Int. J. Nonlinear Anal. Appl.
Volume 12, Special Issue, Winter and Spring 2021, 2105-2123
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2021.6040

Harmonium note and triad music transcription using neural networks

Surekha B. Puri[a,*], Shrinivas P. Mahajan[b]

[a]*Department of Electronics and Telecommunication College of Engineering Pune (COEP), Wellesley Rd, Shivajinagar, Pune, Maharashtra 411005, India*
[b]*College of Engineering Pune (COEP), Wellesley Rd, Shivajinagar, Pune, Maharashtra 411005, India*

(Communicated by Madjid Eshaghi Gordji)

Abstract

Learning music requires a two-prong approach which includes theoretical studies and practical exposure to the instrument to be learnt. While previous literature has focused on developing technologies for determining the notes of different musical instruments, the harmonium has not been so popular in this research area. This research focuses on using a hybrid approach for polyphonic triad recognition of the Harmonium music. In this research, over 21000 audio samples of harmonium including notes and triads were taken for the Convolutional-Recurrent Neural Network (CRNN) model training purpose. The recorded audio samples were also used to train the Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) models to comparatively analyze the efficiency of these models. The results indicated that the CRNN model is more efficient, accurate, and precise on a score-based transcription. The proposed system produced 94% accurate results for triad recognition of Harmonium. The recognized triads were represented as sheet music using Lilypond. Possible applications of this output are for better understanding of the triad sequences by students or for Automatic Music Transcription of performances.

Keywords: Acoustic modeling, Music language modeling (MLM), Music analysis, Recurrent neural networks, Convolutional neural network.

1. Introduction

A One can say that Music is a Universal language, used for expression by artists since time immemorial. Instrumental Music is the form of music where the playing of an instrument, like a

*Corresponding author
Email addresses: gosavi.12@gmail.com (Surekha B. Puri), spm.extc@coep.ac.in (Shrinivas P. Mahajan)

Received: October 2021 *Accepted:* December 2021

Violin, Santoor, Piano or in our case, a Harmonium. Just like in Natural Language, music can be represented by using transcription on a music sheet, or in a digital medium. The transcription format of representing music is necessary due to multiple reasons. The process of automatic music transcription includes two different activities, including the study of a piece of music and the printing of a score obtained because of this analysis. [9, 18, 19, 11, 10, 22, 21, 20, 17, 1, 4, 5, 25, 26]. Polyphonic AMT is the method involved with changing over an acoustic musical signal into some type of musical documentation, for example, harmonium notes, printed music, Musical Instrument Digital Interface (MIDI) record, piano rolls, and so forth [26]. Every Author has his/her own way of interpreting and implementing and defining AMT. [29].The variability of input signal depending on the kind of musical instrument used. AMT systems which have unconstrained polyphony more complicate the modelling problem. Typically, the model objects to learn the properties of timbre of the musical instrument such that it can capture the variability in the input signal [43] and issues associated to large output space. [21].

Automatic music transcription for harmonium is uncharted territory. Unlike western musical instruments, music sheets are not available for the harmonium. Hence it is difficult to determine the notes from audio music signals of the harmonium [30].

Table 1: The Frequency of notes of octave 1

Note	**Frequency**	**Keys**
C	130.81 Hz	1
C#	138.59 Hz	2
D	146.83 Hz	3
D#	155.56 Hz	4
E	164.81 Hz	5
F	174.61 Hz	6
F#	185 Hz	7
G	196 Hz	8
G#	207.65 Hz	9
A	220 Hz	10
A#	233.08 Hz	11
B	246.94 Hz	12

Since the Indian Classical Harmonium is not a conventional western musical instrument, this research aims to bridge the gap in automatic music transcription of Indian Classical Music (ICM) using the Indian classical harmonium. Indian classical music is instructed primarily through speech by teachers and there are no written musical sheets. Majority of the music transcription mechanisms were deployed for transcription of instruments adopted by western countries like piano/keyboard. There has been negligible research in the harmonium domain and its note or chord transcription. It is also observed that the performance of existing AMT systems in a real-time scenario is very poor [21]. Harmonium notes of single octave structure with their respective numbers are mentioned in Table 1.

There are differences in the inter-tonal gaps as well. In order to understand the functioning of harmonium, its essentials to understand the notes and its corresponding relation with other notes. Basic note representation in music is A,B,C,D,E,F and G and each note has a different frequency which leads to differentiate each note from another note. There is a need for an integrated or hybrid approach which will combine both the models, acoustic as well as Music Language Model (MLM),

which works on polyphonic audio signals [3, 35, 6, 14, 12, 13, 15, 16, 2, 24, 23, 9, 18, 19, 11, 10, 22, 21, 20, 17, 1, 4, 5, 25, 26], using advanced neural networks for higher transcription accuracy.

The proposed system makes usage of the Convolutional Recurrent Neural Network (CRNN) approach for determining polyphonic as well as monophonic notes/triads played via harmonium. The proposed system firstly accepts the audio sample for the harmonium from a professional artist and train the model of CRNN for analyzing the triad (a combination of 3 notes played simultaneously) or notes played in the audio sample. The pre-processing of the audio samples is carried out before passing to the training. Pre Processing of the samples first emphasizes on removing the noise if any, secondly on computing the Fast Fourier Transform i.e. FFT [9] for fetching the pitch frequency range to determine the octave under which the note/triad is played. After that, the sample is divided into smaller audio chunks based on the number of played notes/triads using decibels as the criteria. Once the chunks are obtained, the CRNN model is trained over the chunks and the trained model is then provided with test inputs to determine the accuracy of the predicted notes/triads. The major challenge to obtain the accurate combination of notes played was resolved by using Chromagram data, which graphically represents the bands of the notes detected in the played chord/triad. [17].

Harmonium being the rarest research instrument in MIR, the proposed system required a thorough research right from generating the dataset (harmonium recordings) for training the model, with variation in dataset type such as single note dataset, sequential notes, mixed notes and triads. Dataset generation required professional harmonium players as the training for the neural network model required precise data. Dataset consisted of recording samples of single notes played for 3 seconds, sequential notes audio samples of 12 seconds and mixed notes and triads audio samples of around 15 seconds. The most critical contribution in the proposed system consisted of making the neural network robust enough to identify the difference between the single note and a chord/triad played. Also, the samples recorded for training consisted of combined data, so every recording sample required labelling the audio sample with the corresponding note or triad given input. Secondly, once the dataset generation completed, the dataset cleaning process included silence removal, noise removal for training the neural network model. Music Information retrieval being a very wide area of research, the proposed system significantly contributed in determining single notes, sequential notes and mixed notes and triads i.e. acoustic as well as polyphonic chord recognition individually as well as mixed. The proposed work significantly contributed in using the optimal mechanism to train the model by pipelining the audio to chromagram and chromagram to pitch frequency recognition, which led to achieve highest accuracy with CRNN neural network model. Analysis of CNN, RNN and CRNN provided deeper analysis to choose the best model to deploy for the notes or triads recognition.

