Audio Source Separation for Improved Tabla and Vocal Transcription in Vocal Mixtures

Dissertation

submitted in partial fulfillment of the requirements
for the degree of

Bachelor & Master of Technology

by

Aniruddha Deshpande

Roll No: 150010005

under the guidance of

Prof. Preeti Rao

Department of Electrical Engineering
Indian Institute of Technology Bombay

2020

Dissertation Approval Sheet

This is to certify that the dissertation titled

Audio Source Separation for Improved Tabla and Vocal Transcription in Vocal Mixtures

By

Aniruddha Deshpande
(150010005)

is approved for the degree of **Bachelor & Master of Technology**.

Prof. Preeti Rao
(Guide)

Internal Examiner

External Examiner

Chairperson

Date : _____

Declaration of Authorship

I, **Aniruddha Deshpande** declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

_____

Signature

_____

Roll Number

Date: _____

Abstract

This thesis deals with performing tabla and vocal source separation on Hindustani vocal mixtures with the aim of obtaining separated tracks that yield well to the existing tabla and vocal transcription algorithms. This is motivated by the unsatisfactory performance of existing transcription algorithms on such mixtures and the lack of available source separation models specific to this genre. We build a dataset of Hindustani vocal mixtures composed of vocals, tanpura and tabla. We consider a state-of-the-art source separation method, trained on these vocal mixtures with the corresponding tabla-only target, to find that we get a significantly improved performance for onset detection and stroke classification over that achieved on the mixture. In an attempt to improve the accuracies further with reference to those obtained with the original unmixed tabla, we investigate influencing the deep learning representation for source separation by an additional task-specific loss, namely, the onset detection function cross-entropy, via a suitably designed multitask learning network. We perform vocal separation by removing the tabla estimate from the mixture using Wiener filtering, and then suppressing the tanpura using an available noise reduction tool. Vocal source separation was evaluated on the basis of pitch detection performance of two well known pitch tracking algorithms on the separated vocals, as compared to the corresponding clean vocals.

Keywords: Audio source separation, machine learning, audio transcription, dataset

Acknowledgments

First and foremost I would like to thank my guide **Prof. Preeti Rao** for her constant guidance and support throughout the last year. She would ensure to regularly meet, discuss and provide critical feedback on the progress of the work, which played a central role in shaping the course of the project.

I would also like to thank Rohit, for his help in performing the onset detection and stroke classification evaluations, debugging code and brainstorming ideas.

A special thanks to all the musicians who took the time out to provide us with their recordings, without which our work could not have progressed.

I'd also like to thank everyone in DAPLAB for making it a great environment to work in. I would especially like to thank Sachin and Rohit for their help in setting up my workspace and in resolving any system issues I faced and Krishna for helping me resolve Python and Latex issues much faster than I would on my own. I'd also like to thank Prof. Preeti Rao, Prof. Rajbabu, Rohit and everyone else who ensured that the lab facilities kept running smoothly even during the pandemic.

Lastly I would like to thank my family and friends whose support helped me get through the most stressful periods of the project.

Aniruddha Deshpande
IIT Bombay
(Date)

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Vocal performances, accompanied by the tabla providing the rhythmic framework, are a significant component of the repertoire of khyal, the North Indian classical music genre. Musicological studies of the metrical structure and its relationship to the melodic context can benefit from MIR tasks such as automatic transcription of the tabla, which includes detecting the sequence of tabla onsets and the type of stroke at each onset, and of the vocals which involves obtaining the pitch contour corresponding to the melody. However, given that the performances involve musicians improvising together, we typically have access only to the mixed audio recordings for this musical genre. Even if there are separate mics for the tabla and the vocalist during audio recording, there is usually significant leakage from other sources. Tabla transcription will suffer because standard onset detection algorithms when applied on such an audio are likely to pick out onsets corresponding to the vocals and other accompanying instruments along with tabla onsets. Stroke classification algorithms, which analyse the spectral representation of the audio in the vicinity of the detected stroke onset are also likely to be prone to errors because of the interference from vocals and other instruments. Vocal transcription would also suffer because of spectral leakage from other sources. This motivates the need to develop an audio source separation system that is able to isolate the tabla and the vocals, so we can perform relevant MIR tasks on the individual streams as opposed to performing them on the mixture audio.

Audio source separation in the context of music has been widely studied and has numerous applications such as automatic karaoke, remixing, up-mixing, pre-processing for further analyses on individual streams and others. In recent years various kinds

of deep learning methods have had great success in source separation both in terms of objective evaluation metrics [3] as well as perception. The problem however with available deep learning models trained on data-sets such as MUSDB18 [4], DSD100 [5] which are primarily Western music data-sets is that they fail to generalise to kinds of music and instruments that are not represented in them. An example of this is that when we used the pre-trained Spleeter [6] vocals model on Hindustani vocal mixtures, we could see tabla harmonics (especially those of the right drum stroke, which has higher spectral content) in the estimated 'vocals' track. This is because typical drums, which is the prominent percussion instrument in the data used for training Spleeter do not have the tonal property of tabla strokes. Given that we are interested in studying Hindustani vocal concerts, we could not directly use available trained source separation models for this reason. Hence, there was a need to develop a source separation system that is trained on a dataset comprised of tracks from this genre, and also to synthesise such a dataset from available sources.

1.2 Approach

We consider a state-of-the-art source separation architecture and train it on a dataset of Hindustani vocal concert audios. Since such a dataset is not freely available, we create it by mixing vocals, tabla and tanpura obtained from various sources. Different models are trained for separating tabla and vocal targets.

In this thesis, we attempt to move away from the most commonly adopted view of source separation as an end in and of itself. Source separation quality is often judged using the same standard metrics, irrespective of the task for which it is being developed. We aim to develop a source separation system that separates the tabla and vocal tracks in such a way that the MIR tasks described above perform well when applied on these separated streams, and we evaluate source separation quality by comparing the accuracies of MIR tasks applied on separated tabla/vocals with those obtained by applying the same on clean (unmixed) tabla/vocals.

For tabla separation, we exploit multi-task learning to bring in a loss function that relates to the transcription task. Our proposed architecture thus has two branches which perform source separation and tabla onset detection tasks respectively. Adding an onset detection loss would drive the network to learn features that better preserve onset information, which in turn would provide separated tabla that has better preserved onsets and stroke information.

For vocal separation, we first obtain an estimate of the vocals+tanpura audio, by removing the tabla estimate from the mixture using Wiener filtering. The vocal estimate is obtained from this by suppressing the tanpura using an available noise reduction tool.

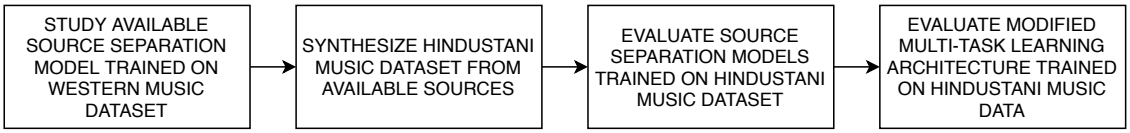


Figure 1.1: Contributions in this thesis

1.3 Organisation

The thesis is organised as follows. Chapter 2 contains the literature survey on the source separation problem. Chapter 3 details the used source separation network architecture and the proposed multi-task learning modification. Chapter 4 describes the dataset that we have developed and used for training. Chapters 5 and 6 contain experiments, observations and results corresponding to tabla and vocal separation respectively. Chapter 7 contains a literature survey on musical complexity, a future application for which we would like to use the developed source separation system. Chapter 8 presents the conclusion and potential extensions of this work.

Chapter 2

Source separation overview

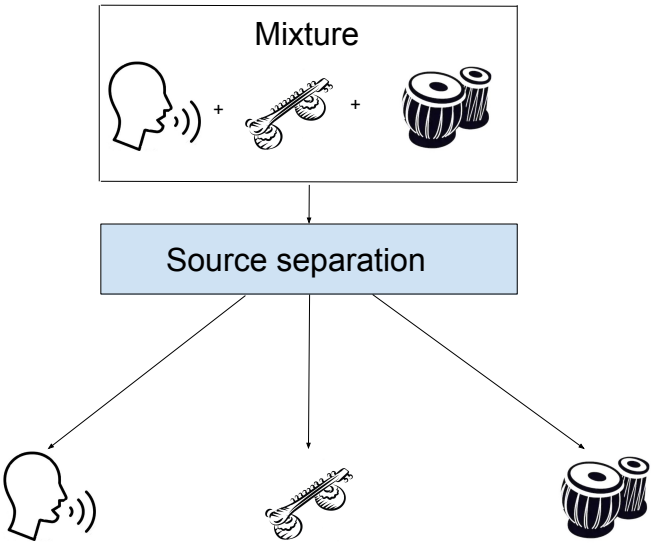


Figure 2.1: The source separation problem shown for the typical Hindustani concert audio with vocals accompanied by tabla and tanpura

The source separation problem has been studied in various contexts over the years. The aim of the problem is to separate the signal corresponding to a given source from a mixture of signals coming from different sources. Let $s_i(t)$ $(1 \leq i \leq n)$ be the signal corresponding to the i^{th} source. The aim is to recover $s_i(t)$ from $x(t) = \sum_{i=1}^{n} a_i s_i(t)$. In our case we want to be able to obtain an estimate of the tabla and vocals from a Hindustani classical music track, which will be a mixture of vocals, tabla and tanpura.

Research in source separation for audio started off being motivated by the problem of speech enhancement, which involves obtaining a clean speech signal from a noisy

recording[7]. The problem of separating vocals from instruments however, is a more difficult problem than the former, as the former can be solved assuming the background noise to be stationary and uncorrelated with the speech signal. In the case of separation of music audio however, the other musical instruments in the background are non-stationary as well as highly correlated with the target instrument to be separated. It is clearly non-stationary because a guitar for example can be playing a different chord or a different pitch at a different time and correlated, because for the song to be audibly pleasing there needs to be some relation between the different music sources in the mixture, like having the same key, tempo and time signature. The applications for separation of vocals [7] and other instruments from a mixture are many, including automatic karaoke, music up-mixing, remixing and others.

