Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 171 (2020) 465–474

Procedia Computer Science

www.elsevier.com/locate/procedia

Third International Conference on Computing and Network Communications (CoCoNet'19)

An Improved RNN-LSTM based Novel Approach for Sheet Music Generation

Mohit Dua[a], Rohit Yadav[b], Divya Mamgai[c], Sonali Brodiya[d]

[a,b,c,d]*Department of Computer Engineering, National Institute of Technology, Kurukshetra 136119, India*

Abstract

It is very well known that Sheet Music is one of the most effective medium for musicians, professional artists and amateurs to access the chords to any songs. Many systems have already been developed which generate sheet music by taking song as input. This paper presents one such system that aims to improve the accuracy of sheet music generated by previous works. Improvement is achieved by working on source separation and chord estimation modules of the previous system. The proposed work utilizes Deep Learning techniques such as Recurrent Neural Network (RNN) with Gated Recurrent Units (GRU) and Long Short Term Memory (LSTM). In source separation module, multi-layered GRU cells for implementing RNN and in chord estimation module, LSTM cells for implementing RNN are used for the implementation. In source separation module, the number of sources that it can separate are also increased to improve the accuracy of chord estimation module.

Keywords: Deep Learning; RNN; LSTM; Sheet music; Source Separation; Chord Estimation.

1. Introduction

There is a huge number of artists already present in the music industry globally. Apart from them, music professionals, learners, students and amateurs also form part of this large music community [1]. Sheet music is one of the sources for learning and playing songs for these artists and amateurs [2]. Sheet music is an aid in the form of a paper consisting of chords and lyrics to a song, and is very helpful in playing instruments such as piano, guitar and violin etc. [3][4]. It enables players to identify the chords, rhythm, pitch to any song or instrumentals.

Before 15th century, music was written on manuscripts by hand. Although music printing had begun by this time, composers' handwritten manuscripts continued to be preferred over the printed sheet music [5][6]. Gradually, sheet music came to be widely accepted in the form of printed paper. In the nineteenth century, music industry was dominated by publishers publishing the sheet music [7]. It was in last years of 20th century that digital sheet music came into being and began its journey over web. Sheet music, since then, is shared over internet and becomes a considerable source of communication among the artists present in music community. From many years, sheet music has been used by musicians to practice music and pass it on to future generations as well [8].

As everything has fallacies, sheet music has some problems too, some of which are addressed while others are not. The first problem is unavailability of sheet music and it is one of the major problems faced by artists presently. This unavailability may be due to various reasons including copyright issues where access to any resource (sheet

Corresponding author. Tel.: +91-946-658-8448
E-mail address: er.mohitdua@gmail.com

10.1016/j.procs.2020.04.049

music here) is denied to others through formal rules and laws. Secondly, some local songs or songs that are not so famous may not have music sheet present at all. There may be a situation where a music artist, student or amateur, with not much experience, may want to play such a song on an instrument but doesn't know the chords. In such cases, these artists and amateurs are left helpless without any alternative to the sheet music since amateurs heavily rely on sheet music to fulfil their ambition of playing others' songs.

Further, there is no denying the fact that sheet music is a vital part of music theory which forms the basis of music education. It helps music students understand music better and without any malaise. Instruments are learnt easily with the help of sheet music. This makes sheet music large part of publishing educational music as well as it forms an integral part of music theory and is included in music schools as a part of course where students are taught to read sheet music and this sheet music is further referred to, in order to play any song. A student can apply his/her learning through practice with the help of sheet music. Practice is only possible through sheet music. The more the number of sheets a student refers to, the better he/she gets at playing an instrument or practice vocals along.

Besides students, sheet music plays an important role in every artist's life indeed. It is responsible for making artists more diverse and assists them in reaching their potential. Additionally, it is through sheet music that various musicians communicate with each other. A musician can write his composition in the form of a sheet music. These compositions are exchanged, discussed and agreed upon through the means of sheet music. While a musician can share his composition through sheet music, he can access others' composition through the same medium and enjoy playing other artists' composition with their consent. Other alternatives for sheet music may be present; nevertheless, sheet music still dominates the music industry as the choice for the medium of musical composition.

A solution to this problem is developing systems that can generate sheet music for these artists [9]. From quite some time, some work has already been going on in this field. Various systems have been made to produce sheet music based on different techniques. One such system has been discussed in [10] which uses deep learning techniques. It takes songs and lyrics as input and gives out sheet music as the final product. Likewise, plenty of systems have been deployed to accomplish the same task.