2. Method

Monophonic music transcription is the topic of attention for a few years. Now, it's the need of the era that polyphonic chord recognition is practically realized. Only speech recognition or only speech analysis does not completely or precisely provide transcription of chords [13]. Techniques that have been researched earlier include endwise neural network mechanisms for music transcription [24]. But the simple neural networks lack the desired or the requisite efficiency and take longer to transcript the input audio file. The proposed model is used to identify triads, hence the model is trained using recordings of specific triads. Figure 2 depicts the model for complete structure of acoustic model and music language model (MLM) system flow using neural network. As long as the proposed system is getting data, it will be stored in the buffer. The term "data" in the proposed system refers to different audio samples which are recorded from the harmonium. This means that if the proposed system is receiving a particular frequency audio sample, it would be first stored in a buffer. Then that

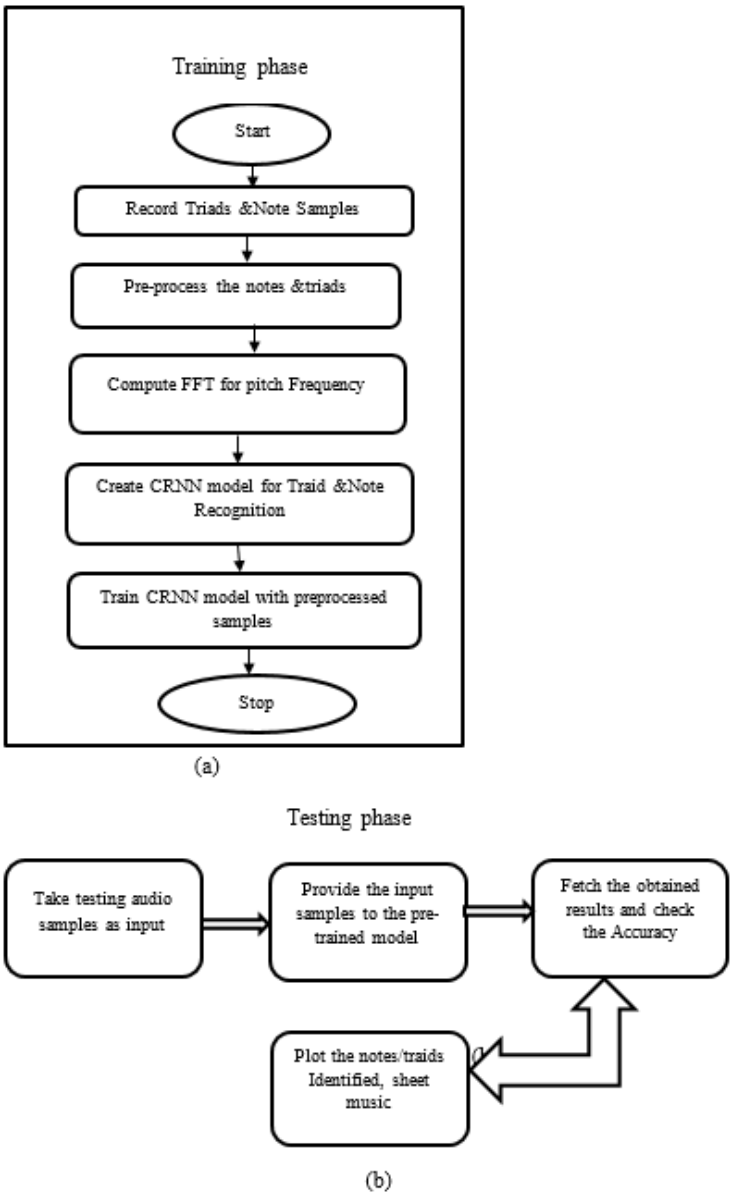

Figure 1: (a) & (b) Proposed complete structure of acoustic and language modeling system flow using Neural Network.

particular buffer window would be given to Fast Fourier Transform (FFT). When a new frequency is to be captured in another buffer window, the previous buffer data would be shifted down and it will then be stored in the buffer.

The proposed system calculates a frequency output for each recognized triad i.e. 130 Hz to 988 Hz. The proposed system chooses the determined frequency output for each window. Based on this output, the proposed system initially determines the octave and then based upon the frequency, it determines the nearest note. Thus it predicts the note along with the octave. The proposed system does this until it gets the detected frequency output for every window in the buffer. After this, the proposed system checks if the number of frames of the buffer is greater than or equal to the FRAMES_PER_FFT and then accordingly it returns the expected output, i.e., the note or chord along with its frequency in 130 Hz to 988 Hz.

Convolutional Recurrent Neural Network (CRNN) is a deep learning and machine learning technique for machines to comprehend the features of a harmonium chord with foresight and remember

the features suggested if the name of the new chord is fed to the classifier [13-20]. The model is trained using recording samples for some specific chords. After some initial testing, it was discovered that using an initial base learning rate of a few samples which worked well in fitting the training data - it provided a stable increment in accuracy and seemed to successfully converge. Once the improvements in the loss stagnated, the process was terminated manually and decremented the learning rate so as to try and increase the optimization of the loss function. Training took approximately an hour for 10 epochs for each model.

The Table II shows some of the combinations that were taken during the collection of the dataset from the harmonium. This table contains the combinations of 3 notes which will together make a triad. There are 12 notes in a single octave. So, by combinations, we get 220 triads in one octave. There are three octaves, so we get a total of 660 triads on the entire harmonium assuming only intern octave triads and not intra octave combinations. All the triads were manually captured from the harmonium. There are fixed frequency ranges for every octave in the harmonium. Table II shows the frequency range out the octave from which the triad or the note is being played. Thus, frequency is very important in any given audio.

Table 2: Frequency range of the octaves

Octaves	Range (Hz)
Octave 1	130 to 247
Octave 2	261 to 494
Octave 3	523to 988

After this, system tried to determine the triads using the audio time series of sample. This was done using machine learning algorithm like Decision Tree Algorithm, Random Forest Algorithm and Support Vector Machine Algorithm to classify the notes or triads. This audio time series was realized using the librosa [27] library. Each of the audio file was loaded with a sampling rate of 16000 Hz. This was because by default the recordings were of about 44100 Hz. Down sampling or decimation helps reducing data size, compression, Thus, due to this, audio samples were down-sampled to 16000 Hz for reducing the size of each sample and thereby use more samples for better analysis. This audio time series was further stored in a data frame. Using this we tried to determine the triad that was played but the accuracy of such kind of model was very low (about 20 %) which was not acceptable. Thus, we forfeited this approach as this was not relevant.

As the audio time series was not reliable extracting the audio features using Mel-frequency Cepstral Coefficients (MFCCs) was tried. [9] Again, this was further stored in a data frame on which different machine learning algorithms were applied. But even using MFCCs, the accuracy of the model did not improve much, and it increased to about 25 %which was again not desired [**?**].

A similar approach was further done for Short Time Fourier Transforms (STFT) and using the Constant-Q transform (CQT). The highest accuracy of the lot was achieved using CQT [34] which was about 32 % which was still very less. We tried different machine learning classifiers such as Random forest classifier, Support Vector Machines, K-Nearest neighbor, but none of this provided good results. Even boosting techniques such as Adaboost were applied but it did not change the accuracy by much. [23].