Over the years several methods have been used to tackle the music source separation problem, including but not limited to Non-negative matrix factorisation, harmonic methods, but in recent years deep-learning and data driven methods have shown the best performance,both in terms of perception as well as objective measures to evaluate the quality of separation. This is why we have chosen a deep learning based approach towards this problem.

In this chapter, we first review the different kinds of source separation techniques, followed by which we discuss the different measures we will be using to evaluate the quality of separation.

2.1 Source separation methods

2.1.1 Modelling the lead signal: Harmonicity

This method exploits the fact that vocal melodies are harmonic and will have a fundamental frequency associated with them. The approach involves estimating this fundamental frequency at every time frame and also then estimating the energy of the higher harmonics. This information is then used to reconstruct the lead signal.

The fundamental frequency is estimated using suitable pitch detection methods or using available score of the music. The reconstruction of the voice is done either by sinusoidal synthesis or by filtering out any parts of the signal that do not lie close to the detected harmonics. It's performance may also be affected in the case where some of the accompaniment is harmonic as well.

These methods have a few shortcomings associated with them. First, is the assumption that the vocal signal is always harmonic, as certain phonemes may be unvoiced,

whispered or saturated making it difficult to deal with a harmonic model. The second is that it heavily relies on the pitch detection algorithm working well on a mixture.

2.1.2 Modelling the accompaniment as redundancies

This set of methods is based on the assumption that the accompaniment is somehow more redundant than the lead signal, in that they are likely to be more structured and have a repeating pattern. This could involve using non-negative matrix factorisation to decompose the mixture spectrogram, which is then used to identify the different components of the mixture. These individual components are then re-synthesised to give the lead and accompaniment signals.

These methods however, are unable to take into account the situations where there are accompaniments that may not always be repetitive or redundant, and occur less frequently in the audio. They also do not take into account the fact that the lead signal may also be redundant or repetitive and fail to handle these cases.

Another set of methods also exist which jointly model the signal as being harmonic and the accompaniment to be redundant.

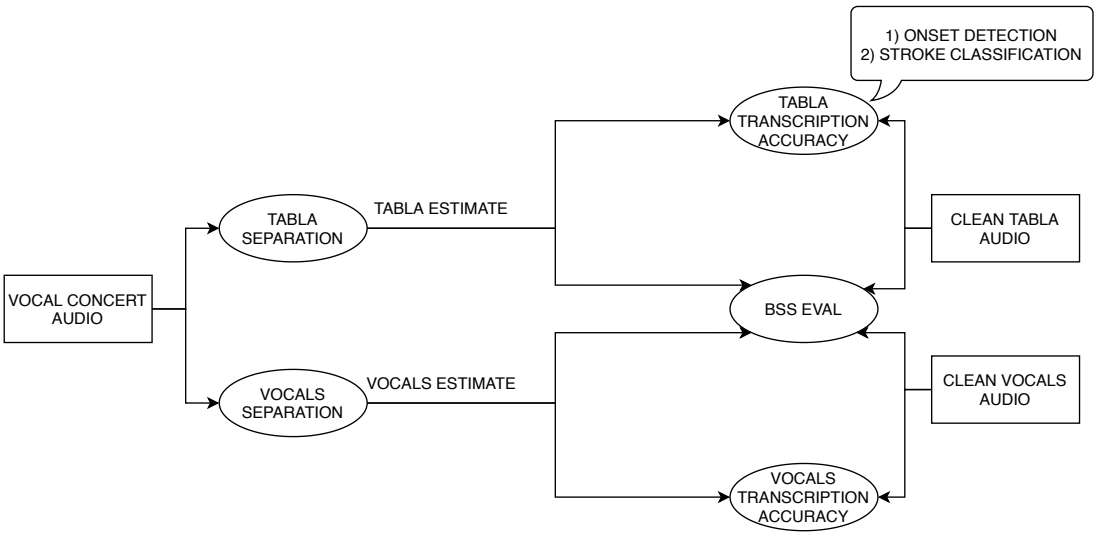
2.1.3 Data driven approach

Data driven approaches use large amount of data, containing information about the mixture and the individual components. Machine learning models are then trained using this data to learn how it should perform the separation. Typically, these methods involve using the network to estimate a time frequency mask, which when applied over the input mixture spectrogram gives an estimate of the spectrogram of the desired component.

2.2 Evaluation measures

Evaluating the results of source separation has been a difficult problem for quite some time. There is no one objective metric to evaluate source separation results. The BSS eval metrics [3] [8] which have been used as a part of the Signal Separation Evaluation Campaign (SiSec) are the most commonly used source separation quality evaluation measures. These measures involve projecting the estimate signal into noise and signal sub-spaces and using these quantities to come up with various SNR terms, each having a specific significance. We will also use specific measures for tabla and vocal separation that evaluate the accuracy of transcription obtained by applying standard transcription

algorithms on these separated tracks. These measures are important because we are developing our source separation system specifically as a pre-processing step for transcription. All of the evaluation measures described in the following sections require access to the corresponding clean target audios.



2.2.1 BSS eval

The BSS eval metrics involve decomposing the estimated signal into various components and using these components to calculate some SNR values. Let,

$$x(t) = \sum_{j=1}^{n} a_j s_j(t) + n(t) \tag{2.1}$$

where, $x(t)$ is the linear mix of the sources and $n(t)$ is measurement noise. $\hat{s_j}(t)$, the estimate for the j^{th} signal source can be decomposed as :-

$$\hat{s_j}(t) = s_{target} + e_{interference} + e_{noise} + e_{artifact} + e_{spat} \tag{2.2}$$

This decomposition can be based on orthogonal projections. Consider the vector spaces :-

$$P_{s_j} := \Pi\{s_j\} \tag{2.3}$$

$$P_{\mathbf{S}} := \Pi\{(s_{j'})_{1 \leq j' \leq n}\} \tag{2.4}$$

$$P_{\mathbf{S},\mathbf{n}} := \Pi\{(s_{j'})_{1 \leq j' \leq n}, (n_i)_{1 \leq i \leq m}\} \tag{2.5}$$

P_{s_j} is the vector space defined by the original source vector, $P_{\mathbf{S}}$ is the vector space defined by all the source vectors and $P_{\mathbf{S},\mathbf{n}}$ is the vector space defined by all the source vectors as well as the measurement noise vectors. Since in our case we have only one measurement, we put $m = 1$. The four decomposition terms are given by :-

$$s_{target} := True\, source\, signal \tag{2.6}$$

$$e_{interf} := P_{\mathbf{S}}\hat{s_j} - P_{s_j}\hat{s_j} \tag{2.7}$$

$$e_{noise} := P_{\mathbf{S},\mathbf{n}}\hat{s_j} - P_{\mathbf{S}}\hat{s_j} \tag{2.8}$$

$$e_{artif} := \hat{s_j} - P_{\mathbf{S},\mathbf{n}}\hat{s_j} \tag{2.9}$$

$$e_{spat} := P_{s_j}\hat{s_j} - s_{target} \tag{2.10}$$

The global performance measures for source separation are given by the following :-
Source to Distortion Ratio

$$SDR := 10\log\frac{||s_{target}||^2}{||e_{interf} + e_{noise} + e_{artif} + e_{spat}||^2} \tag{2.11}$$

Source to Interference Ratio

$$SIR := 10\log\frac{||s_{target} + e_{spat}||^2}{||e_{interf}||^2} \tag{2.12}$$

Source to Artifacts Ratio

$$SAR := 10\log\frac{||s_{target} + e_{interf} + e_{noise} + e_{spat}||^2}{||e_{artif}||^2} \tag{2.13}$$

Source image to Spatial distortion Ratio

$$ISR := 10\log\frac{||s_{target}||^2}{||e_{spat}||^2} \tag{2.14}$$

The SDR and SIR terms are particularly useful for evaluating source separation. It's benefits over other existing methods are the following :-

1. They range from $-\infty$ to $+\infty$
2. Does not make any assumptions on the type of method used for separation
3. The different quantities defined above allow to distinguish between different kinds of estimation errors

2.2.2 Tabla transcription accuracy

We are performing tabla source separation with the aim of performing accurate tabla transcription on Hindustani vocal concert mixtures. Hence, we would like to evaluate the quality of a tabla source separation system based on how available tabla transcription algorithms perform on source separated tabla as compared to their performance on clean (unmixed) tabla. Tabla transcription has two components, first is onset detection, which gives the time instants at which tabla strokes occur and the second is stroke classification, which gives the sequence of the types of strokes that are played. The algorithms and evaluation measures used corresponding to both of these are described below.

Onset Detection

To perform onset detection, we use a spectral flux based method [9]. A novelty curve is first obtained by calculating the flux on the STFT magnitude of the audio signal computed using a window size of 92ms and a hop size of 23ms. Peaks in the novelty above a selected fixed threshold are then marked as onsets. We will refer to this series of steps as the OD algorithm. The ground truth tabla onsets for the test set are obtained by applying the OD algorithm on the clean tabla audios and manually correcting any errors through a combination of observing the spectrogram and listening to the audio.

For evaluation, the OD algorithm is applied to the outputs of the source separation model and the predicted onsets are compared to the ground truth onsets. The performance is evaluated using precision, recall and f-score metrics, where a predicted onset is considered a hit if it lies within ±25ms of an unmatched ground truth onset and a false alarm otherwise, as in [10]. The fixed threshold value is varied from 0.1 to 0.9 to obtain a precision-recall curve, from which the values at the point of best f-score are reported. The precision, recall and f-score values are defined as :-

1. **Precision:** It is the ratio between the number of frames that are correctly predicted as onsets to the total number of frames that are predicted as onsets ($= \frac{TP}{TP+FP}$).
2. **Recall:** It is the ratio between the total number of frames that are correctly predicted as onsets to the total number of ground truth onset frames ($= \frac{TP}{Total\,GT\,onsets}$).
3. **F-score:** It is the harmonic mean of precision and recall ($= \frac{2 \times precision \times recall}{precision + recall}$).

where, TP, FP and GT are true positive, false positive and ground truth respectively.