This paper discusses a system that aims to provide a more accurate sheet music over the one generated in the previous work [10]. The accuracy of previous system is around 76% and our system intends to increase this number. This is mainly achieved by working on source separation and chord estimation module. Both the processes: (1) separating vocal from non-vocal part and (2) chord estimation are built upon deep learning mechanisms. First, these modules are trained using datasets and then tested. Source separation or separating vocal and non-vocal part uses RNN with GRU cells while chord estimation uses RNN with LSTM cells [11][12].

2. Fundamentals

2.1. Recurrent Neural Network (RNN)

RNN or Recurrent Neural Network is deep learning model made up of neurons. It is mainly useful when considering sequential data as each and every neuron can utilise its internal memory to store information about the previous input. This works more like a loop where output of a neuron at one particular stage is fed to itself as input in the subsequent stage. This, in fact, appears like output of one neuron acts as input to other but actually, there exists only one neuron performing both the tasks [13]. RNN has been regarded potentially significant for applications related to music, particularly music composition [14].

In RNN, the output depends on the previous computations unlike the previous neural networks where previous computations had to be known in order to predict the next output. RNN can be thought as a network that captures all the computations or memorises everything that has happened so far. What this precisely means is that RNN considers two inputs; one is the current input and the previous computations act as the second input [15]. Like other neural networks, it contains an input layer, hidden layers and an output layer. A typical or conventional neural network is of the form:

$$X_t = Y_t \tag{1}$$

i.e. for a set of input, we have corresponding set of output. However, in case of RNN, this simple relation is modified to:

$$[X_t, X_{t-1}] = Y_t \quad (2)$$

i.e. the current input X_t and the input from the past, X_{t-1} is used to predict the output Y_t. Further, in an unfolded form, RNN looks like the below figure:

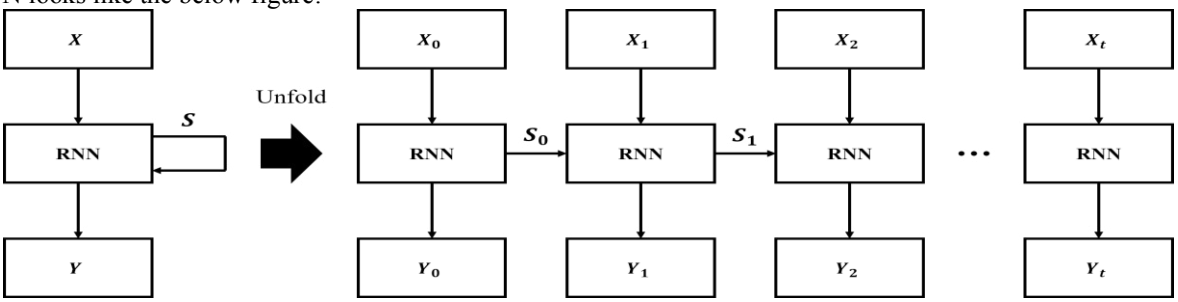

Fig 1: RNN in detailed unfolded form

The equation for prediction in RNN is:

$$h_t = F_n(h_{t-1}, x_t) \quad (3)$$

where, h_t is the new state, F_n is some function that performs computation with some parameters n, h_{t-1} is the previous state and x_t is the current input.

For training the network, a categorical cross entropy loss function is commonly used. From each output node, the gradients are computed as backward pass through the network. This technique is Back-propagation-through-time (BPTT) technique.

2.2. Long Short Term Memory (LSTM)

LSTM or Long Short Term Memory was introduced in the mid-90s by German Researchers as a variation of recurrent net with the long short-term memory units. It is a step ahead of RNN. While Recurrent network suffered with the problem of vanishing gradient and explosion gradient, this was proposed as a solution to the above mentioned two problems. This was achieved by explicitly introducing into the network, a cell as a unit. This cell performs task of decision making by considering previous memory, current input and previous output. The memory is altered by generating a new output. This property of LSTM has been utilised in the field of music as well be it composition or transcription modelling etc. A lot of work has been done in this field using LSTM. We achieved the objective of chord estimation using LSTM [14].

The building block of RNN is Long Short Term Memory (LSTM). LSTM follows selective procedure and filters information. LSTM selectively remembers pattern for long [16]. Looking at the architecture of LSTM, the input and the output layer are similar to the ones in RNN. The difference lies in the middle cell or the repeating module. Repeating module in LSTM contains 4 layers instead of one, like present in RNN. These layers present in LSTM interact with one another and this interaction forms part of the decision making process in LSTM. The figure given below presents the structure of the repeating module.

