CLASSIFYING HINDUSTANI INDIAN RAGAS WITH CONVOLUTIONAL NEURAL NETWORKS

BY: RISHOV S. CHATTERJEE

May 14, 2019

Introduction

I have been exposed to Hindustani classical music ever since I was small. I was raised in an Indian household to immigrant parents and my mom is a vocalist and an exponent of Indian classical music. Indian classical music is categorized into two distinct forms: Hindustani and Carnatic, which are practiced in North and Southern India. Unlike western classical music, Indian classical music is very old form and typically doesn't have clear structures but largely depends on the performers or instrument players own elaboration of a melody. Indian classical music is defined by two basic elements – it must follow a Raga (classical mode), and a specific rhythm, the Taal [1]. Most compositions follow a Raga and I have noticed that even experts sometimes have difficulty of telling which Raga a particular song or composition is based on. This is particularly challenging for novices or beginners. Being a data science major, I quickly became attracted to this problem of "Raga detection". My intuition said that machine learning algorithms and techniques could help classify a composition into a main Raga on which it is based. Thus begins my journey to explore and hence this senior thesis.

I will mainly focus on North Indian form which is referred to as the Hin-

dustani classical music. Compositions in Hindustani classical music also are based on a drone, i.e., a continual pitch that sounds throughout the concert, which is tonic [2]. This drone acts as a point of reference as the performer is expected to come back to this home base after a flight of improvisation. The variations and complexity in Hindustani music stems from its use of notes that comprise a Raga. There are seven main musical notes (also called swaras) – Sa, Re, Ga, Ma, Pa, Dha and Ni – along with five intermediate notes (flats and sharps) referred to as "vikrit swaras". The seven notes are referred to as Shuddha and belongs to the saptak (a scale). The flat notes are called "komal" and the sharp notes are called "teevra". A raga consists of at least five notes, and each raga provides the musician with a musical framework within which to improvise [3, 4 5]. The specific notes within a raga can be reordered and improvised by the musician. Ragas range from small ragas like Bahar and Shahana that are not much more than songs to big ragas like Malkauns, Darbari and Yaman, which have great scope for improvisation and for which performances can last over an hour. Each raga traditionally has an emotional significance and symbolic associations such as with season, time and mood [6]. The raga is considered a means in Indian musical tradition to evoke certain feelings in an audience. Hundreds of raga are recognized in the classical tradition, of which about 30 are common [7].

The swaras in a raga can be played in three octaves, the first or lower octave starting from 130 Hz, then middle octave starting at 260 Hz; and upper octave from 520 Hz. The artists are allowed to improvise over the definitions

of raga to create their own renditions. If you listen to two performance of the same raga, they may sound strikingly different to novice ears, though they still retain the rules and defining qualities of ragas.

The rest of the thesis is organized as follows. In Chapter 2, we take a closer look at ragas to understand certain nuances and patterns they exhibit. In Chapter 3, we discuss Librosa, a python package for audio and music signal processing. In Chapter 4, we cover background and related work done on identifying Indian ragas using machine learning and other methods. In Chapter 5, we present a deep learning methodology for raga classification using Convolutional Neural Networks (CNN). Chapter 6 presents the details of the dataset and image data generation. In Chapter 7, I present the data preprocessing steps for the CNN algorithm to be used for raga detection. In Chapter 8 I present the results and analysis of this project. Finally, I conclude in Chapter 9 with a summary of findings and future work.

