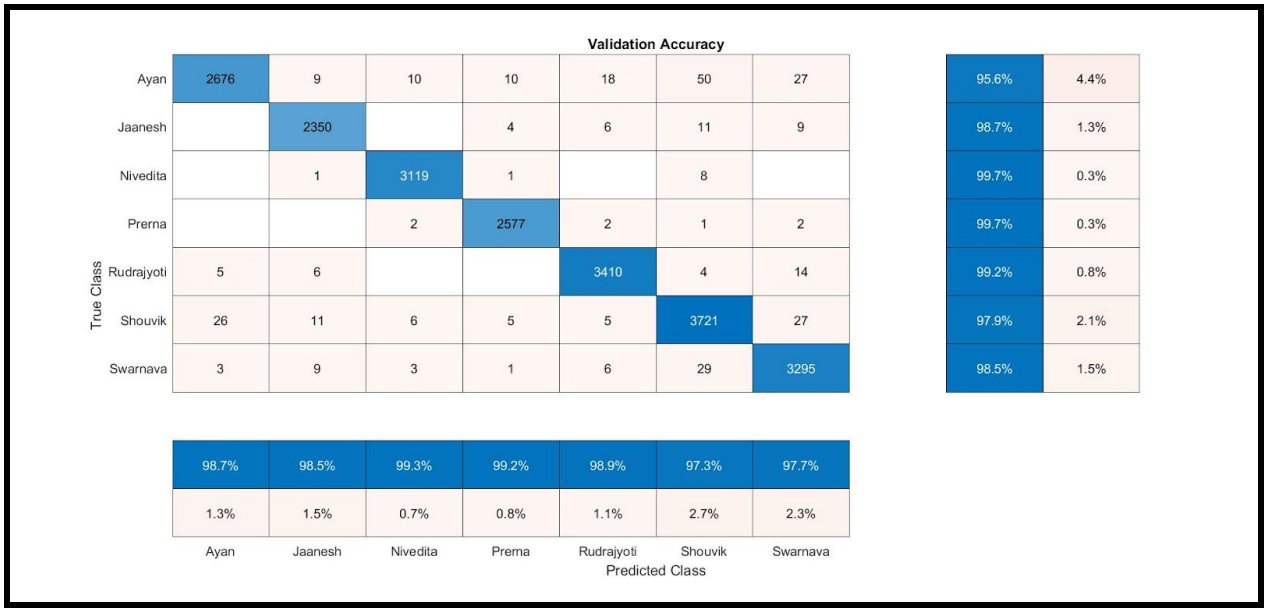
**<u>Combination 1:</u> k = 1, Distance Metric used: Euclidean**