Stroke classification

One significant difference between tabla and common Western percussion instruments like the drums is the presence of low-frequency harmonic components in the tabla (see figure 2.2), which vary with the type of stroke played. This makes the problem of tabla source separation for transcription even more complicated than for its Western counterpart, as it involves preserving not only the wide-band energy burst corresponding to the onset, but also the harmonic components in the low-frequency regions right after the onset, which are crucial in identifying the type of stroke being played. For this reason we use tabla stroke classification accuracy as a measure of evaluating tabla source separation quality.

Tabla stroke classification, or bol transcription as it is sometimes known, commonly involves detecting stroke onsets and identifying the bol or stroke [11, 12]. In the present evaluation for stroke classification only, we use ground-truth onsets in order to focus on the stroke classification errors uninfluenced by onset location uncertainty. While the outputs of a tabla transcription system can encompass all the distinct tabla strokes (bols), an important level in the taxonomy of bols is the classification into resonant and damped strokes [13]. We restrict ourselves to the following four broad stroke categories - resonant bass, resonant treble, resonant both, and damped. This is based on the distinct musical roles that these stroke categories play in marking important sub-divisions in the cyclic pattern of the tabla accompanying the lead musician [14].

For stroke classification into the prescribed categories, the input audio is first separated into the two frequency bands: 50-150 Hz and 200-1000 Hz. These bands capture the major harmonic energies of the resonant bass drum and resonant treble drum strokes respectively. Next, at every specified onset instant, we extract the region from 20ms before the instant extending to the next onset instant in each filtered signal and carry out further analyses on these segments. Resonant and damped strokes can be distinguished from each other by the rate of decay of the spectral energy in two regions, one close to the onset instant and the other, 25 ms or more removed. Linear fits to the short-time logarithmic energy computed at 5 ms intervals are examined in terms of the best fitting slope values, separately in each frequency band. The estimated slopes are subjected to empirically selected thresholds to detect the presence of a resonant stroke separately in each band (corresponding to treble and bass strokes for the higher and lower bands respectively). If a resonant stroke is not detected in either, then the stroke is classified as being damped.

To obtain the ground truth stroke classes on the test set, the algorithm is applied

on the clean tabla audios, and any errors are manually corrected. The output of this algorithm when applied to the separated tabla audios from the various source separation networks is compared to the target strokes, and the 4-way classification accuracy is reported.

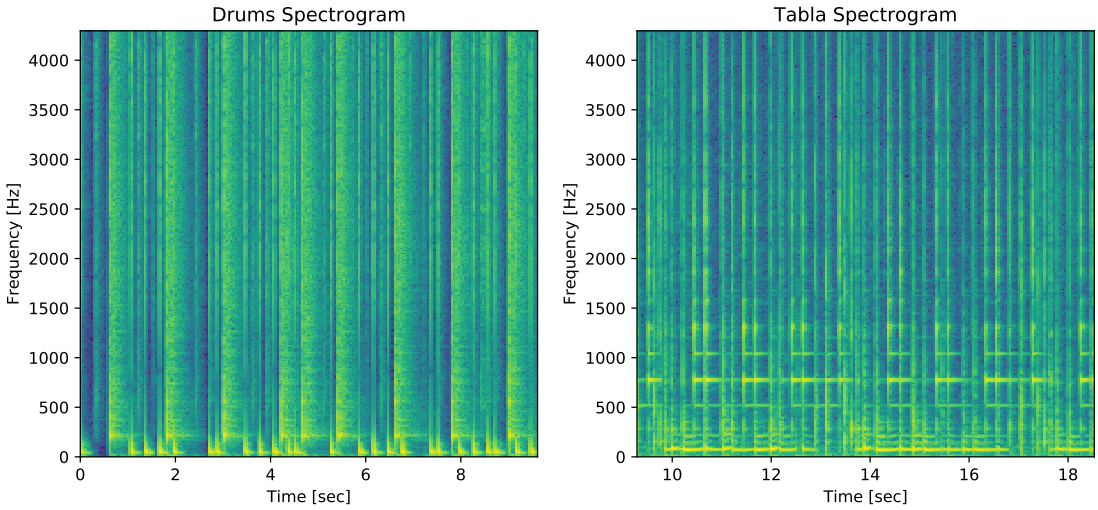


Figure 2.2: From left to right: Spectrogram of (a) Drums (b) Tabla. Note the harmonic components present upto ∼1.5kHz in the tabla spectrogram

2.2.3 Vocal pitch tracking accuracy

Similar to the above transcription based evaluation metrics for tabla separation, we will evaluate our vocals separation results by comparing the pitch track extracted from the estimate of the vocals obtained by the source separation network to the pitch track extracted for clean vocals (ground truth). We will be obtaining pitch estimates using the CREPE [15] and PYIN [16] algorithms.

A pitch tracking algorithm has two sets of outputs, the pitch track and the voicing confidence probabilities. The pitch track is just an array containing the frame-wise fundamental frequency estimates. The voicing confidence probabilities gives a number between 0 and 1 which gives the confidence with which the algorithm estimates a given frame to be voiced. If the value is greater than a given threshold the frame is estimated to be voiced, and if lower than the threshold it is detected as un-voiced. To evaluate the accuracy of a given pitch estimate, we account for the accuracies of both the pitch track as well as the voicing confidence estimates.

The performance of the pitch tracker in [15] is evaluated by applying the algorithm

on a sound track, whose exact pitch track is known and the estimated pitch is compared with this ground truth. The track with exact known pitch track is obtained using the analysis/synthesis framework presented in [17]. This involves using a pitch tracker on a given audio, and using this pitch track to re-synthesise the audio using sinusoids. However, for our purpose we will use the pitch track obtained by applying the algorithm on clean vocals as the ground truth, and then compare the pitch track obtained by applying the algorithm on the vocals extracted by our model.

In this section we first give a brief overview of the two pitch detection algorithms used, followed by a description of the measures used to evaluate pitch accuracy.

PYIN [16]

The Probabilistic YIN or PYIN [16] algorithm is a monophonic pitch tracking algorithm. It consists of a series of steps. The first step involves computing a difference function for each frame, which is proportional to the negative of the autocorrelation function of the frame. The position of the peaks of the autocorrelation function are indicative of the wavelength of the signal, hence the minimas of the difference function indicate the wavelength. Each minima value and position is then finetuned by parabolic interpolation, to account for the discretisation due to sampling. This is followed by a probabilistic thresholding step, and minimas with values lesser than the threshold are considered. If no minima is below the threshold, some probability is assigned to the event that the frame is unvoiced. This step gives a probability distribution of estimated pitches for each frame based on the distribution of thresholds used. The pitches are discretised into 480 bins, ranging from 55Hz to 880Hz (4 octaves), in steps of 10 cents (0.1 semi-tones). Each frame is now considered to be a timestep in an HMM, the pitches are the states of the HMM, and the probability distribution of these states is obtained from the thresholding step. For each pitch there are 2 states, corresponding to voiced and un-voiced states. The transisition probabilities are fixed such that it allows only upto a given amount of pitch difference between consecutive frames. Viterbi decoding of this HMM gives the estimated pitch track. The HMM smoothing serves to make it more robust to noise, but because the algorithm depends on the difference function, it performs poorly in a polyphonic setting. The difference function computed for the sum of two signals is significantly different than that for only one of the signals.

CREPE [15]

Convolutional REpresentation for Pitch Estimation (CREPE) is a deep learning based pitch detection model, which uses a CNN architecture. It is trained using two different datasets, the RW-synth, which has a fixed timbre over all tracks, and MedleyDB which contains tracks with realistic timbre. For each time frame, the model outputs a probability distribution over 360 discrete pitch values, ranging from 32.70Hz to 1975.5Hz (6 octaves) in 20-cent intervals. The model is trained by minimizing the binary cross entropy error between the predicted and target pitch vectors, where the target pitch vector is a 360 dimensional vector with the ground truth bin having a value of 1, which is Gaussian-blurred in frequency to reduce penalty for nearly correct predictions. The available CREPE implementation performs a temporal smoothing using Viterbi decoding. This model is found to be significantly more robust to different kinds of noises as compared to the PYIN algorithm.

Pitch accuracy measures

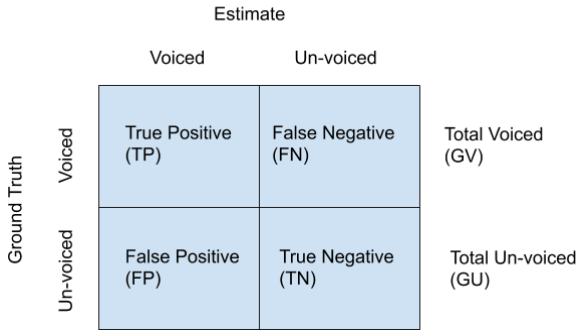


Figure 2.3: Confusion matrix of frame voicing estimates

In order to define the pitch evaluation metrics we will use the terms in figure 2.3. We use the following metrics to evaluate the performance of a given pitch track as compared to the ground truth. :-

1. **Voicing recall:** This is defined as the ratio between the number of frames that are correctly labelled as voiced in the estimate, to the total number of voiced frames in the ground truth($= \frac{TP}{GV}$).

2. **False alarm** - This is defined as the ratio between the number of frames that are

incorrectly labelled as voiced in the estimate, to the total number of un-voiced frames in the ground($= \frac{FP}{GU}$)

3. **Raw pitch accuracy:** This is defined as the ratio between number of voiced frames for which the estimated pitch is within 50 cents ($1cent = (\frac{1}{100})^{th} semitone$) of the ground truth value to the total number of voiced frames. This includes the pitch guesses for frames that were estimated as unvoiced($= \frac{TPc+FNc}{GV}$).