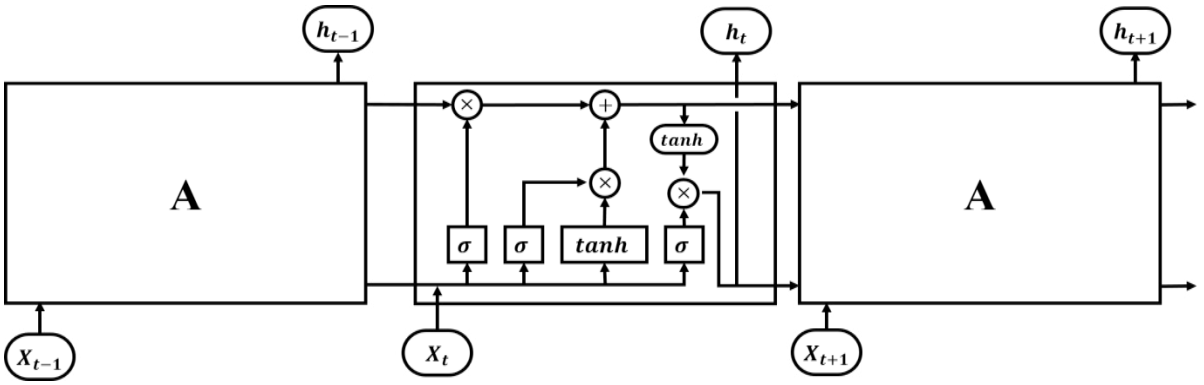

Fig 2: Repeating module of LSTM

The horizontal line present with the plus and the cross sign is the cell state and acts as the key to LSTM. LSTM is capable of modifying information present in the cell state. The cell state acts as a belt that decides how much of input to process from each of these 4 layers. The '+' and 'X' present on the cell state are the gates. Gates decide whether the information must be let through it. They are made up of a sigmoid neural net layer and a point wise multiplication operation. The sigmoid layer gives number between 0 and 1 as output, this number describing how much information to let through. The higher the number, the more the amount of the information allowed through the gates.

The first step is to decide what information should be thrown away, via the forget layer and the forget activation layer. Secondly, the new information that has to be stored in the cell states must be decided, which is done with the help of input gate layer and the input activation function given above. The next stage is to update the new cell state. This is followed by computing the output state, done by the output activation function.

3. Proposed System

The user is asked to provide the song for which he/she wants the sheet music generated. The system currently accepts English songs only.

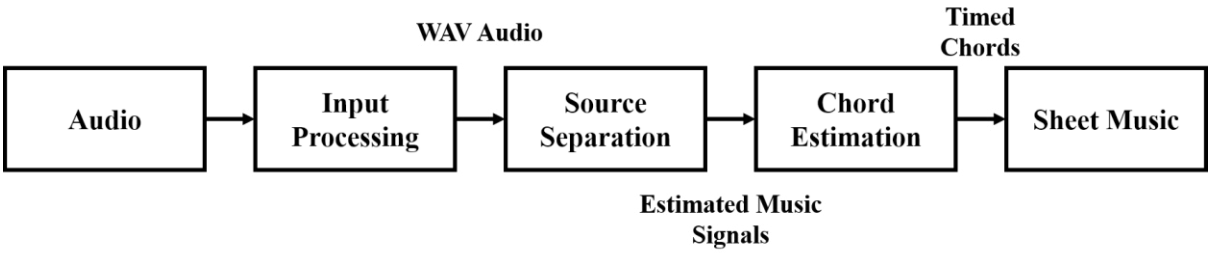

Fig 3: Proposed System Architecture

3.1. Initial Processing of Song

This input is accepted from the user in various audio formats. However, the format chosen for processing of the audio is .wav. This audio song in various formats is processed and converted into the required format using pyDub, which is the python library used for audio manipulation [17]. PyDub uses either ffmpeg or avconf for file conversion. In our system, we used ffmpeg. The working of ffmpeg is given by the following figure.

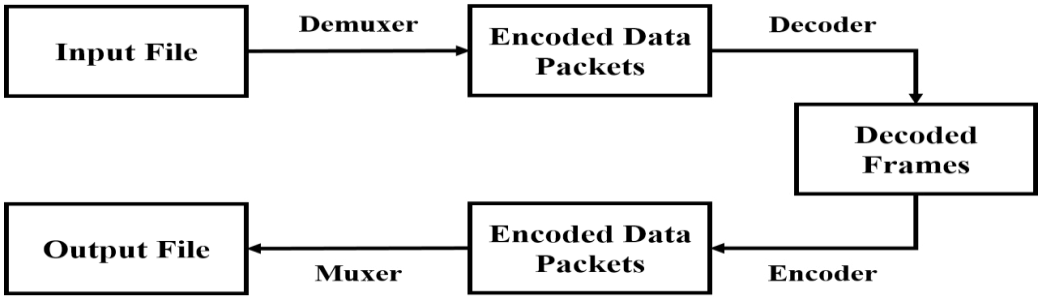

Fig 4: Process of audio conversion in ffmpeg

Ffmpeg calls the libavformat library which contains demuxers, and then reads input files and retrieves encoded data from the packets. These encoded packets are given to the decoder which then generates uncompressed frames for the filtering process that follows it. Post filtering process, the encoder is used to encode these packets and give out encoded packets as the final output. Finally, it is the muxer that writes these packets into the output file.