Indian Classical Ragas

Raga can be identified by various parameters. The particular choice of notes, Ascending and Descending sequences (known as arohana and avarohana pattern), nature of inflexion on different notes (gamaka/meend), characteristic phrases (pakad) all can be helpful to classify a raga. These are further described below:

1. Choice of Notes

   A rāga has a given set of notes (swaras), on a scale, ordered in melodies with musical motifs. The Indian tradition suggests a certain sequencing of how the musician moves from note to note for each rāga, in order for the performance to create a rasa (mood, atmosphere, essence, inner feeling) that is unique to each rāga. Theoretically, thousands of rāga are possible given 5 or more notes, but in practical use, the classical tradition has refined and typically relies on several hundred. For most artists, their basic perfected repertoire has some forty to fifty rāgas [8 -10]. Each raga has a different set of swaras that constitutes it. There must be the notes of the rag. They are the allowd swar. This concept is

similar to the Western solfege. There must also be a modal structure. This is called that in North Indian music and mela in Carnatic music. There is also the jati. Jati is the number of notes used in the rag. Rāga in Indian classic music is intimately related to tala or guidance about "division of time", with each unit called a matra (beat, and duration between beats) [11]. A rāga is not a tune, because the same rāga can yield an infinite number of tunes [12]. A rāga is not a scale, because many rāgas can be based on the same scale. Each raga tends to have a "Vadi" swara, a king swara on which maximum focus is given in a performance [1]. It is also known as the most frequently occurring swara in a particular raga. It is followed by Samvadi (next in importance), then Anuvadi. The swaras that are not allowed in a particular raga are known as Vivadi swaras (enemy notes).

2. Arohana/Avarohana

   There must also be the ascending and descending sequence of notes. This is called arohana /avarohana. Arohana and avarohana are the descriptions of how the raga moves. The arohana, also called aroh or arohi, is the pattern in which a raga ascends the scale. The avarohana, also called avaroh or avarohi, describes the way that the raga descends the scale. Both the arohana and avarohana may use certain characteristic twists and turns.

3. Pakad

The pakad or swarup, is a defining phrase or a characteristic pattern for a raga. This is often a particular way in which a raga moves; for instance the "Pa M'a Ga Ma Ga" is a tell-tale sign for Raga Bihag, or "Ni Re Ga M'a" is a telltale sign for Yaman. Often the pakad is a natural consequence of the notes of arohana / avarohana (ascending and descending structures). However, sometimes the pakad is unique and not implied by the notes of the arohana /avarohana. It is customary to enfold the pakad into the arohana / avarohana to make the ascending and descending structures more descriptive.

4. Gamakas

   Gamakas are better known as ornamentations used in Hindustani music system. These are inflexions and rapid oscillatory movements taken across swaras.

We now take one common raga as a running example and explain how the notes behave with respect to the above definitions and terms. Yaman emerged from the parent musical scale of Kalyan. Considered to be one of the most fundamental ragas in Hindustani tradition, it is thus often one of the first ragas taught to students. Yaman is a heptatonic (Sampurna) Indian classical raga of Kalyan Thaat. Yaman's Jati is a Sampurna raga.

Arohana: Sa Re Ga Ma(Kori Ma/tivra Ma i.e. Ma#) Pa Dha Ni Sa'

Avarohana: Sa' Ni Dha Pa Ma ((Kori Ma/tivra Ma i.e. Ma#)) Ga Re Sa

The ascending Aaroha scale and the descending style of the avroha includes all seven notes in the octave (When it is Shadav, the Aroha goes like N,RGmDNS' , where the fifth note is omitted; Pa but the Avaroha is the same complete octave). All the scale notes (called swaras) in the raga are Shuddha, the exception being Teevra Madhyam or prati madhyamam.

Vadi and Samavadi

Vadi is G (ga) and Samvadi is N (ni).

Pakad or Chalan

Kalyan has no specific phrases or particular features, many musicians avoid Sa and Pa in ascend or treat them very weakly. You will often hear N0 R G M+ D N S' in ascent and S' N D M+ G R S in descent). Sa is avoided in beginning the ascend such as N0 R G M+ P D N S'. To summarize the rules that make a raga:

A raga has at least 5 notes, including the Shadja. More than 5 is definitely ok. We can then add a further complexity, by choosing different swara in the ascent (Aaroha) and the descent (Avaroha). So the number of combinations can then be 5-5, 5-6,5-7,6-5,6-6,6-7,7-5,7-6 and 7-7. Five is Odhav, six is Shadhav and 7 is Sampoorna. So, an odhav-sampoorna raga will have 5 swar in the aaroha and 7 in the awaroha.