4. **Raw chroma accuracy:** This is defined as the ratio between number of voiced frames for which the estimated pitch is within 50 cents ($1cent = (\frac{1}{100})^{th} semitone$) of the ground truth value, or a different octave of the ground truth pitch to the total number of voiced frames. This includes the pitch guesses for frames that were estimated as unvoiced.

5. **Overall accuracy:** This measure combines the voicing accuracy measures along with the pitch accuracy measures. It is defined as the ratio between the sum of number of frames labelled as unvoiced and the number of frames labelled as voice, having estimated pitch within 50 cents of the ground truth to the total number of frames in the audio($= \frac{TPc+TN}{GU+GV}$).

The subscript c refers to frames for which pitch has been correctly estimated upto a given tolerance.

We have used the *mireval* [18] python library to compute the pitch track evaluation metrics.

Chapter 3

Network architecture

We have chosen the network architecture of *Open Unmix* [1] as the baseline for our task. The choice was made because this network gives state-of-the-art separation performance on using the MUSDB18 dataset [19]. However, this is not the only well-performing deep learning architecture. *Spleeter* [6] uses the U-net [20] architecture. *SVSGAN* [21] uses generative adversarial networks, in which a second network specifically trained to distinguish between the target source audio and the output of the source separation model is used to improve the performance of the source separation model. Both these models also give state-of-the art source separation performance. We choose *Open Unmix* over these, because of the easy availability of its open source implementation using Pytorch.

The *Open Unmix* network trained on MUSDB18 dataset performs very well on western music [1]. MUSDB18 is a data-set consisting of 150 songs (nearly 10 hours) belonging to different genres along with separate vocals, drums, bass and other instruments stems. This trained model's performance on Hindustani classical music however is quite poor (as shown in chapter 5). We propose to first train this network on a Hindustani music data-set, and then propose a multi-task learning based modification to this architecture that would help further improve tabla source separation performance in the terms of improved tabla transcription as mentioned in chapter 1.

The organisation of this chapter is as follows. We first provide a detailed description of the *Open Unmix* architecture, the training procedure employed and the hyper-parameters of interest. This is followed by an overview of the multi-task learning approach, which we will be using to modify the baseline architecture. The next section details the architectural modifications that we introduce. In the last section, we provide an overview of the pytorch implementation of the network architecture.

3.1 Open Unmix overview

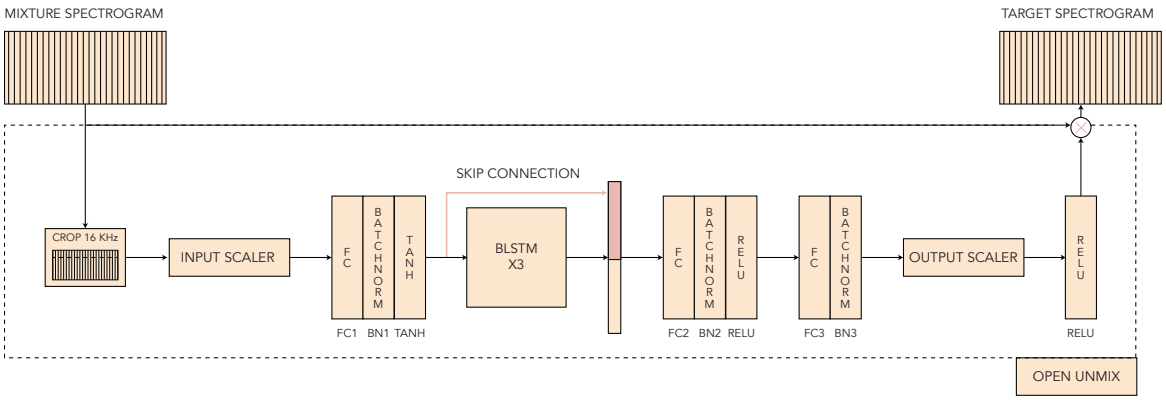


Figure 3.1: Open-unmix network architecture [1]

3.1.1 Architecture description

The network architecture is showed in figure 3.1. It consists of a tanh activated fully connected layer (FC1) followed by a 3-layer bidirectional LSTM (BLST), followed by 2 fully connected (FC2, FC3), Relu activated layers. Each fully connected layer is followed by batch normalisation (BN1, BN2, BN3). It takes as input the A brief description of the components of the architecture is given below.

3.1.2 Data representation

The network takes the mixture spectrogram as the input and is trained to estimate a time-frequency mask, which when applied on the mixture spectrogram gives the estimated spectrogram of the clean target. Before feeding into the first fully connected layer, the input mixture spectrograms are cropped to 16kHz and normalised using mean and variance computed on the mixture spectrograms of the training dataset used (Input scaler). The output of the last layer is scaled using the mean and variance computed on the target spectrograms of the training dataset (Output Scaler).The window-size and hop-size used for computing the spectrograms are variable parameters (default=4096, 1024).

Bidirectional LSTM layer

LSTM layers are used to model temporal data. They are an improvement over regular recurrent neural networks (RNNs) in that they have special gates within their cells that allow selective retention and forgetting of past information [22]. They have what is known as a cell state at every given time instant that is a function of the input at that time as well as all the past inputs. In multi-layer LSTMs the n^{th} layer takes the output of the $(n-1)^{th}$ layer as its input. In bidirectional LSTMs the cell state at a given time is not just a function of the past inputs, but of the future inputs as well. This is useful for our problem, because at a given time the network is able to use information from the past as well as the future to perform source separation, which is better than using information just from the past as in a regular LSTM.

Batch Normalisation

The model also uses a batch normalisation layer after every fully connected layer to account for the internal covariate shift [23]. The layer normalises the output of each fully-connected layer to have a mean of 0 and and variance of 1. This has been shown to speed up convergence during training. The intuition behind why this happens is the following. If batch normalisation is not applied, then each time there is an update on the weights of the n^{th} layer, it changes the distribution of the input to the $(n+1)^{th}$ layer. This leads to training of layers happening in a sequential manner, from the early to the late layers. Applying batch normalisation before every layer ensures that the distribution of the input to each layer stays the same throughout the training, causing the layers to start learning simultaneously as opposed to sequentially, thus speeding up training.

Early Stopping

The model also employs Early stopping to prevent overfitting. The patience argument specifies the number of epochs (default=140) to wait after the last time validation loss improved, before stopping training. Using this we can training the model only up to that point till which performance on validation set is improving and stop training if it isn't.

Learning rate decay

The model also uses learning rate decay, which reduces the learning rate by a given factor if a particular metric (here, validation error) does not improve for a given number of epochs (default=80). It is found that learning benefits by reducing learning rate when learning stops improving.

3.1.3 Training Procedure

In each epoch of training the training data is arranged and fed into the network as follows. First, random audio chunks of length given by the variable seq-dur (default value=6s) are taken from the mixture and target (tabla or vocals) stem for each track of the training data. These are then arranged into batches of size given by the 'batch-size' variable. The spectrograms corresponding to these are computed on the fly.

After passing through FC1, tanh and BN1, each example in each batch is now fed into the LSTM framewise, ie. each timestep for the LSTM corresponds to one frame of the spectrogram. Since the LSTM used is bidirectional, the spectrogram is fed in both directions simultaneously. The input of the LSTM is concatenated with the output of the LSTM with a skip connection and is then passed through two fully connected layers, followed by an output scaler to give a time-frequency mask. This mask is multiplied elementwise to the input mixture spectrogram, to get the estimated target spectrogram.

After all the examples in a batch are fed in, the back propogation through time (BPTT) algorithm [22] is used to compute the gradients of the loss function (MSE between estimated and target spectrograms) with respect to the weights. These gradients are then averaged over the batch and weights are updated. This is done one after the other for all the batches. After this the next epoch begins for which random chunks are chosen again and the procedure is repeated.

3.1.4 Hyperparameters

The given network has a large number of hyperparameters which have different effects in the training of the network. In this section we describe some of the important hyperparameters and the effects that they have on the training.

1. **Batch size:** Size of the batches into which data is divided for training

    - A smaller batch size means the noise in gradient estimation is more. This could lead to more generalisable training, but at the same time given a fixed learning rate, convergence is slower.

- A larger batch size also means slower training, because it takes longer to estimate the gradient

2. **Spectrogram Window length:**

   - Broad band spectrograms are better at capturing formant information and transients like tabla onsets, while narrow band spectrograms better capture fundamental frequency of the sound. The performance of the network could vary based on the type of spectrogram used and it would be interesting to observe what window length works best. One intuition could be that for the problem of pitch detection on extracted vocals, using a narrow band spectrogram could give better results

3. **LR decay gamma:** Factor by which learning rate is reduced if validation loss isn't improving

4. **LR decay patience:** The number of epochs for which validation error must not improve after which learning rate decay is applied

5. **Sequence duration:** The length of the chunk taken from each song for one epoch of the training.

   - This directly relates to the amount of context you take into account while training the LSTM. Longer sequence duration means more context, but it makes it harder to train the LSTM .

6. **Loss function used for training:**

   - The *Open Unmix* architecture uses the MSE loss between the estimated and target spectrograms as the loss function for training.
   - In order to perform source separation that yields better to transcription after separation, adding a loss that corresponds to the transcription accuracy could help perform better separation in this context. We do this for the tabla source separation problem, by using a multi-task learning approach and introduce a new loss that corresponds to tabla onset detection accuracy.

3.2 Multi-task learning overview

Multi-task learning (MTL) refers to optimising a network using multiple loss functions, where each loss function could correspond to a different task. Learning shared represen-

tations for related tasks is found to help better generalise the performance on each task, compared to if they are learned independently [24]. There are two methods of using MTL for deep learning [24] (see figure 3.2).