Further, pyDub makes use of AudioSegment class available as immutable objects that contain segments a particular segment for manipulation. Python code is, indubitably, used for this intent. This is simply a module that utilises wrapper of pydub AudioSegment object [18]. In reality, it is this object that makes use of ffmpeg as explained above.

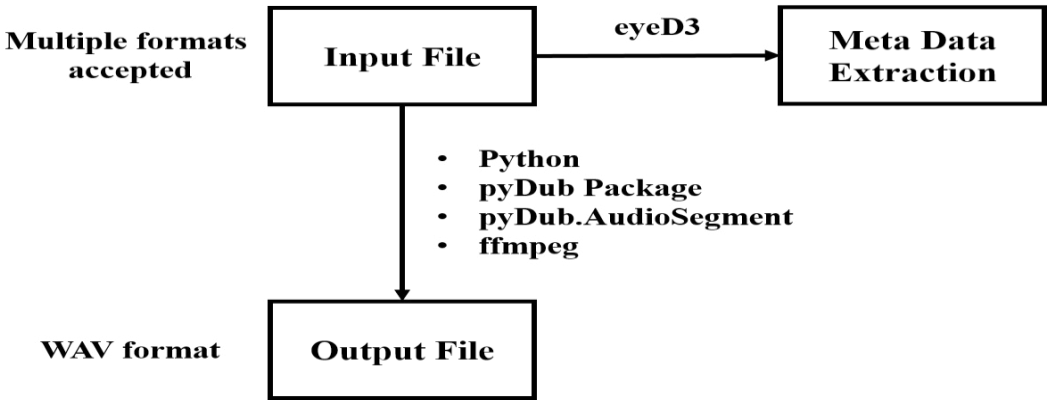

Fig 5: Input Process Module Working

Further processing of audio input requires conversion of audio in a single format, thus .wav format for the reason mentioned above. Thus, all the input audios are stored in .wav format post conversion. The converted audio is fetched further by the source separation module to distinguish and separate the file into two parts. The overall architecture and working of this module is discussed in the next section.

3.2. Source Separation

Typically, an audio source has two parts – vocal and non-vocal. We require separating these two parts of the source. We are utilizing DSD100 data set for training which has 100 songs of 44.1 kHz sample rate each with non-vocal part divided into its component – bass, drum and other [19]. There are a lot of methods to perform source separation and majority of them rely on the assumption that these sources, when expressed in matrix form of time frequency series, have a low rank and are sparse but this assumption is invalid for a majority of music. We will be using Recurrent Neural Network (RNN) with discriminative training for four sources of the audio – bass, drum, music and vocal. We train our model to take magnitude spectra of mixture of audio sources as an input and outputs the estimated magnitude spectra of the component sources – bass, drum, music and vocal.

3.2.1 Model

Our model uses magnitude of the spectrogram of audio signals as feature. For a training epoch t,as an input we pass magnitude spectra of the mixture signal $\mathcal{M}_t$ and magnitude spectra of the target sources -B_t (bass), D_t (drum), M_t (music) and V_t (vocal). As output of our network for the epoch t we have estimated signals for each source - $\widehat{B_t}$, $\widehat{D_t}$, $\widehat{M_t}$ and $\widehat{V_t}$.

We make our model learn to separate all of the component sources together instead of creating separate networks for each source. While training, an audio file is picked from the dataset along with its component sources audio file. They are read as time-series audio signals. These signals are then partitioned based on defined sampling rate (S_r) and duration (T), resulting in a signal having $S_r * T$ data pieces. For each such partition of each source we compute corresponding spectrogram (frequency domain) of the partitioned signal (time domain) using Short-time Fourier transform (STFT) [20]. For our dataset we have set the window and hop size for STFT as 2048 and 1024 respectively based on observation [21].