*Confusion plot*


**K-fold cross validation set accuracy: 98.75%**

**<u>Combination 2:</u> k = 3, Distance Metric used: Euclidean**

*Confusion plot*




**K-fold cross validation set accuracy: 98.46% (slightly decreased)**

**<u>Combination 3:</u> k = 1, Distance Metric used: City block**

*Confusion plot*


**K-fold cross validation set accuracy: 98.83% (best till now)**

**<u>Combination 4:</u> k = 3, Distance Metric used: City block**

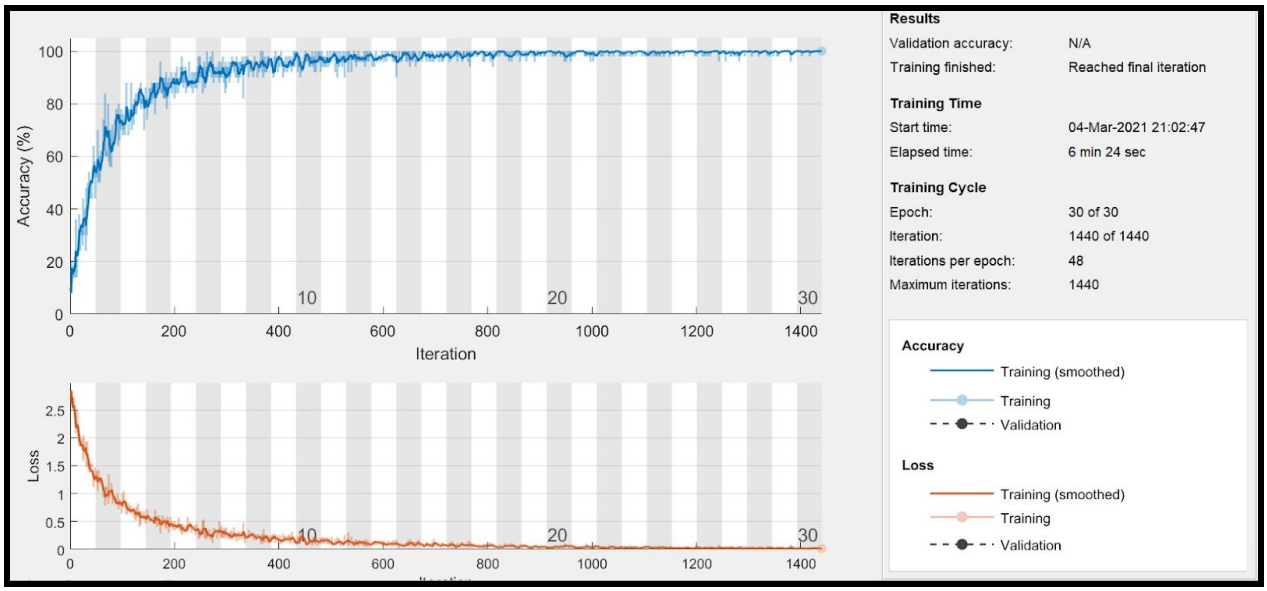
*Confusion plot*


**K-fold cross validation Accuracy = 98.62%**

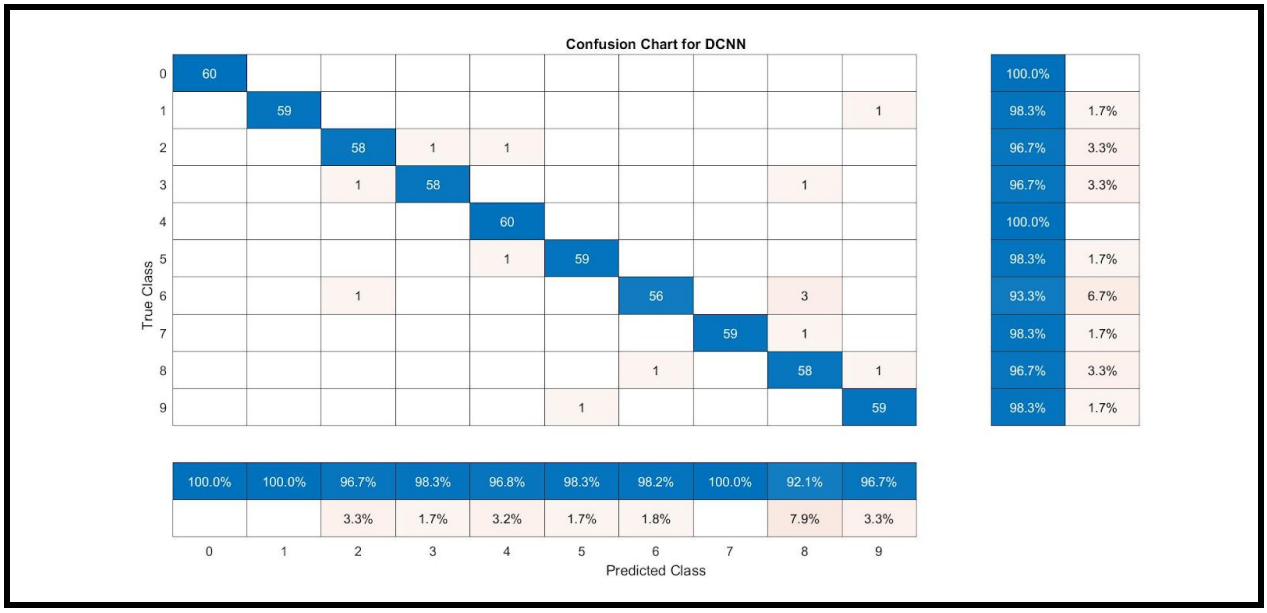
## Digit Identification using a CNN

Digit identification is a complex task. If we approach this task purely, from a signal processing point of view, then we will need a large number of template features. Hence, we have directly used a Convolutional Neural Network to generate the mapping from the feature space to the digit. The training data is used from the github repository: [Jakobovski/free-spoken-digit-dataset: A free audio dataset of spoken digits. Think MNIST for audio.](#) The spectrogram of each speech signal is computed and fed as input to the CNN.

Training Plot:




Confusion Chart: ( Validation Set accuracy = 97.67% )



**Testing on signals available to us:**

First, the signal is pre-processed to remove unvoiced regions. This method of removal of unvoiced regions is slightly different here. The envelope of the signal is calculated and part of the signal is removed where the p-p envelope value is less than 0.1. The above CNN was trained on windows of length 8192. Hence, a window of length of 8192 is taken and strided along the signal at a stride of 2200. The Mel Spectrogram is computed for each of the windows and the CNN classifies a digit on each of the windows.

Example of the CNN working on real signals:

Swarnava/Morn2.wav : Classification: '4' '3' '0' '9' '1' '7' '7'

In this example, we can see that it correctly classifies many digits (5/7). Unfortunately, it does not perform well on many other signals. It is heavily dependent on the position of the window, and since, we don't have any way to know when a digit begins and ends in the entire speech signal, the algorithm performs poorly. Also, the training set did not have any indian pronunciations and that can be 1 more reason for the misclassifications.

# Part 2: Pitch Extraction Algorithm

**Theory of Pitch:**

Pitch is an axiomatically important parameter for automatic speech recognition and understanding (ASRU), compression, and synthesis. It plays an eminent role in both the production and perception of speech. The pitch is defined as how "low" or "high" a harmonic or tone-like source is perceived. Although strictly speaking this is a perceptual property. To give a mathematical meaning to pitch we can consider it to be the fundamental frequency $F_0$ or the fundamental period of the harmonic component of the speech signal.

A general speech signal is denoted as follows:-

$s[n] = h[n] + \eta[n]$ *where* $h[n]$ *is the harmonic component and* $\eta[n]$ *is the noise component*