So finally, in the end, we opted to compute the features using the Chroma vectors that were obtained through Chromagram plots. So, the Chromagram of each of the audio was plotted [33]. In this way, 500 audio samples for each chord or triad played was taken as input and each audio chromagram was plotted. This determines the pitch class of the audio from12different pitch classes. This was further used for image processing using Convolutional neural networks [30].

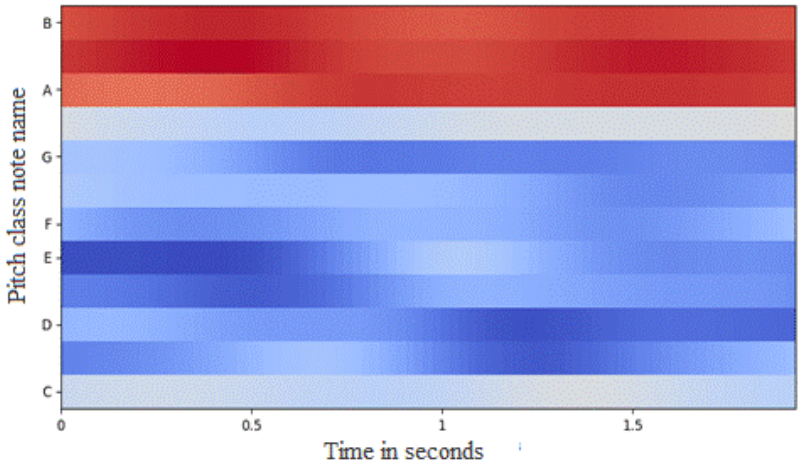

Figure 2: Chromagram for chord ABA#

Figure 2 represents how a chromagram looks like. The X-axis denotes the time in seconds. Similarly, the Y-axis denotes the pitch class. The red part denotes at what pitch the chord was played. This pitch would be different for each of the chords. So, this can be used to determine what chord was played and the time of the audio can be obtained. Chromagrams of different triads will be provided for the training of the neural networks and the triad prediction will be obtained. A similar approach can be done using the determination of a single note. Figure 3 depicts how a particular single note has been played.

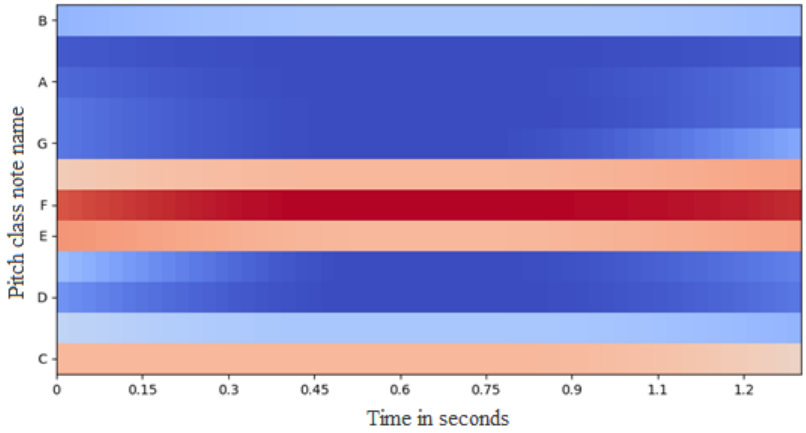

Figure 3: Chromagram for Note F

Figure 4 shows the chromagram for a single note F. When you compare the figures 3 and 4 it can be seen when a single note is pressed you get a dark red area just for a single key whereas when a triad has been pressed you get the noticeable red area for all the three notes. These features can be determined by the neural networks and the triads or the notes in the audio segment can be determined. The system essentially requires making use of classifier to determine the difference between the note and the triad played from the input audio file. The input audio file may have mixed samples played, i.e. some can be simple notes while some can be triads. The waveform of a file which contains 5 chords is shown in Figure 4.

As shown in figure 4, there are 5 different chords/notes played, plotted as a waveform, which can be split on the basis of the onsets detected in the audio samples, where onsets indicate the beginning

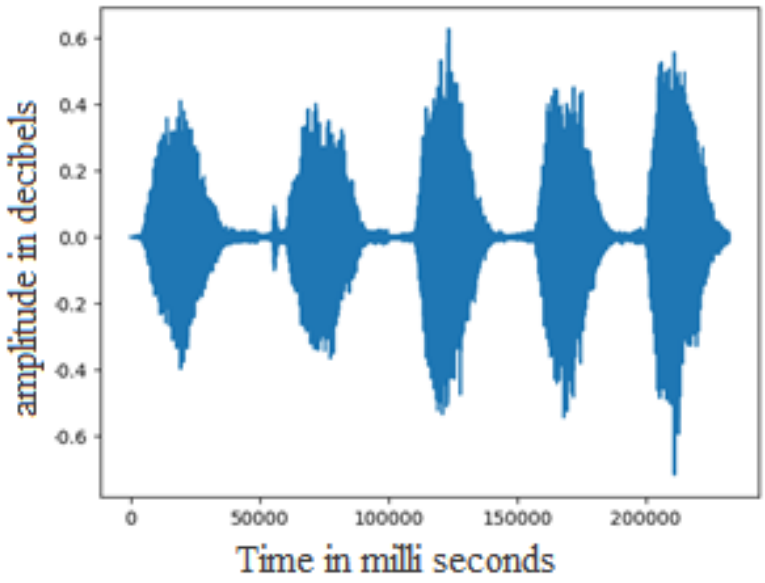

Figure 4: The Waveform for an audio file

of any note or chord played, the using the librosa library with onset_detect function available in librosa package. After this, the waveform of each separated chunk of the chords/notes are plotted. This single waveform is shown in Figure 6 where the waveform of first triad or note is plotted which can be further analyzed. Similarly, other audio samples can be split in the audio chunks. As there is a small amount of silence between one audio chunk and another, we can use this period to obtain different audio segments which are to be worked on.

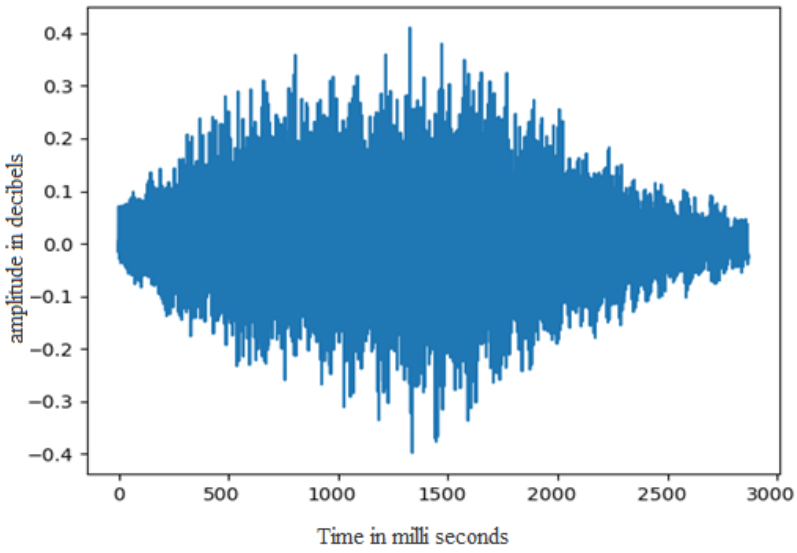

Figure 5: The Waveform for a chunk of an audio file.