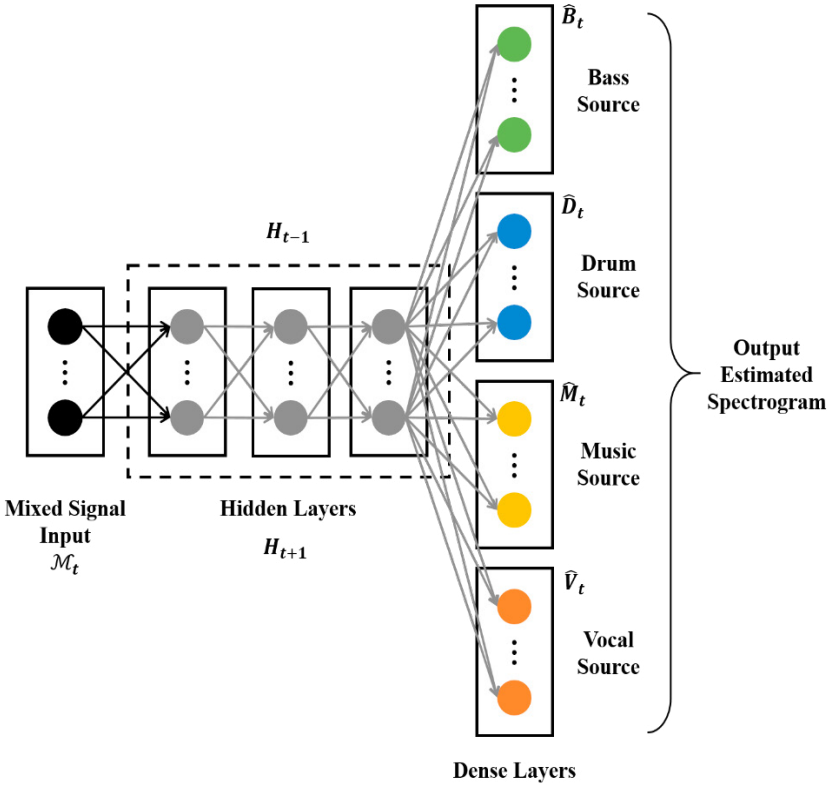

Fig 6: Source Separation RNN Model

For a training epoch t these single entry in the batch are denoted as $\mathcal{M}_t$ (mixed), B_t (bass), D_t (drum), M_t (music) and V_t (vocal). Mixed signals are passed to a multi-RNN cell containing 3 GRU (Gated Recurrent Unit) cells each of 256 units. This multi-RNN cell is connected to dense layer for each source. The dense layer is using ReLU (Rectified Linear Unit) as the activation function and outputs the estimated signal for each of the sources as $\acute{B}_t$, $\acute{D}_t$, $\acute{M}_t$ and $\acute{V}_t$[22]. The estimated signals are further processed to perform optimization on the masking function together with estimation as follows:

$$\widehat{B_t} = \frac{\acute{B}_t}{\acute{C}_t} \odot \mathcal{M}_t \tag{4}$$

$$\widehat{D_t} = \frac{\acute{D}_t}{\acute{C}_t} \odot \mathcal{M}_t$$

$$\widehat{M_t} = \frac{\acute{M}_t}{\acute{C}_t} \odot \mathcal{M}_t$$

$$\widehat{V_t} = \frac{\acute{V}_t}{\acute{C}_t} \odot \mathcal{M}_t$$

where, $C_t = \acute{B}_t + \acute{D}_t + \acute{M}_t + \acute{V}_t$ is the combined estimated magnitude spectra and $\odot$ is the elementwise multiplication operator, also known as Hadamard product.

3.2.2 Network Optimization

At each training step loss is calculated using mean square error function of target source signal and estimated source signal. This loss is minimized using Adam optimizer with an initial learning rate of 0.001[23]. We optimize our neural network to increase similarity between estimated and source signal of the target source and decrease similarity between the estimated signal of one source and signal of other target sources. This is called discriminative training and results in high signal to interference ratio (SIR) in the estimations which is desirable in source separation. Considering two sets of sources $S_t = \{B_t, D_t, M_t, V_t\}$ (source signals) and $E_t = \{\widehat{B_t}, \widehat{D_t}, \widehat{M_t}, \widehat{V_t}\}$ (estimated signals) and discriminative parameter γ (0.01) we compute loss as follows:

$$L = \sum_{\substack{e_t \in E_t \\ s_t \in S_t}} \left(\|e_t - s_t\|_2^2 - \gamma \times \sum_{\acute{s}_t \in S_t - \{s_t\}} \|e_t - \acute{s}_t\|_2^2 \right) \tag{5}$$

3.2.3 Generating Sources

To separate the source signals of a mixed signal, its magnitude spectrogram ($\mathcal{M}_t$) is given as an input to our neural network and the estimated output sources are used to calculate mask of each source. This mask is computed using soft time-frequency masking technique [24]. The mask is applied to the magnitude spectrogram of the mixture signal to calculate magnitudes of different sources. Mask for each source is calculated as follows:

$$m_{S_t} = \frac{\widehat{S_t}}{\sum \widehat{S_t}} \tag{6}$$

where, $\widehat{S_t}$ is estimated magnitude spectrogram of the source.