Librosa: Processing Musical Information in Python

The research field of music information retrieval (MIR) has been evolving rapidly. Taken broadly, it covers the area of musicology, digital signal processing, machine learning, information retrieval and library science. As digital music service platforms such as iTunes, Spotify and Pandora have grown, so has been the need for MIR tools which to date has been largely written by programmers as scripts using C++ or MATLAB. In recent years, interest has grown within the MIR community in using (scientific) Python as a viable alternative [ref]. LibROSA is a python package for music and audio analysis. It provides the building blocks necessary to create music information retrieval systems. Here I provide a brief introduction to basic ideas and elements in libROSA as I have used this package in conducting bulk of the work in my thesis.

In general, librosa's functions tend to expose all relevant parameters to the caller. While this provides a great deal of flexibility to expert users, it can be overwhelming to novice users who simply need a consistent interface to process audio files. To satisfy both needs, they define a set of general

conventions and standardized default parameter values shared across many functions.

An audio signal is represented as a one-dimensional numpy array, denoted as y throughout librosa. Typically the signal y is accompanied by the sampling rate (denoted sr) which denotes the frequency (in Hz) at which values of y are sampled. The duration of a signal (d) can then be computed by dividing the number of samples (y) by the sampling rate (sr):

$$d = \frac{y}{sr}$$

```
# duration_seconds.py
from librosa import load
import os
# Getting all mp3 files of raga Bhairav from dataset
fileList = os.listdir("data/Hindustani/mp3/Bhairav")
# Creating dictionary to store number of samples and sampling rate
sampleDict = {}
# y is number of samples, sr is sampling rate of the audio file
for file in fileList:
  y, sr = load(file)
  sampleDict[file].append(y)
  sampleDict[file].append(sr)
  # calculating signal duration
  duration = y / sr
  sampleDict[file].append(duration)
```

By default, when loading stereo audio files, the librosa.load() function down mixes to mono by averaging left- and right-channels, and then resamples the monophonic signal to the default rate $sr = 22050Hz$. Most audio analysis methods operate not at the native sampling rate of the signal, but over small frames of the signal which are spaced by a hop length (in samples). The default frame and hop lengths are set to 2048 and 512 samples, respectively. At the default sampling rate of 22050 Hz, this corresponds to overlapping frames of approximately 93ms spaced by 23ms. Frames are centered by default, so frame index t corresponds to the slice:

```
y[(t*hop_length - frame_length/2):(t*hop_length+frame_length/2)]
```

where boundary conditions are handled by reflection padding the input signal y. For analyses that do not use fixed-width frames (such as the constant-Q transform), the default hop length of 512 is retained to facilitate alignment of results.

The majority of feature analyses implemented by librosa produce two-dimensional outputs stored as numpy.ndarray, e.g., S[f, t] might contain the energy within a particular frequency band f at frame index t. We follow the convention that the final dimension provides the index over time, e.g., S[:, 0], S[:,1] access features at the first and second frames. Feature arrays are organized column-major (Fortran style) in memory, so that common access patterns benefit from cache locality. By default, all pitch-based analyses are assumed to be relative to a 12-bin equal-tempered chromatic scale with

a reference tuning of A440 = 440.0 Hz. Pitch and pitch-class analyses are arranged such that the 0th bin corresponds to C for pitch class or C1 (32.7 Hz) for absolute pitch measurements.

Core Functionality

The librosa.core submodule includes a range of commonly used functions. Broadly, core functionality falls into four categories: audio and time-series operations, spectrogram calculation, time and frequency conversion, and pitch operations [ref]. For convenience, all functions within the core submodule are aliased at the top level of the package hierarchy, e.g., "librosa.core.load" is aliased to "librosa.load".