In this audio segment generated with librosa, the audio timestamp of the segment is acquired. Using these timestamps, the time duration of each of the audio chunks was determined. Along with timestamps, the start time as well as the end time of the audio segment were acquired.

Thus, it was possible to find out the number of audio chunks or segments present in any given audio file. Along with this, we can also determine the silence duration in the audio file.

The next neural network implemented was the CRNN. The CRNN was the most accurate technique with 95 % accuracy in determining the triad or note being played in the audio file. CRNN operates as a combination of a CNN and RNN.

In this, the audio file again divided into chunks based on their silence periods and subsequently the plotting of the Chromagram was used to train the CRNN model. Table IV shows the comparison between various neural networks used and some of the metrics parameters used for checking the accuracy. In order to obtain triad detection from the input audio samples, a large dataset of various audio samples of harmonium triad was used for training the CNN. The CNN and RNN are special cases of DNN (Deep Neural Network) which can be trained rapidly by considering multiple input features.

Mathematically, proposed system can be represented as follows, for the Training Phase:

1. The various audio S input samples are first processed for E extracting various features $\sum$ from the sample and storing them in a reserved format R of a .csv file which then is further used for training the network. The audio samples are processed for silence removal and pitch-class profiling for pitch identification represented as follows:

$$R = f(\sum) + f(P(S)) \tag{1}$$

$$P(S) = \sum_{t_1=0}^{t_2} x^{t_1} - a^{t_1 - t_2} \tag{2}$$

Where, x indicates current raw audio sample, a indicates the silence detected at t_1, t_1 indicates the silence start time, t_2 indicates silence end time $P(S)$ indicates the silence removal function $f(P(S))$ indicates the function that extracts the features, $'+'$ indicates the concatenation of samples from which the silence is removed.

The major concern with performance prediction problem is about the way the features are represented, while on the other hand, conventional techniques of feature short listing emphasize on human selected features. Two major challenges about hand crafted or human provided features are that either human provided features will consume a lot of time, or human expertise will be required to achieve accuracy in feature selection, which will indirectly hamper the goal to classify the sample correctly. Features can be significantly extracted with the use of deep neural networks advancements done in recent times. In this system, CRNNs to model, which we shall now introduce. In proposed CRNN model, features are automatically captured, while the significant features performing major role in prediction from the extracted features are identified by the max-pooling layer is. Initial Phase of the model building is to train the model or feed the model with the input samples and it corresponding labels. Feature maps in convolution layers are the key performance indicators, which are derived out of various kernels being applied on single sample input. From these feature maps, the most prominent value is captured with the help of max-pooling. The extracted features are then correlated with the trained samples to fit the prediction, so the feature values are passed to the prediction module of the CRNN for predicting the triad in the input audio sample. Each of the captured recordings were converted into their respective chromagram, then these were provided as the training input for the Convolutional part of the CRNN. As it can be seen in Figure 9, the model contains 3 fully connected layers, as the CNN contains fully connected layers then it is further passed into an Artificial neural network. The relevant features are then extracted using the max-pooling of each of the layers. The batch size that was considered is 32. Then all of these were flattened further and then it was passed to another neural network. The Keras library was used in order to

implement this kind of neural network. Tensor-Flow is used at the backend for the training purpose which makes the work easier.

4. The obtained triad prediction is passed to the Lilypond library [8] of Python to get the sheet music representation for music composers to get ready to use the representation for training musicians and composing music. Algorithm 1 and Algorithm 2 shows the subordinate processes of Silence removal ad audio analysis.

Training & Prediction Process
S - Training Input audio samples
$S^{'}$ - Testing Input audio samples
F- Extracted Features
M - Sheet Music Representation
train ()-Training Function
lilypond () - Sheet Music Generation
t_1, t_2- Start and End Time Stamps for determined chord.
d- Duration of Chord
R- Results Obtained
F=train(S)
t_1, t_2, d = silence_removal (I)
R = predict($S^{'}$)
M = lilypond(R)

Algorithm 1 Non Signal Data Removal Algorithm from Input Sample
Identify the silence 'm' from the input sample file 'S' based on the amplitude of signal 'A'.
Input: The input sample audio file 'S'.
$S \leftarrow$ new audio input file. librosa.load(I) $\leftarrow$ time, amp values
$S^{'}$, index$\leftarrow$librosa.effects.trim
(amp values, threshold decibel, frame_ length)
librosa.output.write_wav($S^{'}$)

Algorithm 2 Audio analysis of input sample
Find the sampling rate 'frate' based on signal 'S' from the Noise/Silence eradicated input file $S^{'}$.
Input: The non-signal data eradicated audio file $S^{'}$.
frate, $S \leftarrow$ scipy.io. wavfile.read ($S^{'}$)
time_seconds$\leftarrow$len (s.shape)
t$\leftarrow$1.0/frate
FFT$\leftarrow$scipy.fft (time_seconds, t)
freq$\leftarrow$scipy.fftpack. fftfreq(time_seconds, t)

The existing systems make use of extraction of features from the audio samples manually [20]. So, the proposed system works on the Octave wise Detection of Harmonium Notes / Triads. Here, the proposed system is trying to detect the notes or triads played on a harmonium & represent the Sheet Music for the Notes/Triads along with its frequency. The proposed system makes use of PyAudio and sound device modules to detect the sound from a given input and hence process it further. The proposed system is then using the NumPy module to store this input in an array. The Proposed System is stores the audio in a Numpy array using the sound device, as the sound device offers bindings to use the Port Audio library along with some convenient functions to play and record NumPy arrays containing audio signals and frequencies and which has multi-platform support.

PyAudio is similar to tksnack which is nothing but a multi-platform toolkit for a sound that also has support for TCL/TK (Tool Command Language / Tool Kit) as well as Python. The proposed system is working on the detection of 3 major Octaves that are C3, C4, and C5. So, the proposed

system has divided it into 3 ranges of minimum note & maximum note to detect the notes of 3 octaves. The ranges of $note_{min}$ and $note_{maxx}$ proposed system uses are as follows:

1. from key 1 to key 12 for the C3 Octave.
2. from key 13 to key24 for the C4 Octave.
3. from key 25 to key36 for the C5 Octave

The Proposed System is using a sampling rate of 16000 Hz. and frame size of 2048 which means system is considering 2048 samples in one frame [7].

The formula which proposed system is using here is as follows:

$$s = f_s + f_p \tag{3}$$

Where's' represents samples per FFT, f_s represents frame size, f_p represents frames per FFT.

$$s_f = \frac{f_s^{'}}{f_s} \tag{4}$$

Where 'S f' represents frequency step, fs represents frame size, f's frequency samples.

