Detection of Notes in Indian Classical Music

Mohit Jain {mohit.jain@research.iiit.ac.in} II Prof. Navjyoti Singh {navjyoti@iiit.ac.in}

1. Introduction

Indian (Hindustani) Classical Music has been a major challenge for all note and pitch detection algorithms, mainly due the large number of intricacies involved with this class of music in the form of microtones and the relations and transitions between the microtones. Extraction of specific features for the detection of these microtones is what acts as a bottleneck for most of these algorithms as manually crafting these features becomes extremely difficult. Deep learning techniques have started gaining a lot of popularity in the recent past for the tasks of feature extraction and pattern classification and this project is an attempt at using Deep Learning techniques along with the prominent Algorithms in the domain of note detection along with Shallow Learning techniques to tackle the problem of Note Detection in Indian Classical Music.

Most deep learning techniques require huge amounts of annotated data in order to extract the corresponding features to the classes being sought. Due to the unavailability of any such datasets, specifically for Indian Classical Music, the first idea discussed is the curation of an annotated dataset (section 2) using Nearest Neighbour Techniques (section 3). Next, this dataset created would be used to extract deep-features using a Neural Network having one-class-per-note in the output layer (section 4). Finally, we discuss the results and the future work that can be done in this project to further improve the performance.

2. Dataset Curation

2.1 Distinction of Classes - Chromatic Scale

The first step to create a dataset is to decide on a meta-data representation of the components. Here, the components of our dataset would be short musical clips segmented on the basis of a classical music note being played on an instrument. Therefore, the categories to demarcate these music-clips would be the notes of the Indian Classical Music System. Namely, *Sa, Re, Ga, Ma, Pa, Dha, Ni* (the 7 base/*shuddha* notes of Indian Classical Music) and the slight deviations from these base notes forming the *tivra & komal* notes (half step up & down respectively from the base note). Giving us a total of 12 notes as described in the table (fig. 2.1.1)

Indian Music is famous for its complex use of microtones. But for notation and explanation, we divide an octave into 12 semitones. We use a movable scale, which means that the octave can start anywhere in the pitch ranges. The starting point of the note is the root of the octave and all other notes are defined in the octave in relation to the root. The chromatic scale does not span

Note Name	Notation ID	Sol-fa Syllable	Full Name
sa	S	sa	Shadja
re (komal)	r	re	Rishabha (komal)
re (shuddha)	R	re	Rishabha (shuddha)
ga (komal)	g	ga	Gandhaara (komal)
ga (shuddha)	G	ga	Gandhaara (shuddha)
ma (shuddha)	m	ma	Madhyama (shuddha)
ma (teevra)	M	ma	Madhyama (teevra)
pa	P	pa	Panchama
dha (komal)	d	dha	Dhaivata (komal)
dha (shuddha)	D	dha	Dhaivata (shuddha)
ni (komal)	n	ni	Nishaada (komal)
ni (shuddha)	N	ni	Nishaada (shuddha)
sa	S’	sa	Shadja

Fig 2.1.1 - The Chromatic Scale in Indian (Hindustani) Classical Music [1]

the entire range of variations in the Indian Classical Music, however it is a good enough representation for most of the semitones involved in this intricate framework of music. We will assume the octaves used with the root having C as the keynote.

2.2 Detection of MIDI notes

MIDI notes tend to be much simpler to detect as compared to the Indian Classical Music Notes due to their their direct correlation with the frequency of the audio sample. We use Aubio[3], an audio labelling library, for this task. The MIDI notes are detected by computing the MFCC features (discussed in section 2.4) of the incoming audio samples at a fixed sample rate (generally 44100) in moving windows (generally of size 4096, taking hops of size 512).

These possible notes are then passed on to a note-confidence prediction algorithm, namely *PitchYinFFT* (discussed in section 2.3), to filter out the noise from actual notes.