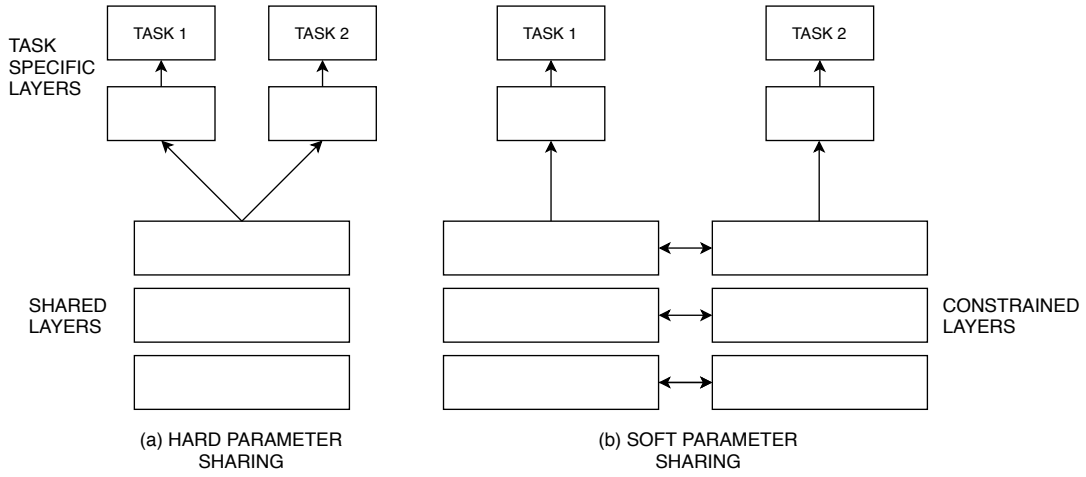


Figure 3.2: Two methods of MTL for deep learning

1. **Hard parameter sharing:** In this approach, the network architectures has hidden layers that are shared across the different tasks, followed by task specific output layers.

2. **Soft parameter sharing:** In this approach, different models with different parameters are used for different tasks. A regularisation term is used in the loss function during training to encourage the parameters of the different tasks to be similar.

Multi-task learning has been used in a variety of contexts in audio signal processing tasks. Sebastian *et. al.*,[25] use the multi-task learning approach on a temporal convolution network architecture for simultaneous tempo estimation and beat tracking. In the context of source separation, speaker source separation is found to improve when a DNN is trained to separate both target and interfering speakers as opposed to when the DNN only separates the target speaker from the mixture [26]. Clement *et. al.*, [27] proposes a model with a shared encoder and independent decoders for simultaneously separating multiple target instruments from music audio instead of using separate models trained independently for different instruments. Very recently, we came across a work that sought to simultaneously separate the instruments and obtain a MIDI transcription of

a polyphonic mixture using a multi-task trained network [28]. They use L2 distance to a MIDI based transcription as the transcription loss for training to eventually obtain improvements in both source separation and transcription.

We use the hard-parameter sharing multi-task learning approach to further improve tabla source separation. The details of our proposed MTL architecture are given in the next section.

3.3 MTL extension of Open Unmix for tabla source separation

In this work we aim to develop a source separation network for the separation of the tabla from a Hindustani vocals concert audio for the express purpose of tabla transcription in terms of stroke type and onset instants. We exploit multi-task learning to bring in a loss function that relates to the transcription task. Our proposed architecture thus has two branches which perform source separation and tabla onset detection tasks respectively (see figure 3.3).

The onset detection branch takes as its input, the output from an intermediate layer of the source separation network (as shown in figure 3.3). It consists of a convolutional layer followed by an average pooling layer and finally a fully connected layer with a sigmoid activation function, so the output can be interpreted as a probability. The convolutional layer uses 2D filters with unit height and a width of 7. This choice was made so that while detecting onsets, information from 3 neighboring frames on either side is available. The average pooling layer pools along the height, which corresponds to aggregating the energy in a given frame. The branch output is a vector of frame-wise onset probabilities.

The target for training the system is obtained by applying onset detection as described in section 2.2.2 on the corresponding target clean tabla spectrogram. The loss is computed as the binary cross entropy (BCE) between the target onset probabilities and the corresponding predicted frame-wise onset probabilities at the output of the MTL branch. We choose binary cross entropy loss because the onset detection can be viewed as a binary classification task for each frame.

3.3.1 Loss Functions

The network is trained using a linear combination of the MSE loss computed on the source separation branch and the BCE loss computed on the onset detection branch.

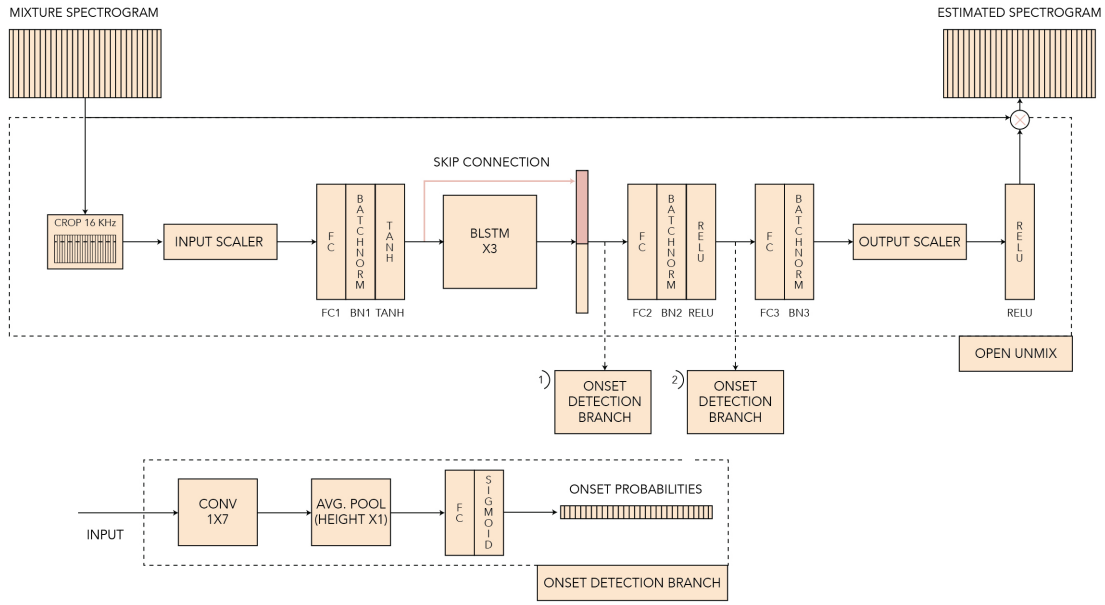


Figure 3.3: Proposed MTL architecture. 1) and 2) refer to the two branching positions (early, late) that we have considered while evaluating this multi-task architecture. The box on top contains the original *Open Unmix* [1] architecture. The box at the bottom shows the details of the tabla onset detection branch that we propose

On observing the loss values, we noticed that the scale of the onset loss was much smaller than the scale of the source separation loss. To counteract this, we scaled the onset loss by 10. The architecture is trained on the following loss expression.

$$L_{total} = (1 - \gamma)L_{SS} + 10\gamma L_{OD} \tag{3.1}$$

where, $\gamma \in [0, 1]$ is the hyperparameter to control relative weighting of the two losses, L_{SS} is the MSE loss corresponding to the source separation branch and L_{OD} is the BCE loss computed on the onset branch. It is computed as in equation 3.2.

$$L_{OD} = -\frac{1}{N}\sum_{n} w_n[p_n \log \hat{p}_n + (1 - p_n)\log(1 - \hat{p}_n)] \tag{3.2}$$

where, p_n and $\hat{p}_n$ are the target and predicted tabla onset probabilities corresponding to the n^{th} frame, N is the total number of frames and w_n is the weight assigned to the frame. $w_n = 0.5$ for frames on either side of target onsets and $w_n = 1$ for the rest. That is, frames on either side of target onsets are weighted half as compared to the rest of the frames, so as to account for the spread of onset events between frames by reducing penalty in the close vicinity.

3.4 Pytorch implementation

Open Unmix as well as our MTL extension of it are implemented in pytorch. The MTL architecture is built on top of the baseline architecture. In this section, we describe the functions of the different scripts that are used for the implementation, and how they interact with each other. More details about the implementation and the complete code can be found in this github repository..

- **train_mtl.py:** Contains script for training the proposed MTL architecture. On running, it saves the trained model weights, as well as the training and validation curves corresponding to the losses of interest. The script saves the model weights in a '.pth' file.
- **data.py:** Contains classes to arrange the dataset in a way that they can be given to the network as input.
- **model_mtl.py:** Contains classes of the model description of the architecture used.
- **test_new.py:** Contains scripts for using a trained model to perform separation and tabla onset detection using the onset detection branch of the MTL architecture.
- **eval_new.py** Uses specified trained model to separate and save separated tracks and the BSS evaluation measures for the entire test set. The separated tracks along with the accompaniment are saved as '.wav' files.
- **eval_onsets.py** Uses specified trained model to obtain and save tabla onset estimates using the onset branch of the MTL architecture, for the entire test set. The estimated framewise onset probabilities are saved in the form of a csv file.
- **od_1_eval.py:** Evaluates the onsets obtained from the onset branch of the MTL architecture.
- **od_2_eval.py:** Takes the separated tabla track as its input, performs the spectral flux onset detection algorithm on it, and evaluates the accuracy.
- **sc_eval.py:** Takes the separated tabla track as its input, performs the stroke classification algorithm on it, and evaluates the accuracy.