Audio and time-series operations include functions such as: reading audio from disk via the audioread package [ref], resampling a signal at a desired rate, stereo to mono conversion, time-domain bounded auto-correlation, and zero-crossing detection. Spectrogram operations include the short-time Fourier transform (stft), inverse STFT (istft), and instantaneous frequency spectrogram (ifgram) [Abe 95], which provide much of the core functionality for down-stream feature analysis. Additionally, an efficient constant-Q transform (cqt) implementation based upon the recursive down-sampling method of Schoerkhuber and Klapuri [ref] is provided, which produces logarithmically-spaced frequency representations suitable for pitch-based signal analysis. Finally, logamplitude provides a flexible and robust implementation of log-amplitude scaling, which can be used to avoid numerical underflow and set

an adaptive noise floor when converting from linear amplitude. Since data may be represented in a variety of time or frequency units, a comprehensive set of convenience functions is provided to map between different time representations: seconds, frames, or samples; and frequency representations: hertz, constant-Q basis index, Fourier basis index, Mel basis index, MIDI note number, or note in scientific pitch notation. Finally, the core submodule provides functionality to estimate the dominant frequency of STFT bins via parabolic interpolation (piptrack) [Smith 11], and estimation of tuning deviation (incents) from the frequency reference A440. These functions allow pitch-based analyses (e.g., cqt) to dynamically adapt filter banks to match the global tuning offset of a particular audio signal.

Spectral Features

Spectral representations are the distributions of energy over a set of frequencies which form the basis of many analysis techniques in MIR and digital signal processing in general. Librosa implements a variety of spectral representations, most of which are based upon the short time Fourier transform. The Mel frequency scale is commonly used to represent audio signals, as it provides a rough model of human frequency perception [ref]. Both a Mel-scale spectrogram and the commonly used Mel-frequency Cepstral Coefficients (MFCC) are provided. By default, Mel scales are defined to match the implementation provided by Slaney's auditory toolbox [ref], but they can be made to match the Hidden Markov Model Toolkit (HTK). While Mel