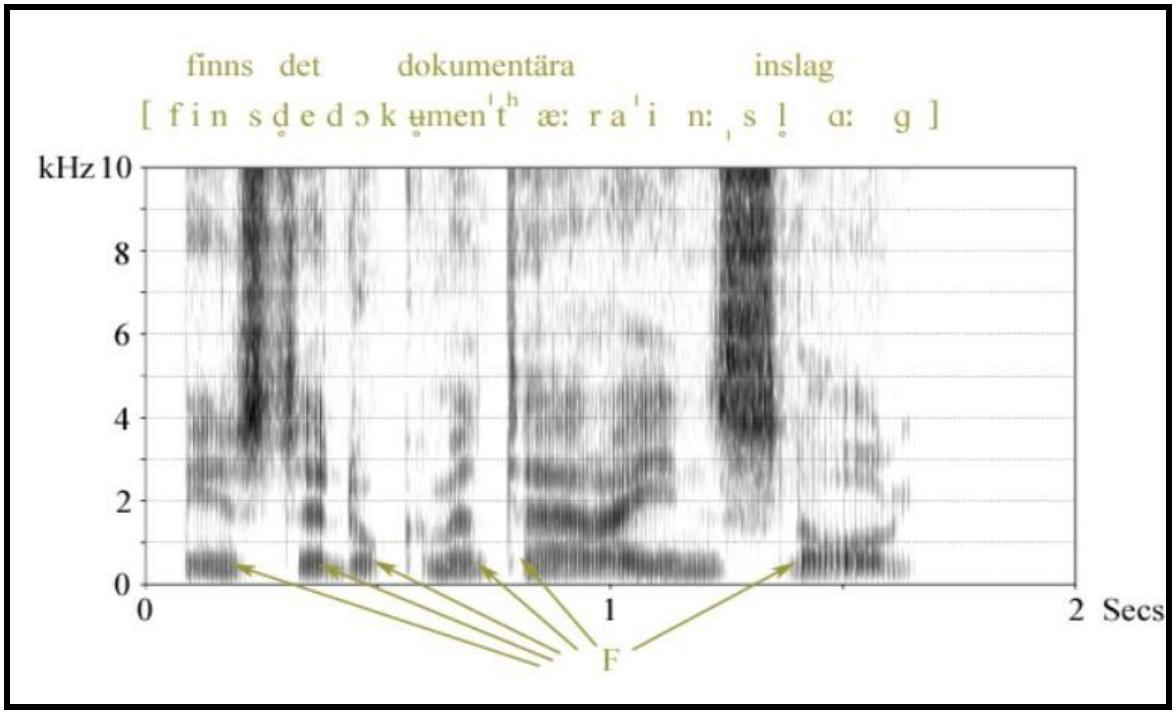
The h[n] can be written as follows:-

$$h[n] = \sum_{k=1}^{N} a_n sin(\frac{2\pi k F_o n}{F_s} + \phi_n)$$

Here $F_o$ is the fundamental pitch of the signal and $F_S$ is the sampling frequency, $a_n$ are the individual amplitudes and $\phi_n$ are additional phases for the individual partial tones. Unfortunately in real world signals like speech typically neither the amplitudes nor the fundamental frequency stay constant over the whole duration of the signal. But when looking closer at for e.g. speech, we see that these parameters normally only change slowly over time.

This behaviour gives us the possibility to assume that the parameters stay constant if we compartmentalize the