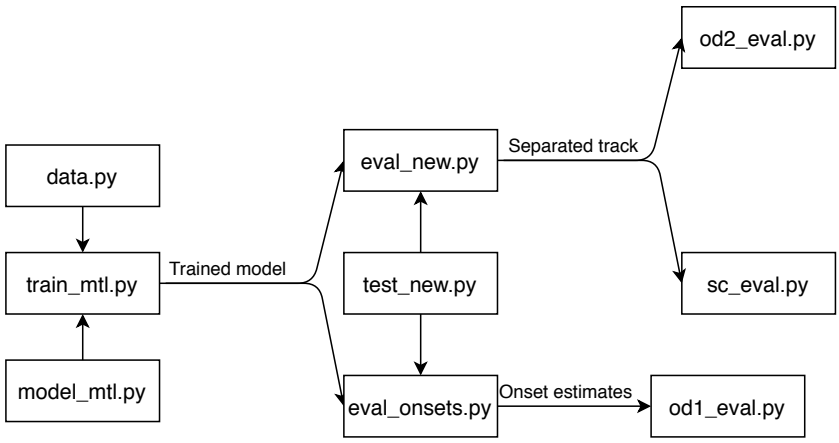


Figure 3.4: Python scripts used for implementation

Chapter 4

Dataset Description

Performance of source separation models is heavily reliant on the data that they have been trained on, leading to poor performance on music not represented in the training data. We aim to perform source separation on Hindustani music mixtures containing vocals, tanpura and tabla, an Indian percussive instrument that is significantly different in terms of its characteristics from any percussion instrument (usually drums) that appear in available source separation data sets. The tabla is a pair of drums with a structure and playing style that permits a rich variety of strokes with further control over their expressive characteristics [29, 11]. This motivated us to synthesise a dataset for training that is representative of the genre we wish to work with. In this chapter we describe the data that we have collected from different sources, and the procedures used to synthesise the source separation dataset using it.

4.1 Overview

As presented in table 4.1, the dataset consists of a total of 301 mixed audio tracks, sampled at 44.1kHz consisting of vocals, tanpura (drone) and tabla, of duration ∼17 hours. This is one of the typical ensembles, the others being inclusion of a second melodic instrument such as the harmonium. Each mixture is represented by its tabla audio, vocals plus tanpura audio and the full mix. We create a test dataset which is representative of realistic vocal concert mixtures. The train and validation datasets are created with an emphasis on efficient utilisation of the limited available solo instrument tracks, so as to get a large number of mixtures without resorting to excessive repetition.

4.2 Vocals plus tanpura tracks

We have 73 distinct vocal tracks spanning 5 different tonics, sung by over 20 different male and female singers, with durations ranging from 2 minutes to 6 minutes. Some of these are recordings of trained Hindustani vocalists that we obtained on request, while others are taken from Youtube. Some of the available recordings had a duration longer than 10 minutes. These were partitioned appropriately and saved as separate tracks. Some of the vocal tracks were made available along with tanpura backing, ie. there was no access to the solo vocal audio for these tracks. So to make the dataset uniform, for the others we mixed the vocals with tanpura corresponding to the tonic of the vocals at 10dB (higher energy vocals). The tanpura was recorded from the iTabla pro application for iOS, and looped over the duration of the vocals.

4.3 Tabla tracks

We have 257 tabla tracks obtained partly from the iTabla pro app, while some are recordings by trained tabla players. We also obtained a solo tabla performance off Youtube, which we de-noised (using inbuilt tools in Audacity), partitioned appropriately and saved as separate tracks. The recordings obtained from the iTabla app are single-cycles, while the others are complete compositions. The recordings span 6 different tonics and have durations ranging from 4 seconds (single cycle) to 3 minutes. The recordings range over greater than 13 tempo values.

Split	Tabla	Vocals+ Tanpura	Mixes	Duration (hours)
Train	215	51	247	15
Valid	36	16	48	1.50
Test	6	6	6	0.25
Total	257	73	301	16.75

Table 4.1: Distribution of component audio tracks and mixtures across dataset splits

4.4 Dataset Split and Mixing

Table 4.1 summarises the data set split into train, test and valid. It is ensured that there are no overlapping singers across the train, validation and test sets. The composition of

each split is described next.

4.4.1 Test set

The test set was designed to represent real world mixes. This meant making sure that the tabla in each track is coherent with the vocal track that it is mixed with, in terms of tonic, timing and style of playing. To obtain this, a trained tabla player was asked to listen to the vocal track over headphones and play the corresponding tabla accompaniment, which we recorded. These recordings were then manually aligned with the vocal+tanpura tracks and mixed at 10dB and 15dB (louder vocals). The levels were heuristically decided by comparison with typical available concert recordings.

4.4.2 Train and Validation sets

	Train-long	Train-short	Valid
Tabla	16 (×3)	199 (×1)	36 (×1/2)
Vocals+Tanpura	12 (×4)	39 (×5/6)	16 (×3)
Mixture	48	199	48

Table 4.2: Table shows the number of long and short tracks of each source present in the train set and number of tracks in the validation set. The × in the brackets indicate the number of mixes that each of the tracks from the corresponding set are used for. (×5/6) indicates that some of tracks are used in 5 mixtures, while others are used in 6. Similarly for (×1/2)

Unlike in the test set, the tabla and vocal+tanpura tracks used to create the train and validation sets are unrelated to each other, ie. they are not necessarily coherent. For both sets, the mixtures are made by looping each tabla track over the duration of the corresponding vocal+tanpura track. Half the mixtures are made at 10dB and the other half at 15dB.

For the train and validation sets, mixing is done such that we utilise the available data to the fullest, while also preventing more than warranted repetition. We first divide the tabla and vocal tracks into two sets each, long($\geq$ 2 minutes and $\geq$ 6 minutes respectively) and short. Some tracks from the short set are set aside to make the validation set. A total of 247 train set mixtures are made by mixing long and short vocals+tanpura tracks

with long and short tabla tracks respectively. A total of 36 validation mixtures are made by mixing the tabla and vocal tracks that were set aside, while ensuring that each track was utilised in a fixed maximum number of mixtures. The number of repetitions and mixtures made in the train data and validation data are detailed in table 4.2.

To create even larger training and validation sets, a number of possible augmentations can be applied on the data. This includes, pith-shifting and tempo-shifting the solo instruments before mixing them. Another way could be to loop multiple shifted versions of a given tabla track over the corresponding vocals+tanpura track, so that one pair of vocals+tanpura and tabla tracks yield more than one mixture. These multiple mixtures would be such that the relative position of the two tracks is different in each mixture. We have not used any of these augmentation techniques in the creation of our dataset. However, future work could employ them to create an even more expansive dataset.

4.5 Dataset preparation scripts

The mixing and arranging of the dataset to be used for the source separation task has been implemented using python. These scripts can be re-used to re-mix the dataset, as and when more data is made available. In this section we will provide a brief overview of the different scripts that are used.

- **make_train_dataset.py**, **make_valid_dataset.py**, **make_test_dataset.py:** Scripts used to prepare the train, validation test sets for the source separation task. The scripts save each mixture in a folder containing the mixture audio, vocals+tanpura audio and the tabla audio.

  The mixing is implemented by first looping the tabla track over the duration of the vocals+tanpura track, followed by normalising their energies, summing the two tracks based on the mixing level (10dB or 15dB) and scaling the obtained mixture to have a fixed energy per sample. It is described by the following equation.

$$u[n] = w\frac{v[n]}{\sqrt{\sum_i v^2[i]}} + \frac{t[n]}{\sqrt{\sum_i t^2[i]}} \tag{4.1}$$

$$m[n] = kN\frac{u[n]}{\sqrt{\sum_i u^2[i]}} \tag{4.2}$$

  where, m[n], v[n] and t[n] correspond to the mixture, vocals+tanpura and mixture tracks respectively. w is the weighting parameter which is a function of the mixing

level (10dB or 15dB), N is the total number of samples, and k is the average per sample energy for the created mixtures (we use k=0.005)

- **mix_tanpura.ipynb:** This notebook has codes to mix the vocal tracks which are available without tanpura, with the tanpura track of the corresponding key. The mixing in this script is also implemented in a similar way as the dataset creation scripts.

More details about the dataset and the scripts used for mixing can be found in this github repository.

Chapter 5

Tabla source separation experiments

In this chapter, we evaluate the tabla source separation performance of the architecture(s) described in chapter 3, trained on the dataset described in chapter 4. We evaluate the separation performance based on the metrics described in section 2.2. This chapter is organised as follows. In the first section, we motivate the need for development of an improved tabla separation model. The next section describes the different models we trained and the experiments we performed. In the last section, we present the results, observations from each of the performed experiments. [1]

5.1 Motivation

Table 5.1 shows the performance of the tabla transcription evaluation algorithms on clean tabla, mixture audios of the test set, as well as tabla estimates obtained by passing the test set mixtures through the *Open Unmix* architecture trained on the drums target using the MUSDB18 [19] dataset. We note how poor the transcription scores are when applied on the mixture audio as well as when applied on the outputs of an available percussion separation model. We also see that while the *Open Unmix* drums model is able to slightly improve the onset detection performance in terms of the f-score, it highly degrades the performance of the stroke classification. This can be explained by the differences in the spectral characteristics of drums and tabla. While both have sharp onset bursts, the tabla also has some low frequency harmonic components which vary

[1] This chapter is largely based on our submission to ISMIR 2020 on "Audio source separation for improved tabla transcription in vocal mixtures"

based on the stroke type. A model trained to separate drums is to some extent able to separate the wide-band energy bursts corresponding to table onset, but it completely fails to preserve the low-frequency harmonic components of the tabla. This is reflected in the poor stroke classification performance.

The performance on the clean tabla is an upper bound that we wish to get closer to via improved source separation. This sets up the need to develop a tabla source separation model that allow us to improve tabla transcription accuracy when provided with only a vocals mixture of Hindustani music.