Then the proposed system is using the split () method to hence display the output note-wise within a particular triad.

The standard sequence of the notes is as follows:

$$C, C\#, D, D\#, E, F, F\#, G, G\#, A, A\#, B$$

Also, the proposed system is generating 4 user-defined functions which are namely:

1. Freq_to_number (f): This stores the numeric frequency value in a NumPy array and proposed system pass frequency (fr) to this method as an argument as shown in equation 5.

$$f^{'} = g(fr) \tag{5}$$

   Where $g(f)$, is the function to convert frequency to number, f is the frequency and $f^{'}$ is the number.

2. Number_to_freq (n): Here, the proposed system is converting the number into frequency so that it can again be detected as a note / triad and the proposed system are passing the number object as an argument to this method as shown in equation 6.

$$fr = g(f^{'}) \tag{6}$$

   Where $g(f)$, is the function to convert number to frequency, fr is the frequency and $f^{'}$ is the number.

3. Note name (n): This returns the name of the triad based on the number which proposed system had derived previously from the frequency which was taken as the input. Hence, the proposed system pass this number as an argument to this method as shown in equation 7.

$$l = f(n) \tag{7}$$

   Where $f(n)$, is the function to convert note to corresponding name, n is the note and l is the note name or label.

4. Note_to_FFT_bin: It basically classifies and converts the input frequency into a number and for this part, the method divides the total number of frequencies by the frequency steps to hence achieve the same. The proposed system will create an empty matrix for every windowed sample so that the output of that can be stored with the same size within the matrix for every sample input. After this, proposed system will initialize the audio to take the input from a .wav file by using the PyAudio [30]. The parameters which proposed system will be passing to this method would be the format of the file, channels for capturing data input, sampling rate, frames per buffer i.e. FRAME_SIZE and after passing these parameters, proposed system will hence initialize the audio for capturing the input samples.

3. Dataset details

The proposed system emphasizes on the instrument which does not have the standard dataset readily available for the training the model for notes and chords together, so the dataset of notes and chords was created having per note 120 samples of 3 seconds approximating to 3600 seconds of notes dataset forming 1.5 GB of the dataset. For the polyphonic triad dataset, per recording of every triad is of 3 seconds on an average and per triad 500 samples were used for dataset creation. Overall, a total of 220 triads were trained with a minimum of 500 samples per triad of 3 s which summed up to 25 GB of the dataset. The recordings were collected from 3 different harmoniums to avoid the biased data and over fitting. The data was split into training and testing dataset to get assured about the authentic prediction of notes or triads.

For optimization for the triad detection of the harmonium, we captured the live recordings from a harmonium to create the required dataset. The dataset collection was very important as the training would be done on the collected dataset. If the dataset was collected wrongly then the training would be done in a wrong way which will, in turn, make the prediction of the chords wrong. Also, if the number of samples collected is in a very small quantity then it might not able to extract the relevant features that are needed for the prediction.

Table 3: A Few triad combinations out of entire triad dataset.

Chords
CC#D
C#DD#
D#EF
GG#A
AA#B
AA#
EFF#
AEF
ABG

To avoid this problem each of the triad was captured 500 times to increase the quantity of the dataset. Along with this, each triad was captured from different octaves which could also further help in determining what triad was played in which octave. Few triad combinations as shown in Table 3. The table 3 shows the subest of triads used for training the model. Overall there are $220 * 3 = 660$ triad combinations used, 220 combinations per octave with ${}^{n}C_{r}$, where n=12 and r = 3 for 12 notes out of which 3 keys played together to form a triad, obtained per octave and harmonium having 3 octaves.

Recordings were captured using a microphone in a room that contained very little noise. The work was further simplified using PyAudio [30] library in python which automatically saved each of the recordings one by one in any given folder. So basically, a folder for each of the chord was made and the recordings were saved.

The harmonium was played manually each time for every recording. Each of the triad was recorded for a time span of about two to four seconds. In this way, the recordings were captured in an easier way using PyAudio [30].

In this research, an optimal selection of features is represented, other than selecting all features. The different mixture of the feature vectors is considered as an input to the CNN and the RNN [31]. The mixture of features that are extracted is used for maximum accuracy and tested against the testing slices. These feature vectors behave as input to the neural network. In this system, one of the well-known machine learning algorithms out there which is cast-off for harmonium musical chord classification i.e. Convolutional recurrent Neural Network is also known as CRNN.

4. Comparative result analysis

Convolutional Neural Network: The proposed system has been implemented on 3 different models in neural network to analyze the implementation accuracy of different deep learning models, namely Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Convolutional Recurrent Neural Network (CRNN). CNN model has been configured with following hyper parameters. Two 2D convolutional layer with input layer with input units as 32 and input shape of (64, 64, 1), followed maxpooling 2D layer, followed by dense layer with "Relu" activation function with 128 input units finally passing the data from the dense layer with "softmax" activation function. The CNN model is compiled with the loss function "categorical_crossentropy" with "adam" optimizer function for 20 epochs with 20% validation test data used. Following are the results obtained for loss and accuracy for the CNN model with average accuracy of 85%.

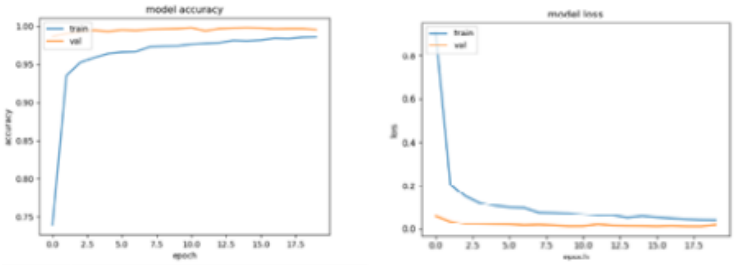

Figure 6: Model accuracy and model loss for CNN Model for predicting notes and triads.

Recurrent Neural Network: RNN model has been configured with following hyper parameters. Three LSTM layer with input layer with input units as 256 for first, 128 for second and 64 for third layer. The model has provided input shape of (64, 64), followed by dense layer with 128 input units, finally passing the data from the dense layer with "softmax" activation function. The RNN model is compiled with the loss function "categorical_crossentropy" with "adam" optimizer function for 20 epochs with 20% validation test data used. Following are the results obtained for loss and accuracy for the CNN model with average accuracy of 86.5%.

4.1. Convolutional Recurrent Neural Network

As the CNN and RNN model were built to identify the system performance for triad/note recognition, the Convolutional Recurrent Neural Network model built using the same dataset with con-

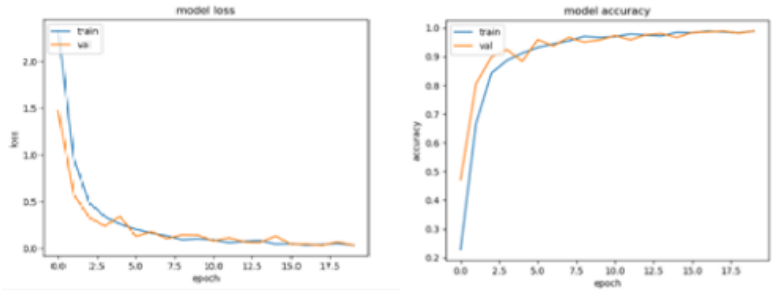

Figure 7: Model accuracy and model loss for CNN Model for predicting notes and triads.

figurations three 2D convolutional layers with input units of 32, 32 and 64 respectively with "relu" activation function having maxPooling2D layer associated with every convolutional layer, followed by 2 Dense layers with 26 and 30 input units respectively. The CRNN model proposed uses SGD optimizer with learning rate 0.01 and loss function being "categorical_crossentropy". The model is compiled using 25 epochs and 30 steps in every epoch with a batch size of 32. The proposed CRNN model for triad and notes classification gives the best performance depicting 93% accuracy.