Magnitude spectrogram of each source is calculated using corresponding mask as follows:

$$C_{S_t} = m_{S_t} \odot \mathcal{M}_t \tag{7}$$

Phase spectrogram of the mixed signal is computed from the magnitude spectrogram by taking inverse tangent of the imaginary and real parts of each complex entry in the magnitude spectrogram. Resulting time-series signal for each source is computed as follows:

$$S_t = InverseSTFT(C_{S_t} \odot \exp(i \odot P_{S_t})) \tag{8}$$

Phase spectrogram is element wise multiplied with i and then its elementwise exponentiation is computed which is elementwise multiplied with the computed magnitude spectrogram in Eq. 4, this is done for each source.

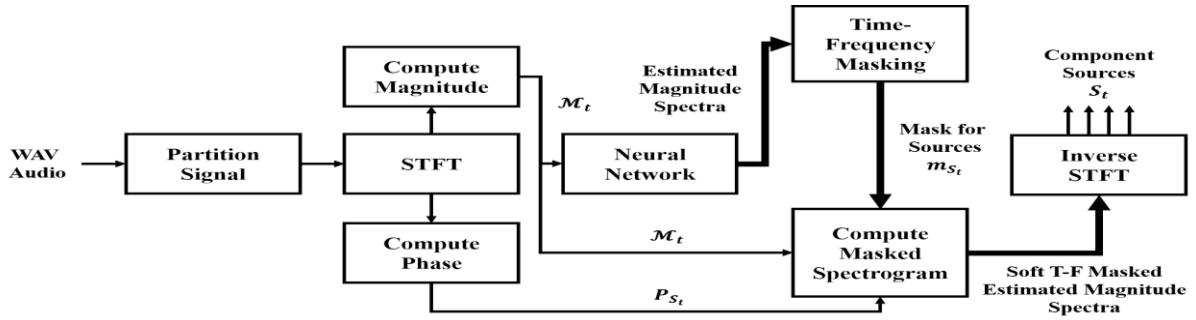

Fig 7: Source Separation Process

The resulting estimated component sources are then written into WAV file and are also used for performing BSS (Blind Source Separation) evaluation [25].

The resultant non vocal file is sent to the chord estimation module whereas the vocal one is used for pause detection and arrangement of time stamp lyrics.

3.3 Chord Estimation

The basic unit of a chord is pitch. Pitch is the perceived quality of a sound which is basically a function of the sound's fundamental frequency. A set of pitches that is harmonic and consists of two or more notes or pitches heard as if sounding simultaneously, is called a chord. A lot of musical instruments use chords as their basis such as guitar, piano etc. The non-vocal part obtained from the source separation method is all made up of combination of some musical instruments which in turn consist of several chords played in a series to give a pleasant music as a result. Our system automatically generates the chords with the help of an input music processed by the feature extraction and pattern matching which are trained on a dataset.

In feature extraction, a portion of the input sound stream is transformed to a DFT which stands for Discrete Fourier Transform [26]. This spectrum generated is presented as pitch class profile (PCP). PCP is a vector that shows the intensities of twelve semitone pitch classes, since it is a twelve-dimension vector [27]. The calculation of PCP is done from the following equation:

$$PCP(p) = \sum_{s.t.M(l)=p} ||X(l)||^2 \tag{9}$$

where, *p* refers to a pitch class, *X(l)* refers to the DFT spectrum, and *M(l)* is basically a table. *M(l)* maps the linear N point DFT frequencies to the non-linear pitch class frequencies. *M(l)* is calculated as:

$$M(l) = round\left(12 * \log_2(f_s . \frac{l}{n})/f_{ref}\right) mod 12 \tag{10}$$

where, $l = 1, 2, ...(N/2 - 1)$, f_{ref} is the reference frequency that corresponds to $p = 0$ and f_s refers to the sampling frequency. This can become a Short Term Fourier Transform if this process is repeated for every consecutive portion of the input. The output of STFT is a series of PCPs. Since the signals from the audio can have variable length window, another DFT transform is incorporated to take this factor into account and favours music signals more than the former method. Constant "Q" cycles are captured by the variable sized windows:

$$A(k) = \frac{1}{N(k)} \sum_{n=0}^{N(k)-1} B(k,n)x(n)\, e^{-j2\pi Qn/N(k)} \tag{11}$$

where, *A(k)* is the k[th] constant-Q spectrum bin, *N(k)* is the k[th] window size, *B(k,n)* is the k[th] window, Q is a constant and *x(n)* is the n[th] input sample. This process is known as constant-Q transformation (CQT)[28].