	P	R	F	SC (%)
Clean tabla	0.99	0.92	0.95	74.90
10dB mix	0.67	0.70	0.68	51.05
15dB mix	0.54	0.65	0.59	45.85
OU-drums (10dB)	0.637	0.809	0.713	40.03
OU-drums (15dB)	0.500	0.814	0.619	41.26

Table 5.1: Onset detection and 4-way stroke classification performance on clean tabla, mixture audio and mixture audios passed through *Open Unmix* trained on drums target using the MUSDB18 [19] dataset. P, R and F refer to precision, recall and f-score respectively. SC (%) refers to percentage stroke classification accuracy

5.2 Models evaluated

All the models evaluated in this section are trained with a batch size of 16, using an Adam optimiser with an initial learning rate of 10^{-3}. Spectrograms with a window size of 92ms and a hop-size of 23ms are used. Every training epoch uses sequences of 6s duration. We use a learning rate decay of 0.3 and patience of 80 epochs. The models are trained up to a maximum of 1000 epochs, however early stopping with a patience of 140 epochs is used to prevent overfitting. The training procedure is borrowed from the original *Open Unmix* [1] implementation, and is as described in section 3.1.3.

We perform a total of 7 experiments based on varying choices of γ, position of introduction of the onset branch, training procedure and loss function choices in order to get insights on their effect on the problem of effective tabla source separation for improved transcription. These experiments are described below. The details of the models trained are summarised in Table 5.2.

Experiment 1 : We first train the *Open Unmix* architecture on our dataset and evalu-

ate its performance, which serves as the baseline. This is equivalent to training our MTL architecture using the loss from equation 3.1 with $\gamma = 0$. The MSE loss is computed over the entire frequency range (0-22.05kHz).

Experiment 2 : We then introduce the onset detection branch to this architecture. However, before we train the architecture for both the tasks simultaneously, we want to verify that the introduced OD branch is performing as expected and learning the tabla onsets. For this reason, we train our MTL architecture using $\gamma = 1$, so that only the onset branch of the architecture is trained, and evaluate the OD performance of the branch output.

Experiment 3 : The aim of this experiment is to decide on the exact training procedure to be used while training the MTL architecture. We train models by first pre-training the source separation branch ($\gamma = 0$) up to a varying number of epochs (0, 75, 100, 125 and complete training), followed by which we introduce the OD branch loss in the training. In this experiment we use $\gamma = 0.8$ for the second part of the training, and introduce the OD branch in the early position as described in figure 3.3.

Experiment 4 : We introduce the onset detection branch to this architecture in the late position as described in figure 3.3 and train it using losses from both the branches with $\gamma = 0.8$, right from the start. This is done based on the outcome of the previous experiment.

Experiment 5 : Next, we train only the *Open Unmix* architecture ($\gamma = 0$) using MSE loss computed only between the bins of the spectrograms corresponding to frequencies lesser than 4kHz. This modification was motivated in part by the observed outcomes of experiments 1 and 4, as explained further in the next section.

Experiments 6 & 7 : The onset detection branch (ODB) is now introduced to this architecture where the source separation MSE is computed only for bins up to 4kHz. We experiment with two possible positions of branching from the source separation architecture, which are before FC2 (early) and before FC3 (late), ie. the second and third fully connected layers of *Open Unmix*. Each of these model architectures are trained using a range of values of loss weighting (γ) as seen in table 5.2. Based on the outcomes of experiment 3 which are described in the next section, we train both the branches of the MTL architecture from the start, without pre-training any of the branches individually first.

	Network Type	MSE BL	ODB position	γ	Pretraining
1	OU	22.05kHz	-	0.0	-
2a	OU+ODB	22.05kHz	Early	1.0	-
2b	OU+ODB	22.05kHz	Late	1.0	-
3a	OU+ODB	22.05kHz	Early	0.8	0
3b	OU+ODB	22.05kHz	Early	0.8	75
3c	OU+ODB	22.05kHz	Early	0.8	100
3d	OU+ODB	22.05kHz	Early	0.8	125
4	OU+ODB	22.05kHz	Late	0.8	0
5	OU	4kHz	-	0.0	-
6a	OU+ODB	4kHz	Early	0.8	0
6b	OU+ODB	4kHz	Early	0.6	0
6c	OU+ODB	4kHz	Early	0.4	0
6d	OU+ODB	4kHz	Early	0.2	0
7a	OU+ODB	4kHz	Late	0.8	0
7b	OU+ODB	4kHz	Late	0.6	0
7c	OU+ODB	4kHz	Late	0.4	0
7d	OU+ODB	4kHz	Late	0.2	0

Table 5.2: Models trained and evaluated. Except for model 1, all are trained on our dataset described in chapter 4 .OU refers to the *Open Unmix* architecture. OU+ODB refers to *Open Unmix* along with the onset detection branch. ODB position early and late refer to onset branch being introduced before the FC2 layer and before the FC3 layer. MSE BL is the frequency up to which we compute the MSE loss on the spectrograms. γ decides the relative weighting of the losses. A higher γ corresponds to more weight to the onset branch loss. Pretraining refers to the number of epochs upto which the source separation branch was trained before introducing the ODB

5.3 Results and observations

Tables 5.3 and 5.4 contain the evaluation metrics for the outputs of the source separation branch and onset detection branch respectively of the trained models . The salient observation for the performed experiments are presented below.

Experiment 1 : Separated tabla from the *Open Unmix* architecture trained on our dataset (model 1) gives significantly improved onset detection as well as stroke classification performance as compared to that obtained on the mixtures as well as that obtained

	10dB mix					15dB mix				
Model	P	R	F	SC (%)	SDR	P	R	F	SC (%)	SDR
1	0.892	0.770	0.827	64.31	1.905	0.845	0.731	0.784	59.99	1.035
2a	-	-	-	-	-	-	-	-	-	-
2b	-	-	-	-	-	-	-	-	-	-
3a	0.879	0.772	0.822	62.67	1.878	0.842	0.751	0.794	58.93	1.035
3b	0.880	0.779	0.827	64.40	1.912	0.840	0.756	0.796	60.67	1.037
3c	0.874	0.783	0.826	62.98	1.863	0.827	0.754	0.789	59.38	1.026
3d	0.876	0.778	0.824	63.51	1.889	0.835	0.756	0.794	59.37	1.025
4	0.872	0.774	0.820	62.13	1.883	0.834	0.760	0.796	59.38	1.038
5	0.933	0.848	0.889	63.75	1.915	0.846	0.802	0.824	58.46	1.041
6a	0.960	0.848	0.901	61.35	1.856	0.902	0.820	0.859	59.30	1.035
6b	0.939	0.851	0.893	64.96	1.924	0.886	0.792	0.836	60.16	1.046
6c	0.952	0.854	0.900	63.11	1.904	0.892	0.816	0.852	59.32	1.055
6d	0.938	0.845	0.889	63.73	1.906	0.876	0.799	0.836	60.17	1.052
7a	**0.959**	**0.842**	**0.896**	**65.18**	**1.900**	**0.896**	**0.818**	**0.856**	**60.94**	**1.023**
7b	0.953	0.859	0.904	63.77	1.903	0.891	0.825	0.857	59.33	1.056
7c	0.935	0.857	0.894	64.48	1.9045	0.863	0.816	0.839	58.23	1.054
7d	0.939	0.860	0.898	64.11	1.859	0.858	0.832	0.845	57.80	1.031

Table 5.3: Performance of the source separation branch of the trained models in terms of the metrics described in 2. P, R and F refer to precision, recall and f-score for onset detection respectively. SC (%) is the percentage stroke classification accuracy. SDR is the source-to-distortion ratio.

on the separated tabla from the drums target *Open Unmix* architecture trained on the MUSDB18 [4] dataset as shown in table 5.1. This experiment highlights the effectiveness of our genre specific dataset in improving tabla separation performance.

Figure 5.1 (left) shows the validation MSE plot obtained during the training of model 1. The clear decreasing trend indicates that the train dataset we have synthesised is large enough to train the model to generalise well and improve the MSE between clean and estimated tabla over the independent validation set as well.

Figure 5.1 (right) shows the plot of the validation BCE between the target onsets (obtained by applying the OD algorithm on clean tabla) and the novelty curve estimated by applying spectral flux on the tabla estimate during the course of the training. Henceforth, we will refer to this as the BCE SF loss. The decreasing trend of this plot

	10dB mix			15dB mix		
Model	P	R	F	P	R	F
1	-	-	-	-	-	-
2a	0.956	0.867	0.909	0.932	0.826	0.876
2b	0.949	0.861	0.903	0.918	0.813	0.862
3a	0.944	0.879	0.910	0.909	0.845	0.876
3b	0.944	0.879	0.910	0.909	0.845	0.876
3c	0.938	0.852	0.893	0.895	0.812	0.852
3d	0.940	0.877	0.907	0.911	0.851	0.880
4	0.934	0.867	0.899	0.908	0.821	0.862
5	-	-	-	-	-	-
6a	0.950	0.860	0.900	0.916	0.814	0.862
6b	0.905	0.822	0.861	0.879	0.771	0.822
6c	0.948	0.869	0.906	0.903	0.820	0.860
6d	0.940	0.844	0.889	0.906	0.784	0.840
7a	0.960	0.860	0.910	0.917	0.821	0.866
7b	0.949	0.862	0.903	0.910	0.820	0.862
7c	0.939	0.86	0.898	0.891	0.799	0.842
7d	0.939	0.854	0.895	0.886	0.822	0.853

Table 5.4: Onset detection performance of the ODB of the trained models. P, R and F refer to precision, recall and f-score for onset detection respectively.

indicates that training *Open Unmix* on the MSE target also leads to the model being trained in such a way that the output tabla gives increasingly better onset detection. This kind of plot was particularly useful for us in the later experiments, as it was a quick way of gauging how much source separation has improved in terms of giving improved onset detection, without having to formally test the performance on the test set.

Experiment 2 : Figure 5.2 shows the training and validation plots of the BCE loss of the ODB for model 2a (similar observation for model 2b). The decreasing trend as well as the significantly small final value of loss in both the training and the validation plots is an indication that the ODB is working as expected and is learning to detect tabla onsets.