2.3 PitchYinFFT Algorithm - Note confidence

This algorithm estimates the fundamental frequency corresponding to the melody of a monophonic music signal (eg. solo sitar, veena, etc). It is an implementation of the YinFFT algorithm[4], which is an optimised version of the Yin algorithm for computation in the frequency domain. It takes the input spectrum of the audio sample (preferably created with a hann window) and outputs the detected pitch and the confidence with which the pitch is detected. The PitchY-

inFFT algorithm comes bundled with the Aubio library and we would be using the same for computing the confidence of our MIDI-note-segments.

With the information of note-confidence, we set a threshold (set as 0.7 for this project) on the confidence prediction of the note and discard all predictions that are below the confidence measure of this threshold. Next, since the confidence measure for a note is divided across multiple time-steps taken in the audio sample segments, we have to combine together all continuos time-segments with similar MIDI note predictions and having prediction confidence above thresholds. We simply stack together all such continuous audio samples and for better coverage of the audio-note, pad the segment with one hop each before and after the predicted segments.

Finally, the MFCC feature is computed on the stacked-and-padded note-segment and this acts as our feature for the note to be used in the clustering techniques of section 3.

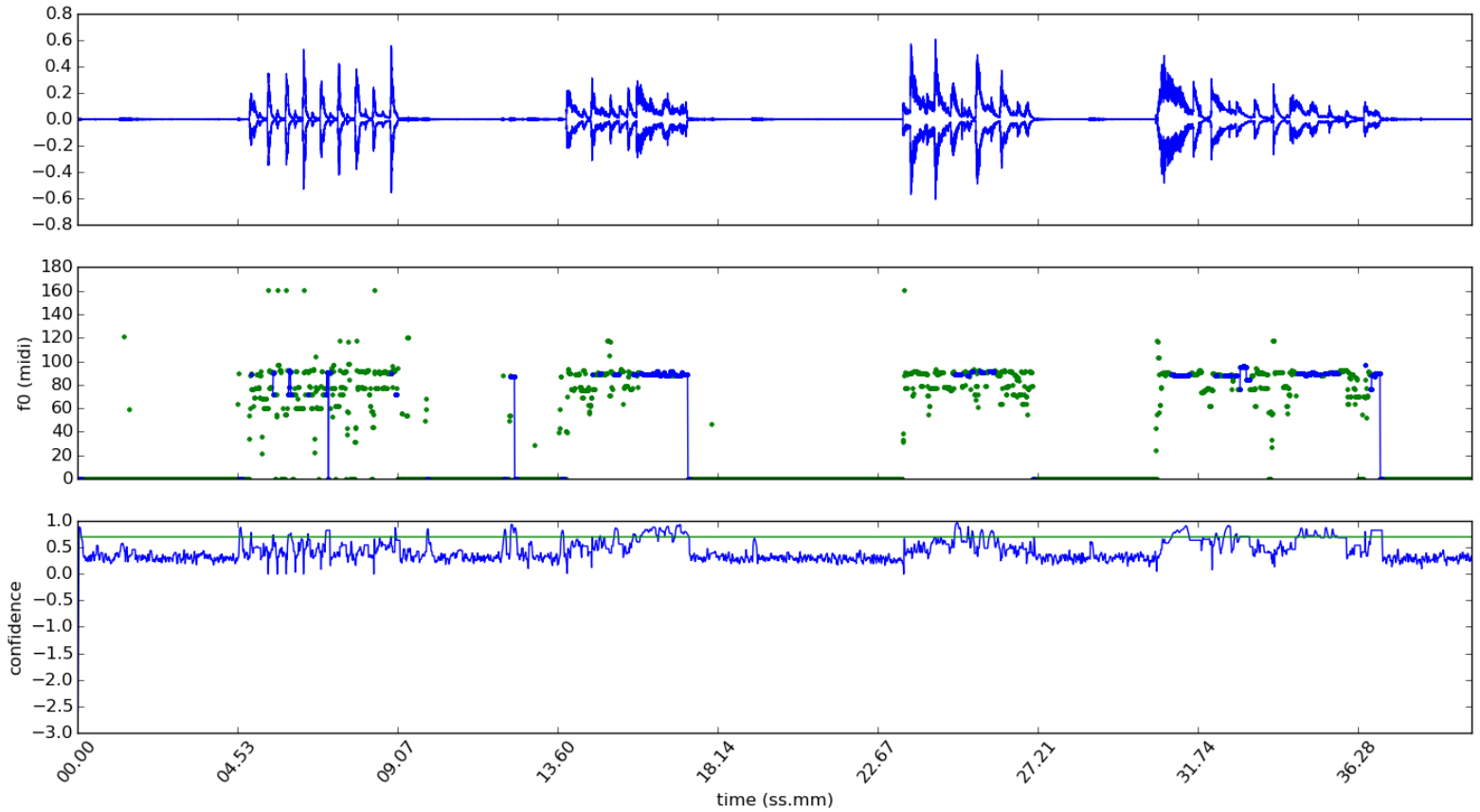

Fig. 2.3.1 - Visual Representation of the Note-Segmentation Pipeline

Top : Input audio waveforms.
Mid : Detected MIDI note frequencies.
Bottom : Corresponding confidence of the detected MIDI frequencies.

2.4 Extracting MFCC Features

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. [2]

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. The 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.

For computation of the MFCC features for our note audio segments, we would again be using the Aubio Library[3].

3. Nearest Neighbour Techniques

Nearest Neighbour techniques refer to the class of algorithms which use a distance metric (usually Euclidean) between the feature points (generally assumed of same dimensions) to perform the task of data-clustering on the input samples.