Table 4 shows the comparison between the different neural networks and the metrics that were used in order to measure how accurate or precise the model predicts the prediction that is required.

Table 4: Comparative analysis of various neural network models

Neural Network	Convolutional Neural Networks	Recurrent Neural networks	Convolutional Recurrent Neural Network
Accuracy	85 %	84 %	94.89 %
Precision	0.90	0.90	0.95
Recall	0.91	0.89	0.94
F1 score	0.91	0.90	0.93

5. Experimental results with CRNN

These experimental results depict a proposed system was provided with various recordings as input where the recordings consisted of 5 to 6 different triads from every octave. Harmonium container 3 Octaves namely Mandhar, Madhya and Taar. These octaves are named based on the frequency range the octave comprises. So the first octave is Mandhar which indicates low frequency, second octave is Madhya which means medium frequency ranges and third octave is Taar, which indicates high frequency range compared to previous two octaves. Notes from these three octaves are played to form various combinations of triads. Every recording has a different triad sequence and different triad combinations. Before the recordings are given as input, every recording is divided into chunks based on the detected onsets and accordingly the features of every chunk are taken into consideration for training. As the Frequency and the Chords predicted from the CRNN model are obtained, it needs to plot them on the piano roll and the music sheet [25]. Plotting the proposed system using the horizontal broken bar available in Matplotlib as shown in figure 8.

We get the notes/triad played along with the time duration of each note/triad and plot the Broken Horizontal Bar with Time on X-Axis and Notes on the Y-Axis. We store all the Notes in the list and then Label Encode it using Label Encoder () and then convert them to digits using fit transform.

Through this we convert the notes [$'C','C\#','D','D\#','E','F','F\#','G','G\#','A','A\#','B'$] into [3 , 4, 5, 6 , 7 , 8 , 9 ,10 ,11 , 0, 1, 2]. Through this, we get a particular digit for each note specified. This process of Label Encoding is necessary because we cannot plot strings on the bar graph. So, if we get $C\#$ as a note played, it will plot the graph using digit 4 on Y-Axis. Suppose we get the First triad played as [$''D\#G\#F\#''$] and the time duration of it in the list as [0, 3] meaning the triad started playing at 0 and continued playing for 3 seconds. Another example is if we get the triad played as [$''A\#C\#F\#''$] and time list as [10, 2] this list received represents that the detected note or traid has been started playing at tenth second and have been identified to be played for 2 second, meaning the triad was played at 10 seconds and continued playing for another 2 sec. Then we split the triad into Notes. For Example [$''D\#G\#F\#''$] into [$''D\#'',''G\#'',''F\#''$].

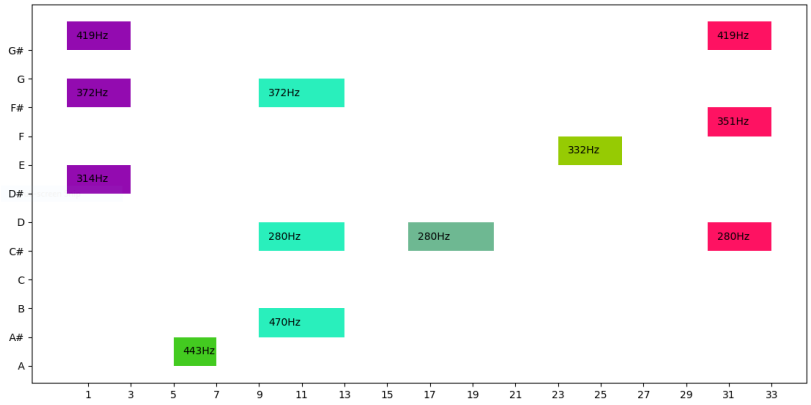

Figure 8: Frequency Plot for Detected Triads from input sample.

Now we specify the same color for all the notes played in particular time duration. To do that we used random mechanism to get a value in between 0-1 and then specify it during the plotting of the graph in RGB format. Figure 6 shows the frequency plot for detected triad from corresponding octave.

The next step is to plot the notes on the graph using Matplotlib with the X-axis plotting the note and y plotting the start and duration seconds to accomplish this, the system iterates through a list of notes. For example [$''D\#'',''G\#'',''F\#''$] we first plot $''D\#''$ on the graph with time for which it is played and assign a color to it. We also add the details of frequency to it to identify from which octave it is played. We plot all the notes played along with time duration and the result is a piano roll.

To the plot, then specify X-Axis using Sticks as the time separated by 2 secs. The digits on Y-Axis are plotted by renaming them to their Notes name. We then specify the frequencies we get from CRNN and then specify it on the note and show the piano roll. The results obtained for the predicted triad from the above classification results are passed on to the Lilypond library to generate the final music sheet representation of the predicted triad and musical notations so that obtained result can be used by musicians or music composers for generating music from the symbolic representation of musical notes and triad. The proposed system makes use of Sci-kit learn [32] modules along with the Tensor Flow module for implementation of the predictive model and CRNN for the training of the sample and triad prediction. The results obtained for the predicted triad from the above classification results are passed on to the Lilypond library to generate the final music sheet representation of the predicted triad and musical notations. Figure 9. Shows the music sheet generated by the system for the entire triad sequence.

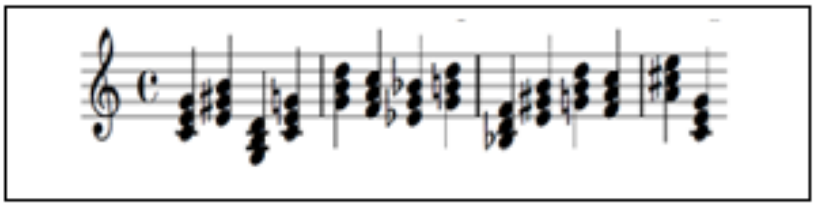

Figure 9: Harmonium Musical Triad Mode Notation Sheet of an audio signal.

6. Evaluation metrics

The proposed system evaluates the precision & recall for determining the effectiveness of the implementation of the proposed system. Precision tries to solve the following equation (8).

$$Precision = \frac{T_p}{T_p + F_p} \tag{8}$$

Where T_p = True Positive, T_n = True Negative, F_p = False Positive and F_n= False Negative.