signals into small enough sections in time. Such signals are called quasi-stationary. So the first step towards a pitch estimation is to divide the signal into small enough blocks (or windows).

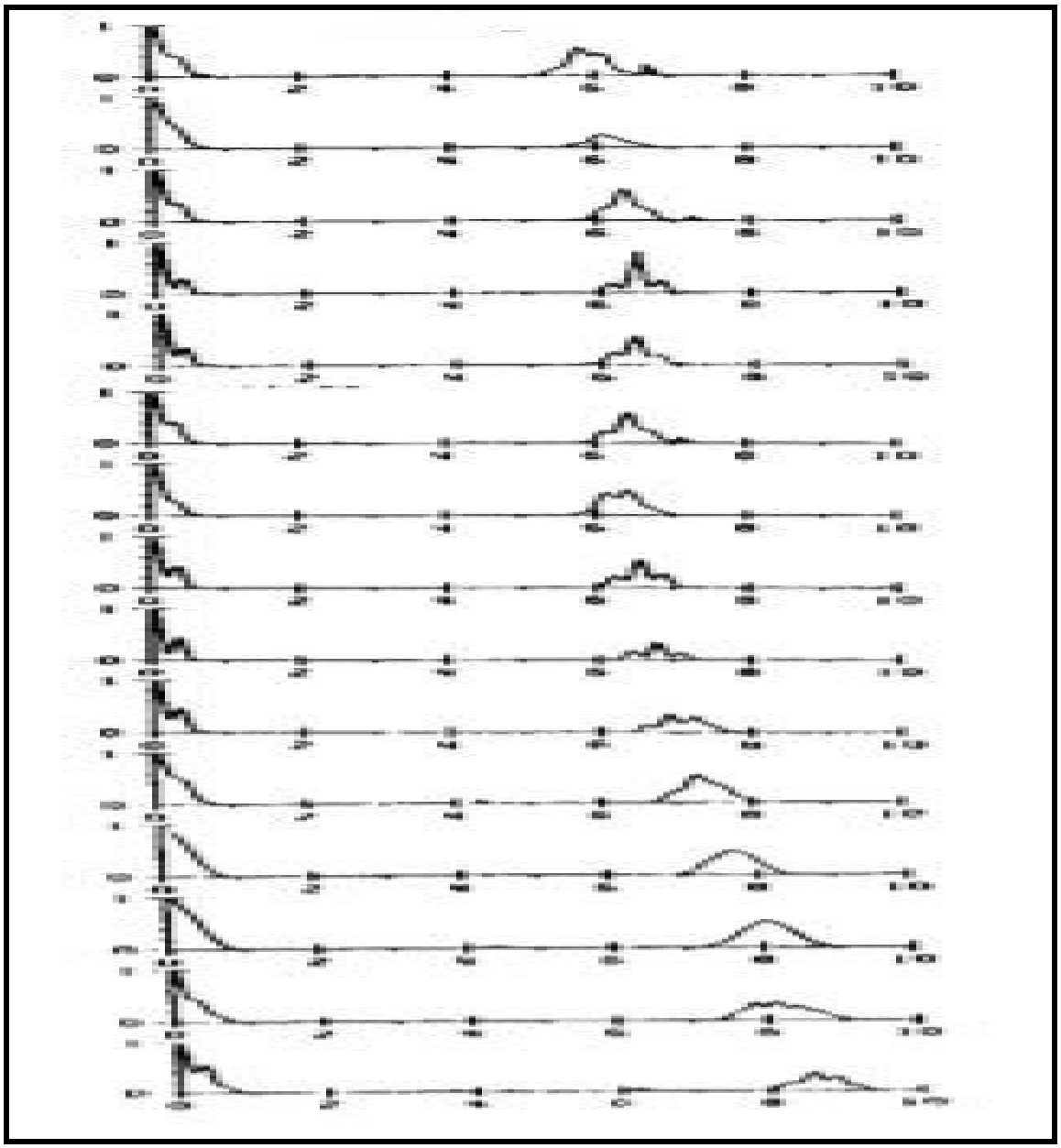
For reliable pitch determination it is necessary to remove the formant structure of the speech signal. A formant is a concentration of acoustic energy around a particular frequency in the speech wave. There are several formants, each at a different frequency, roughly one in each 1000Hz band. Or, to put it differently, formants occur at roughly 1000Hz intervals. Each formant corresponds to a resonance in the vocal tract. Following is the figure that highlights the formant structure:-