Feature extraction is done using Hidden Markov models which perform segmentation on the chromagram of the

audio signal [29]. The output given by our model is a collection of time stamped chords based on the input music component of the audio from source separation module.

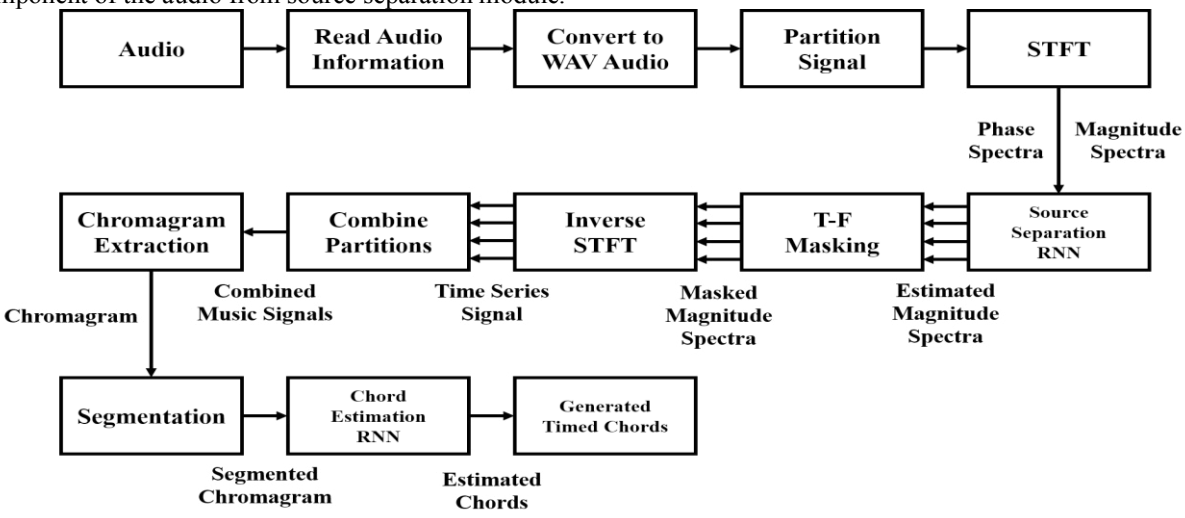

Fig 8: Overall process

4. Experimental Results

Evaluation of source separation process is done using BSS (Blind Source Separation) evaluation metric which is a standard tool for measuring the performance of source separation [25]. It uses original source signals as ground truth signals. The estimated signal is decomposed into its contribution to the target source and noise. Using DSD100 dataset along with extensive discriminative training we were able to significantly improve upon the previous works with our proposed model [19]. Below are the BSS evaluation results, higher is better, of our model compared with previous work:

Table 1. Comparison of proposed model with model outlined in [30]

Model	SDR (signal-to-distortion ratio)			
	Bass	Drum	Music	Vocal
Outlined in [30]	2.89	4.00	3.24	4.86
Proposed Model	6.38	7.68	5.45	9.2

Evaluation of chord estimation is done on the basis of accuracy of estimated chord annotations, there note and duration, as compared with the original version. We were able to increase this accuracy up to 78% by using RNN with LSTM cells trained on self-sourced dataset.

5. Conclusion

We have achieved the objective of improving an existing sheet music generator. The intent of enhancing this system has been fulfilled by working on two modules of the system. The results for these modules as compared to the ones for the modules in the previous systems are better by a considerable extent.

Sheet music generator is still not accurate to the considerable levels and leaves a large room for improvement. As future work, the lyric arrangement can be inculcated in the system itself so as to reduce user overhead. Currently, the user provides the lyrics along with the song. This input can be reduced to just the song. Secondly, there is no pause detection mechanism in the system, which can be worked upon on. Additionally, time synchronization of lyrics and alignment of lyrics and chords accurately can be done. This system currently accepts English songs and has been trained to accept the same. Thus, it can further be extended to songs in other languages.

References

[1] J. Lee and J. Downie (2004), "Survey of Music Information Needs, Uses, And Seeking Behaviours: Preliminary Findings," *ISMIR*

[2] Patel, A. (2003), "Language, music, syntax and the brain," *Nature Neuroscience* (**6**): 674-681

[3] Fremerey, C., M., M., & M., C. (2009), "Towards Bridging the Gap between Sheet Music and Audio." *Knowledge Representation for Intelligent Music Processing*.

[4] Suzy, S., Neuman, B., & Megan, L. (2017, August 10), "The Importance of Sheet Music to Music Theory." Retrieved from *TakeLessons Blog: https://takelessons.com/blog/sheet-music*

[5] Huron, D. (2001), "Is music an evolutionary adaptation?" *Annals of the New York Academy of Sciences* (**930**): 43-61.