Recall tries to solve the following question: What proportion of original positives was identified correctly? Mathematically, recall is defined as shown in equation (9)

$$Recall = \frac{T_p}{T_p + F_p} \tag{9}$$

Precision & recall values of independent implementation mechanisms are as displayed in Table 5. Figure 11 is a graphical representation of different models of neural networks that have been used in this technique.

F1 Score is defined as the score which details the performance of the system which is calculated with respect to the true negative, true positive, False negative and false positive. Mathematically, F1 Score is represented as follows in equation (10):

$$Fl_{Score} = 2 * \frac{(precision * recall)}{(precision + recall)} \tag{10}$$

Accuracy is measurement of performance and it is merely a ratio of an observation that has been predicted correctly to the total of the observations. The consideration is that the higher the accuracy, the better the model. It is indeed true that accuracy is an abundant measure . For our model, the proposed system achieved an accuracy rate of approximately 94%.

$$Accuracy = 2 * \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{11}$$

As per the trained CNN & RNN Models, the determined pitch is plotted graphically in figure 9. The figure 6 shows the detected pitch from the audio input and the corresponding energy with which it's used in the audio sample. The detected polyphonic triad in a single chunk is plotted in figure 6. The detected triad of the input audio sample comprises of multiple notes as the input audio sample is a polyphonic audio sample. Traditional ANN makes use of various activation functions like sigmoid, ReLu etc., like function used for threshold, Gaussian function, linear piece wise function, sigmoid function, and so on. In proposed system, linear piece wise function is deployed to work, [15] which is defined as $F : y = x; x_2(0; 1]$, as shown in Figure 10. The parameters of CRNN cannot be negative

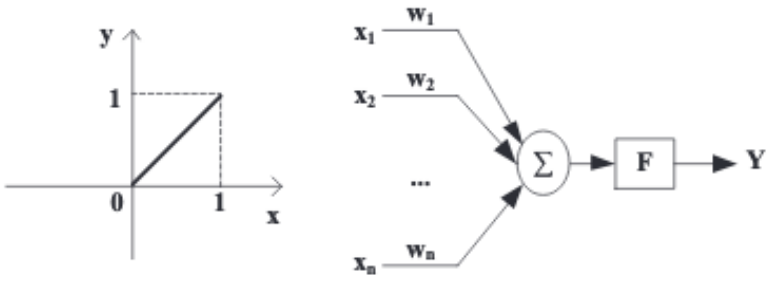

Figure 10: CRNN Activation function and node computing [17]

Table 5: performance analysis of proposed system

Accuracy	Precision	Recall	f1-score
0.94	0.95	0.94	0.93

because of the use of rectified linear activation function (ReLu) as this is the most efficient and advanced activation function in neural networks and the maximum value of x is 1.0 [15]. Table V gives an idea about the metrics that were measured after the model was ready using CRNN.

The training of the CRNN was done in 25 epochs. As the accuracy is pretty good then the other neural networks it can easily be used to determine the audio to a very high level.

For a single note or triad, there were 500 recordings taken from 3 different harmoniums, which lead to 500 chromagrams of each note or triad to be used for training the model. So, there were recordings done for triads as well as notes. This model was first tried on 15 epochs, but the accuracy which was close to 60 to 65% was unacceptable. Later the model was tested on 45 epochs, but the model became over fit as the accuracy was close to 100% after a certain epoch and then the accuracy deteriorated further. This meant that after 35 epochs the model was giving outputs at 100 % which would later turn out to be a major problem. It was observed that in 15 epochs the model was under fitted on the given data and on 45 it was over fitted. Hence to get the optimal accuracy the model was tested on 25 epochs which turned out to have an accuracy of 95 percent which was acceptable. This works as if it is the combination of the neural networks and this goes on until the audio segments that get divided are identified. The learning rate equally plays a vital role while training a model. The learning rate has been decided over multiple iterations being run over minimum learning rate of 1e5, to 0.1, and then 1 and finally the learning rate of 0.6 was identified as the most suitable one. While trial and error was done, not entire dataset was used, so as to reduce the time in deciding the learning rate. So the subset of entire training set was made with 20% dataset for deciding the learning rate.

In figure 11 the graph a & b shows the accuracy and how it is improved as the training goes on increasing an at epoch 25 it can be seen the accuracy is close to 1 which means the model is pretty well trained. Similarly, Figure 9 shows the model loss that gets less like the training of the model increases.

7. Conclusions

In this paper, Harmonium Triads detection has been proposed and determined using CNN and RNN. The proposed system makes use of the dataset generated by professional artists as input. A 44100 Hz audio signal at 32 bits per sample input which was down-sampled to 16000 Hz frequency was used for sampling the input signals. The database for the harmonium audio file is gathered

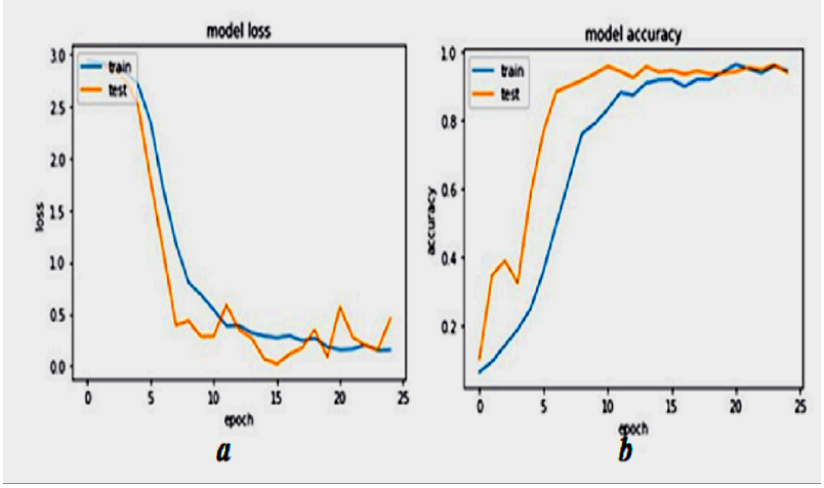

Figure 11: Graph of the model accuracy [17]

from a professional artist for the proposed system development. Features of the audio .wav files are extracted and represented as Chromagram [32]. Also, the processed audio file is used to determine advanced parameters like onset times, note detection and frequency analysis. The desired parameters of the audio file are then given as input to the neural network for training and the trained dataset is obtained for predicting and estimating the triad from the test dataset.

In contrast, to the prior techniques proposed for monophonic and polyphonic piano note detection, is being proposed for polyphonic harmonium note detection has gained more efficiency as there are a few octaves to be processed in a harmonium in comparison with a piano & therefore, the parameters to remain compared are a few in number to decide the notes or triad from 3 octaves. As, in comparison to piano, harmonium has lesser number of keys and so the combination of keys and training samples required are less, so the proposed system on harmonium triad recognition can work faster compared to triad recognition on piano. The proposed system makes use of the Lilypond library to convert the predicted triad from the input file to generate the music sheet which can help the music composers or musicians to use the end product of the proposed system directly for a reference. The proposed system makes use of Lilypond 2.18.2 version of Windows and the corresponding Python.ly module of Python to integrate Lilypond commands with Python for obtaining the '.ly' file which has the script for the generation of sheet music.