The dark bands are the formant formation while the white spaces are the anti formant formation. This formant structure can be avoided by using the centre clipping method. The centre clipping method is used to locate an amplitude level for each section of the speech signal such that any level below the threshold is equated to zero thereby reducing the formation of the formant but at the same time retaining the pitch information.

Then a m-delay auto correlation is calculated for the centre clipped signal. Since the segment is assumed to be periodic with a period P the autocorrelation would obtain a maximum at the lag value = P.

$R_{XX}[m] = \frac{1}{N}\sum_{k=m}^{N-1} x[k]x[k-l]$ where N is the length of the time segment l is the delay period and when l=P we get the maximum value



This way the autocorrelation function can give the pitch value. The important thing is to decide the centre clipping method to remove the formant formation. There are errors associated with this method and thus there are modern methods to multiply the centre clipped signals with window functions(ADMF methods) that can clearly highlight the maximum and thereby helps to find the pitch of the signal.

There are other methods wherein Direct Cosine Spectrum of power of input sound waveform can help to find the pitch. There are methods wherein the sampling rate is increased to find the pitch of the waveform.

## References:-


1. Pitch extraction algorithm for voice recognition applications by R. Sankar. Department of Electrical Engineering, University of South Florida, Tampa, FL, USA
2. A PITCH EXTRACTION ALGORITHM IN NOISE BASED ON TEMPORAL AND SPECTRAL REPRESENTATIONS C. Shahnaz, Student Member, IEEE, W. -P. Zhu, Senior Member, IEEE, and M. O. Ahmad, Fellow, IEEE Centre for Signal Processing and Communications, Dept. of Electrical and Comp
3. Novel pitch extraction methods using average magnitude difference function (AMDF) for LPC Speech Coders in Noisy Environments Suma S.A., Dr. K.S.Gurumurthy University Visvesvaraya College of Engineering, City Campus , K.R. Circle Bangalore 01, India.