[6] Turner, K. M. (2017), "Review: Historic American Sheet Music," *Journal of the American Musicological Society*, **70** (**2**): 565-575.

[7] Cohen, E., & L., K. A. (1990), "Old age in America represented in nineteenth and twentieth century popular sheet music." *The Gerontologist*, **30** (**3**): 345-354.

[8] Gan, T. (2005), "Musica colonial: 18th century music score meets 21st century digitalization technology," *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '05, Denver*.

[9] Luth, N. (2002). "Automatic Identification of Music Notations," *WEDELMUSIC*.

[10] Hsu, Y. L., Lin, C. P., Lin, B. C., Kuo, H. C., Cheng, W. H., & Hu, M. C. (2017), "DeepSheet: A sheet music generator based on deep learning," *Multimedia & Expo Workshops (ICMEW)*: 285-290, Hong Kong: IEEE.

[11] Hori, T., Nakamura, K., & Sagayama, S. (2017), "Music chord recognition from audio data using bidirectional encoder-decoder LSTMs," *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*.

[12] Leglaive, S., Hennequin, R., & Badeau, R. (2015), "Singing voice detection with deep recurrent neural networks," *Acoustics, Speech and Signal Processing (ICASSP). Brisbane, Australia*.

[13] Camron, G. (2016, August 12), "Recurrent Neural Networks for Beginners" – Camron Godbout – Medium. Retrieved from *https://medium.com/@camrongodbout/recurrent-neural-networks-for-beginners-7aca4e933b82*

[14] Hochreiter, S., & Schmidhuber, J. (1997), "Long short-term memory," *Neural Computation*, **9** (**8**): 1735-1780.

[15] Lipton, Z. C., Berkowitz, J., & Elkan, C. (2015), "A critical review of recurrent neural networks for sequence learning," *Cornell University Library*.

[16] Sak, H., Senior, A., & Beaufays, F. (2014), "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *International speech communication association*.

[17] James, R. (2011, May 2), jiaaro/pydub. Retrieved from *https://github.com/jiaaro/pydub*

[18] James, R., audio_segment - pydub 0.9.5 documentation. (Pydoc.io) Retrieved from *https://www.pydoc.io/pypi/pydub-0.9.5/autoapi/audio_segment/index.html*

[19] Liutkus, A., Stöter, F. R., Rafii, Z., Kitamura, D., Rivet, B., Ito, N., ... & Fontecave, J. (2017, February). The 2016 signal separation evaluation campaign. In International conference on latent variable analysis and signal separation (pp. 323-332). Springer, Cham.

[20] Griffin, D., & Lim, J. (1984), "Signal estimation from modified short-time Fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **32** (**2**): 236-243.

[21] Smith, J. O., & Serra, X. (1987), "PARSHL: An analysis/synthesis program for non-harmonic sounds based on a sinusoidal representation," 290-297, *CCRMA, Department of Music, Stanford University*.

[22] Abien, F. A. (2018), "Deep Learning using Rectified Linear Units (ReLU)," *arXiv preprint arXiv:1803.08375* .

[23] Diederik, P. K., & Ba, J. (2015), "Adam: A Method for Stochastic Optimization," *3rd International Conference for Learning Representations. San Diego*.

[24] Toroghi, R. M., Faubel, F., & Klakow, D. (2012), "Multi-channel speech separation with soft time-frequency masking," *SAPA-SCALE Conference*.

[25] Pal, M., Roy, R., Basu, J., & Bepari, M. S. (2013), "Blind source separation: A review and analysis," *2013 International Conference Oriental COCOSDA held jointly with 2013 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE). Gurgaon: IEEE*.

[26] Takuya, F. (1999), "Realtime chord recognition of musical sound: A system using common lisp music," *Proceedings of the 25th International Computer Music Conference*: 464-467.

[27] Shepard, R. N. (1964), "Circularity in judgments of relative pitch," *The Journal of the Acoustical Society of America*, **36** (**12**): 2346-2353.

[28] Brown, J. C. (1991), "Calculation of a constant Q spectral transform," *The Journal of the Acoustical Society of America*, **89** (**1**): 425-434.

[29] Raphael, C. (1999), "Automatic segmentation of acoustic musical signals using hidden Markov models," *IEEE transactions on pattern analysis and machine intelligence*, **21** (**4**): 360-370.

[30] Uhlich, S., Porcu, M., Giron, F., Enenkl, M., Kemp, T., Takahashi, N., & Mitsufuji, Y. (2017, March), "Improving music source separation based on deep neural networks through data augmentation and network blending," *Acoustics, Speech and Signal Processing (ICASSP)*: 261-265), IEEE.