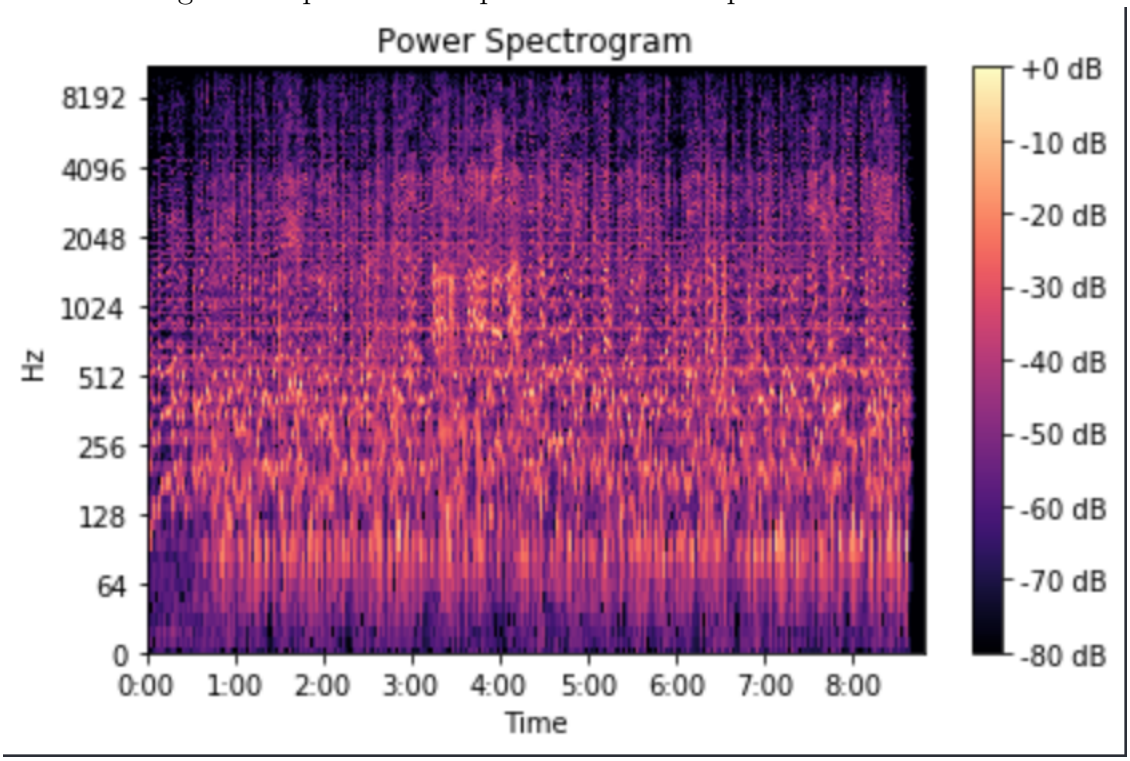
scaled representations are commonly used to capture timbral aspects of music, they provide poor resolution o pitches and pitch classes. Pitch class (or chroma) representations are often used to encode harmony while suppressing variations in octave height, loudness, or timbre. Two flexible chroma implementations are provided: one uses a fixed-window STFT analysis (chroma _ stft) [ref] and the other uses variable-window constant-Q transform analysis (chroma_cqt). An alternative representation of pitch and harmony can be obtained by the tonnetz function, which estimates tonal centroids as coordinates in a six-dimensional interval space using the method of Harte et al. [ref]. Figures 1,2,3, and 4 illustrate the difference between STFT, Mel spectrogram, chromagram, and Tonnetz representations.

Display

The display module provides simple interfaces to visually render audio data through matplotlib [ref]. The first function, display.waveplot simply renders the amplitude envelope of an audio signal y using matplotlib's fill_between function. For efficiency purposes, the signal is dynamically downsampled.

Mono signals are rendered symmetrically about the horizontal axis; stereo signals are rendered with the left-channel's amplitude above the axis and the right-channel's below. An example of waveplot is depicted in Figure 2 (top). The second function, display.specshow wraps matplotlib's imshow function with default settings (origin and aspect) adapted to the expected

Figure 1: Specshow Outputs for STFT Representation



defaults for visualizing spectrograms. Additionally, specshow dynamically selects appropriate colormaps (binary, sequential, or diverging) from the data type and range [ref]. Finally, specshow provides a variety of acoustically relevant axis labeling and scaling parameters. Examples of specshow output are displayed in Figures 1,2,3, and 4.