We use the k-nearest neighbour algorithm on the MFCC features of the detected note-segment audio samples. The computation of the MFCC over the segmented sample brings down the size of all the features to a uniform dimension (usually chosen as 13). Then, we can compute the Nearest Neighbours of the MFCCs of the audio samples and hence cluster our data according to chosen note-centres. These note-centres act as the ideal samples for the classical notes (hence the best case scenario is using 12 notes, one for each note of the Indian Classical Music System). All audio samples lying in the neighbourhood of the note-centre can be assumed to belong to the same category as that of the note-centre.

A better way to tackle the problem of clustering is to assign a score-matrix to each sample with 12 entries (the number of notes in our system of music) and instead of directly assigning the category of the note-centre to an input sample, we can simply increment the score of the note-centre's note class in the score matrix. Final classification of the note is given to the class with the maximum score. This technique also allows for multiple note-centres of the same note-class.

The choice of the note-centres is done manually by an expert of Indian Classical Music and the corresponding dataset is curated with these audio samples as the basis. The implementation of k-NN techniques is done using the Scikit-learn library[5].

To extend this system to note detection in a different Musical System, i.e. number of notes is different, or in the case where we don't have a labelled audio samples for every note in our expected note-classes or when we are not sure of the number of the note-classes there are in our target function, we can predict the number of categories obtained by computing the Nearest Neighbours for multiple iterations with an incremental approach, assuming one-more-class at each iteration (k=k+1, in k-NN based approach) and then compute the inter-class and intra-class variance of the clusters obtained. The best solution would be one which maximises inter-class variance while minimising intra-class variance.

A possible overall objective function could be chosen as,

Find K;
Such that, max(LOSS), where;
LOSS(K) = [inter-class-var(K)/intra-class-var(K)]

We now have a dataset of annotated audio samples of Indian Classical Music Notes. This dataset may have inconsistent tags attached as the corresponding note-label, however, we assume this to be simply the noise in the dataset.

4. Deep & Shallow Learning Techniques

Training a full-fledged deep learning architecture requires huge amounts of training data if trained using the general back-propagation techniques. Even with the methods discussed in the above sections, we would require annotations of a few hundred GBs of data to train a deep network with sufficient accuracy. Hence, we use the deep-learning techniques only for the task of feature extraction by training an auto-encoder network.