8. Future directions

The results of the tested recordings show that the proposed system will determine the triad sequences from the recordings which are played at a slower pace and the working of this will require vigorous input of raga recordings which will not only determine the triad sequence, but will also determine the played raga in the test sample.

References

[1] S. Adavanne, A. Politis and T. Virtanen, *Direction of arrival estimation for multiple sound sources using convolutional recurrent neural network*, Proc. Eur. Signal Process. Conf. 2018.

[2] M. Arjovsky, S. Chintala and L. Bottou, *Wasserstein generative adversarial networks*, Int. Conf. Machine Learn. 2017, pp. 214–223.

[3] E. Benetos, S. Dixon, Zh. Duan, *Sebastian ewert: automatic music transcription: an overview*, IEEE Signal Process. Mag. 36(1) (2019) 20–30.

[4] S. Chakrabarty and E.A.P. Habets, *Multi-speaker localization using convolutional neural network trained with noise*, Proc. Machine Learn. Audio Process. Workshop at NIPS, 2017.
[5] S.Y. Chang, B. Li, T.N. Sainath, G. Simko, and C. Parada, *Endpoint detection using grid long short-term memory networks for streaming speech recognition*, Proc. Interspeech 2017.
[6] J. Devlin, M. Chang, K. Lee and K. Toutanova, *BERT: pre-training of deep bidirectional transformers for language understanding*, arXivpreprintarXiv: 1810.04805, 2018.
[7] E.L. Ferguson, S.B. Williams and C.T. Jin, *Sound source localization in a multipath environment using convolutional neural networks*, Proc. IEEE Int. Conf.Acoustics, Speech, and Signal Process. 2018.
[8] T. Gan, *Música colonial: 18th century music score meets 21st century digitalization technology*, JCDL '05: Proc. 5th ACM/IEEE-CS Joint Conf. Digital Libraries, pp. 379.
[9] M. Huzaifah, *Comparison of Time-Frequency Representations for Environmental Sound Classification usingConvolutional Neural Networks*, arXiv:1706.07156v1 [cs.CV], 2017.
[10] N.P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden and A. Borcherset al., *In-datacenter performance analysis of a tensor processing unit*, in IEEE Computer Architecture(ISCA), 2017 ACM/IEEE 44th Annual Int. Symp. (2017) 1–12.
[11] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A.v.d. Oord, S. Dieleman and K. Kavukcuoglu, *Efficient neural audio synthesis*, arXiv preprintarXiv:1802.08435, 2018.
[12] T. Kawashima and K. IchigeK, *Automatic piano music transcription by hadamard product of low-rank NMF and CNN/CDAE outputs*, IEEJ Trans. Electron. Inf. Syst. 139(10) (2019) 1106–1112.
[13] M. Kolbæk, Z.H. Tan and J. Jensen, *Monaural speech enhancement using deep neural networks by maximizing a short-time objective intelligibility measure*, Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process., 2018.
[14] J. Lee, J. Park, K.L. Kim and J. Nam, *Sample-level deep convolutional neural networks for music auto-tagging using raw waveforms*, Proc. 14th Sound and Music Comput. Conf. Espoo, Finland, 2017, pp. 220–226.
[15] W. Li, L. Cao, D. Zhao, X. Cui and J. Yang, *CRNN: Integrating classification rules into neural network*, Proc. Int. Joint Conf. Neural Network. 2013, pp. 1–8.
[16] Q. Liu, Y. Xu, J.B. Jackson, W. Wang and Ph. Coleman, *Iterative deep neural networks for speaker-independent binaural blind speech separation*, Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. 2018.
[17] S. Mishra, B.L. Sturm and S. Dixon, *Local interpretable model-agnostic explanations for music content analysis*, ISMIR 2017.
[18] M. Müller, D.P.W. Ellis, A. Klapuri and G. Richard, *Signal processing for music analysis*, IEEE J. Selected Topics in Signal Process. 5(6) (2011) 1088–1110.
[19] Musical scales, [Online]. Avalable: https://heptagrama.com/musical-scales.htm.
[20] B. PuriS and S.P. Mahajan, *Optimum feature selection for harmonium note identification using ANN*, 10th Int. Conf. Comput. Commun. Network. Technol. 2019.
[21] S.B. Puri and S.P. Mahajan, *Review on automatic music transcription system*, Int. Conf. Comput. Commun. Control and Automation 2017.
[22] P. Raguraman, R. Mohan and M. Vijayan, *LibROSA based assessment tool for music information retrieval systems*, IEEE Conf, Multimedia Information Processing and Retrieval 2019.
[23] A. Román Antonio Pertusa, *Jorge Calvo-Zaragoza: Data representations for audio-to-score monophonic music transcription*, Expert Syst. Appl. 162 (2020).
[24] M. Schedl and S. Böck, *Polyphonic piano note transcription with recurrent neural networks*, IEEE Int. Conf. Acoustics, Speech, and Signal Process. 1988.
[25] A. Schlüter, *Learning to pinpoint singing voice from weakly labeled examples*, ISMIR 2016.
[26] S. Sigtia, S. Dixon and E. Benetos, *End-to-End neural network for polyphonic piano music transcription*, IEEE/ACM Trans. Audio, Speech, and Language Process. (2016) 927–939.
[27] Y.C. Subakan and P. Smaragdis, *Generative adversarial source separation*, IEEE Int. Conf. Acoustics, Speech and Signal Process. 2018, pp. 26–30.
[28] D. Wang and J. Chen, *Supervised speech separation based on deep learning: an overview*, arXiv:1708.07524, 2017.
[29] T. Weyde, S. Sigtia, S. Dixon, G. D'Avila, E. Benetos, N. Boulanger-Lewandowski and S. Artur, *A Hybrid Recurrent Neural Network For Music Transcription*, School of Electronic Engineering and Computer Science Centre for Digital Music 1411.1623, 2014.
[30] G.A. Wiggins, S. Liu, L. Guo and F. Cong, *A parallel fusion approach to piano music transcription based on convolutional neural network*, ICASSP 2018-2018 IEEE Int. Conf. Acoustics, Speech and Signal Process., 2018.
[31] J. Xu, B. Tang, H. Man and H. He, *Semi-supervised feature selection based on relevance and redundancy criteria*, IEEE Trans. Neural Networks Learn. Syst. 28(9) (2017).
[32] A. Ycart and E. Benetos, *Polyphonic Music Sequence Transduction with Meter Constrained LSTM Networks*,

Conf. ICASSP 2018-2018 IEEE Int. Conf. Acoustics, Speech and Signal Process. 2018.
[33] J. Zhang, X. Yu, W. Wan and J. Liu, *An audio retrieval method based on Chroma gram and distance metrics*, Int. Conf. Audio, Language Image Process. 2010.
[34] A.Zhou, *Chord detection uUsing deep learning*, Int. Conf. Music Inf. Retrieval 2015.
[35] G. Zweig, C. Yu, J. Droppo and A. Stolcke, *Advances in all-neural speech recognition*, Proc. ICASSP, 2017.