Figure 2: Specshow Outputs for Mel Spectrogram Representation

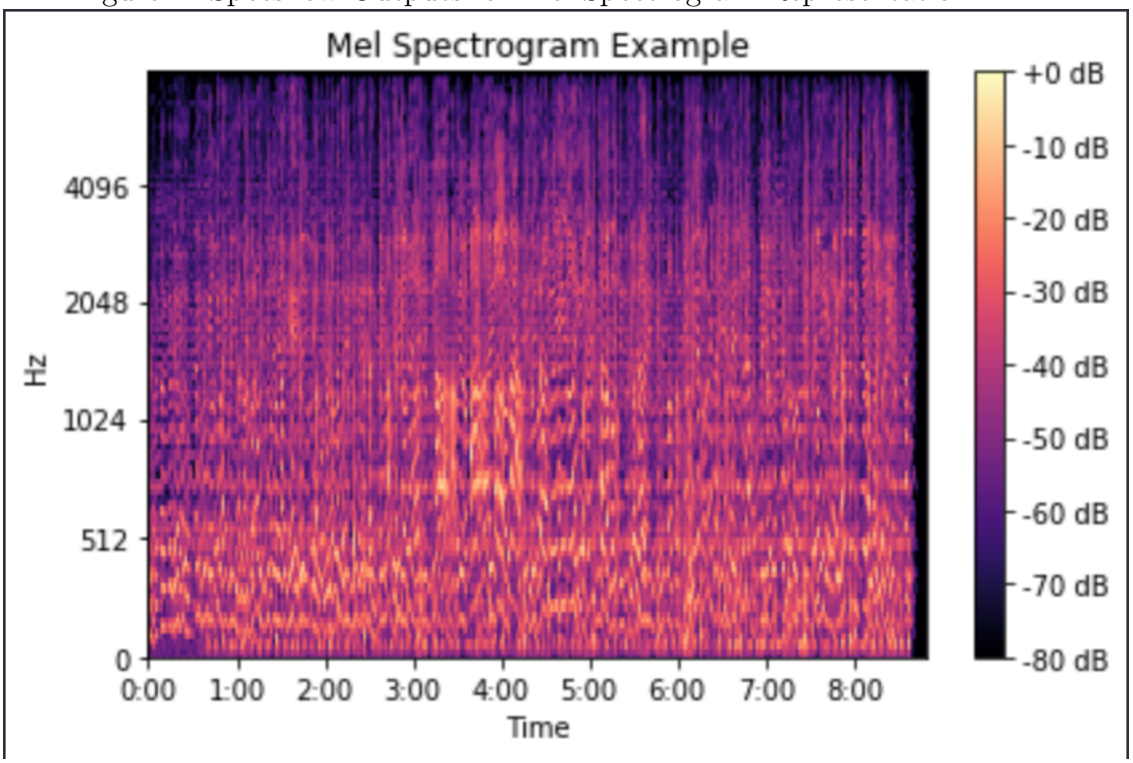


Figure 3: Specshow Output for Chroma_CQT Representation

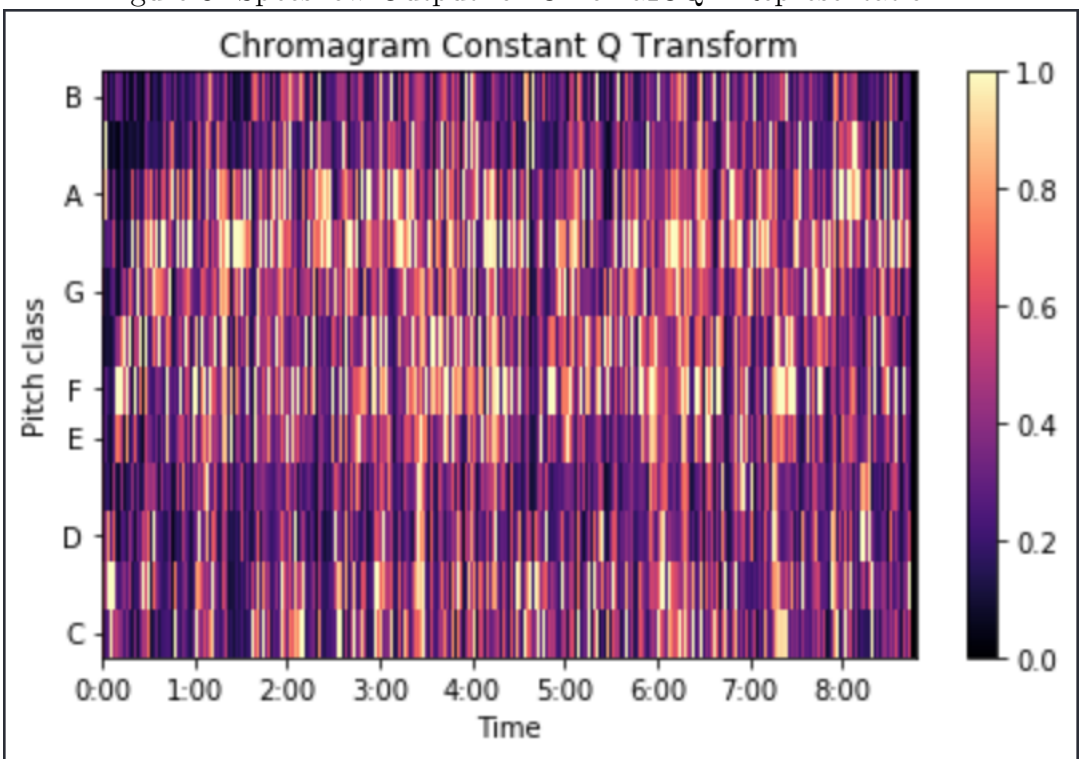
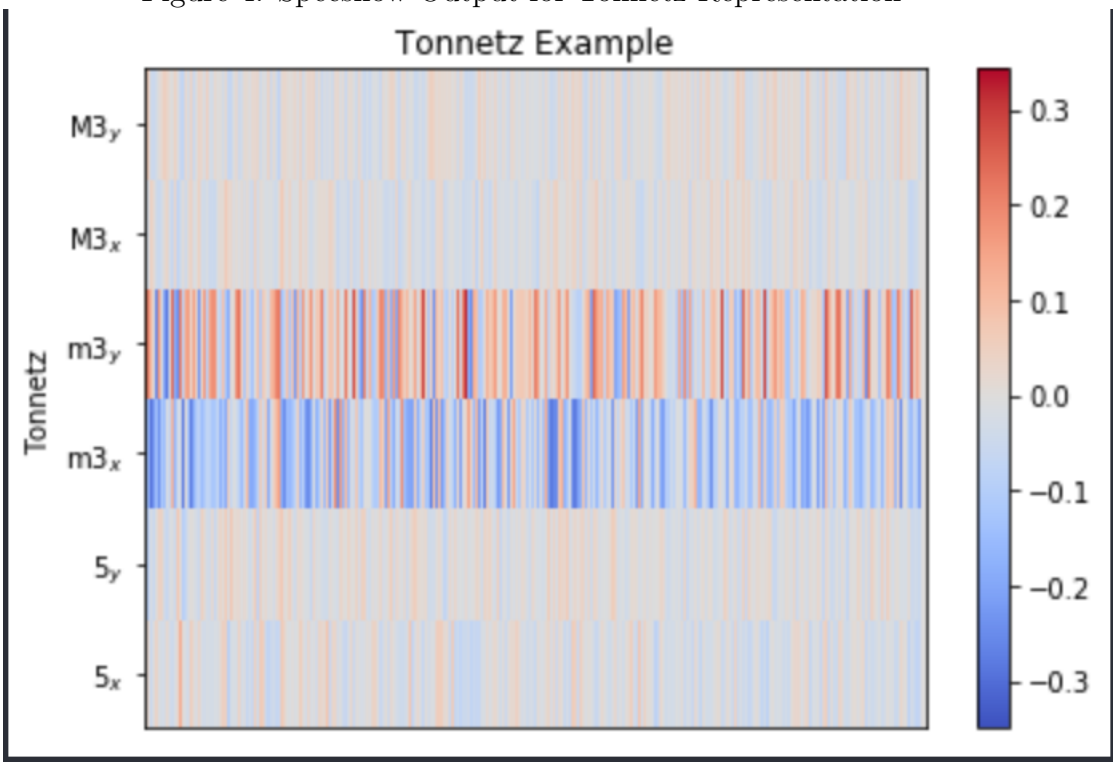


Figure 4: Specshow Output for Tonnetz Representation



Background and Related Work

Methodology

Initial Dataset and Image Data

Data Preparation and Processing

Results and Analysis

Future Steps

Bibliography

Appendix