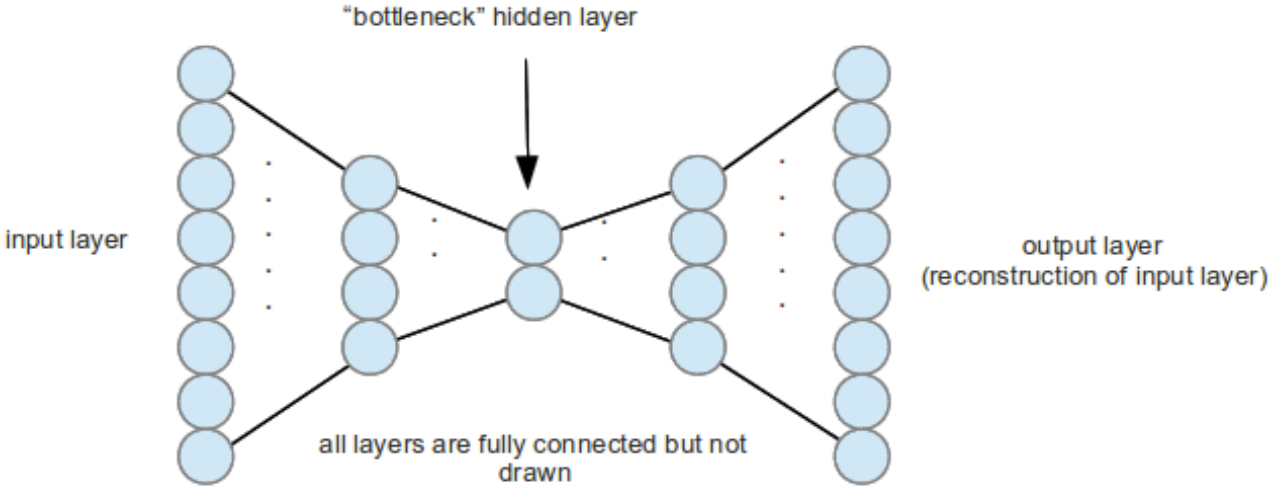

Fig. 4.1 - Visualization of an Auto-Encoder Network

The principle behind the bottleneck-shaped architecture is that the number of neurons in the middle layer is lower than in the other layers as shown in Fig. 4.1. A network with bottleneck can be structured in two sections: (i) Section 1 from the first layer to the bottleneck layer, with a gradual decrease of the number of neurons per layer, functions as an encoding or compression process which compacts relevant information and discards redundant information, and (ii) Section 2 from the bottleneck layer to the last layer with a gradual increase in the number of neurons per layer. The function of this part can be interpreted as a decoding process. An additional benefit of bottleneck architectures is that they can reduce overfitting by decreasing the system complexity. We consider the output of the last layer of this Auto-encoder network as our feature vector learnt by the network.

Now, we use the prominent shallow learning techniques to classify this intermediate feature vector into a specific category. For this project, we chose the Support Vector Machines (SVMs). The intermediate-reconstructed output vector of the deep architecture (Auto-encoder) is fed as an input to the SVM and this SVM classifies the note sample into one of the note categories.

5. Conclusions

In this project, we have shown how the task of data curation for a large audio sample dataset can be performed with minimal supervision. Furthermore, we have described an architecture of an amalgamation of Deep and Shallow learning techniques to tackle the problem Note detection in Indian Classical Music and shown how this framework can be easily extended to any other System of Music. The architecture described also involves creation of feature vectors for audio samples which can be easily classified by the state-of-the-art shallow learning techniques.

All codebase for these experiments is available online on Github : https://github.com/NightFury13/NoteClassification

6. Acknowledgements

I would like to thank Prof. Navjyoti Singh for giving me the opportunity to pursue this idea in the form of a Digital Humanities project and helping me increase my knowledge in this domain. I would also like to thank Achyuth Narayan for his continuous words of guidance and detailed technical discussion throughout the project.

7. References

[1] - Raag Hindustani : Notes : https://www.raag-hindustani.com/Notes.html
[2] - MFCC : Wikipedia : https://www.wikipedia.org/wiki/Mel-frequency_cepstrum
[3] - Aubio Library : https://www.aubio.org/
[4] - P.M. Brossier, “Automatic Annotation of Musical Audio for Interactive Applications”, QMUL, London, UK, 2007.
[5] - Scikit Learn Library : https://www.scikit-learn.org/