This can further be confirmed by observing the accuracy of the onsets predicted by the ODB of both models 2a and 2b on the test set (see table 5.4). Both model predictions give F-scores of around 0.9 on 10db mixtures and 0.x on 15dB mixtures, indicating

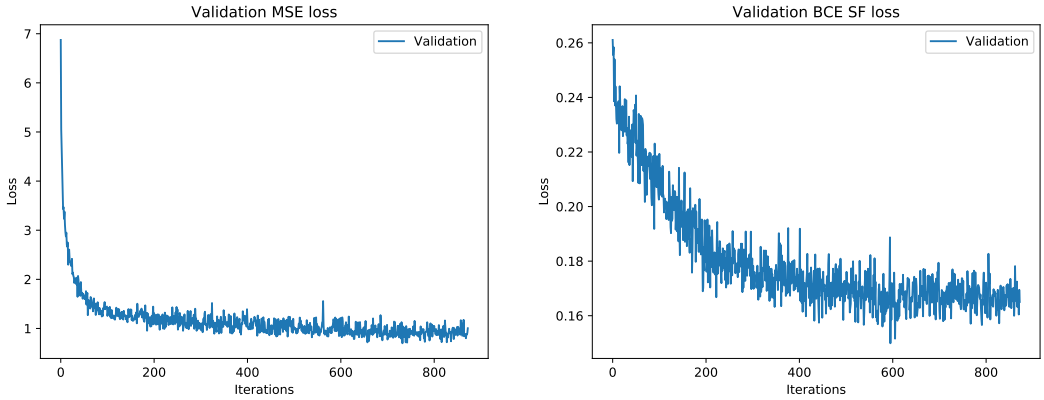


Figure 5.1: Validation error plots during model 1 training. From left to right: (a) Validation MSE (b) Validation BCE SF

that the branch has learnt to detect tabla onsets reasonably well.

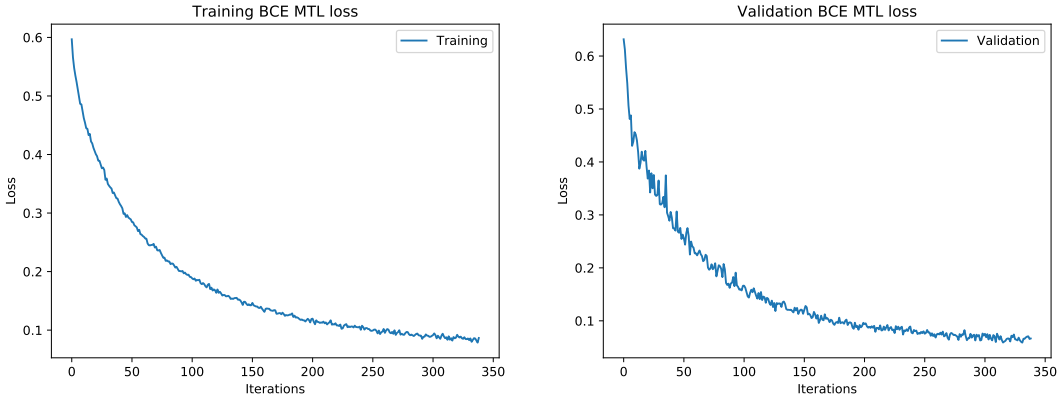


Figure 5.2: From left to right: (a) Train MSE loss plot for model 1 (b) Validation BCE SF loss

Experiments 3 & 4 : Observing the results of experiment 3, ie. models 3a-3d in table 5.2 we can see that there are minor differences in the onset detection and stroke classification performance of the different models, however there is no significant difference that would compel us to choose one training procedure over the other. Hence, for all the following experiments we choose to train both the branches simultaneously without doing any pre-training of the source separation branch.

Model 4 is the same as model 3a (both trained without any pre-training of the source separation branch), the only difference being the position of introduction of the ODB.

Comparing the performance of models 3 and 4 with model 1 (see table 5.3) we note

that adding an onset branch to the *Open Unmix* architecture has a negligible impact on onset detection performance, while stroke classification performance either becomes slightly worse or stays nearly the same. These experiments however helped us obtain useful insights via observations on the different spectrograms. As expected, the predicted tabla spectrograms deviate from the corresponding target spectrograms in different ways depending on the architecture, and in particular, the influence (position, loss weighting) of the introduced onset detection branch.

Observing the spectrograms in figure 5.3, we can see that the output obtained from model 4 is significantly distorted with vocals spectra above 5kHz, as compared to the output from model 1. This, along with the observation that training MSE loss finally settles at similar values for model 1 and 4 (see figure 5.4) indicates that bringing in the onset branch loss changes the distribution of the mean squared error across the time-frequency regions. The errors are worse at the higher frequencies, probably at the cost of better matching in lower frequency regions. This significant distortion however is not reflected in the onset detection and stroke detection performance which degrade only slightly. This, along with the understanding of tabla acoustics led to the insight that onset detection is large reliant on the frequency spectrum up to roughly 4kHz. This motivated us to restrict MSE loss computation to the 0 - 4kHz region in the ensuing experiments.

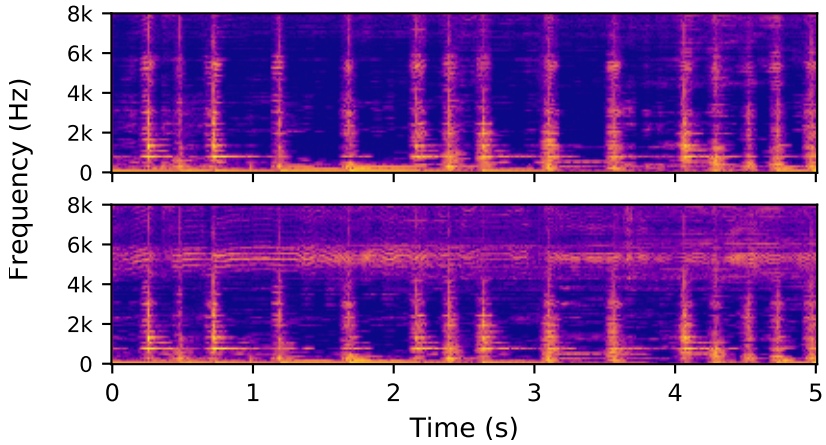


Figure 5.3: Spectrograms of tabla outputs (0-8kHz shown) predicted by model 1 (Top) and model 4 (Bottom). The spectrograms correspond to an excerpt from a 10dB test set example

Experiment 5 : Training *Open Unmix* on MSE loss restricted up to 4kHz (model 3) significantly improves onset detection performance over the case where MSE uses the entire frequency range. The stroke classification performance in this case is degraded.

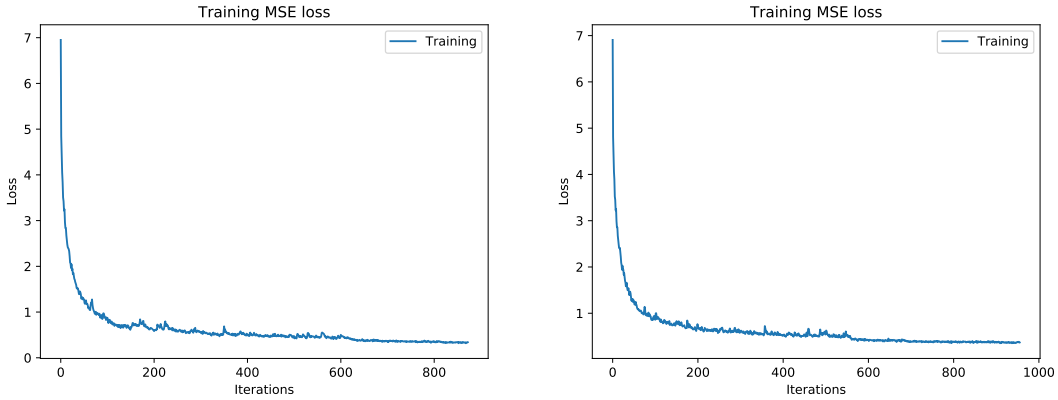


Figure 5.4: (From left to right) Training MSE plots for (a) Model 1 (b) Model 4

Experiments 6 & 7 : Introducing the onset branch now (models 6,7) further improves onset detection performance. This improvement is even more pronounced in the case of 15dB test mixtures. Observing the values in table 5.3 we can see that the precision is what improves as compared to model 5. Recall stays nearly the same with the introduction of the onset branch. Stroke classification performance also shows improvement over the baseline case, for some of the MTL models (especially model 7a). This indicates that introducing the onset branch and loss to the source separation architecture leads to source separation that not only better filters out onsets which are not from the target source (indicated by higher precision), but also better preserves spectral content of the strokes to which the target onsets are associated. This can be confirmed by observing the output spectrograms of model 5 and 7a in figure 5.5 in comparison with the corresponding target and mixture spectrograms. Model 7a not only emphasizes onset frames more clearly, but also suppresses spectral artifacts that could be mistaken for a tabla onset. Model 7a on the other hand has more artifacts than Model 5 in regions that do not matter in onset detection computations, like continuous patches of vocal harmonics. At point (a) in the figure, which is an onset, we can see that model 7a performs cleaner separation as compared to model 5. Point (b) which has no onset is more distorted in the model 7a output as compared to that of model 5. At point (c) there is a vertical spread in model 5 output which resembles an onset. This is eliminated by model 7a. At point (d) an onset which is weakly detected by model 5, is much stronger in model 7a.

Observing the validation BCE SF loss curve in this case also gives an indication that this approach improves source separation in terms of better tabla onset detection. Figure 5.6 shows the Validation BCE SF plot for model 7a. Comparing it with the same plot for model 1 (see figure 5.1 (right)) we can see that the loss settles at a lower value

for model 7a as compared to model 1.

The position of introduction of the onset branch and the loss weighting (γ) impact onset detection as well as stroke classification performance, but there is no discernible pattern to be able to to predict what position or which γ is likely to give better performance. We note that model 7a gives improved performance over both the tasks.

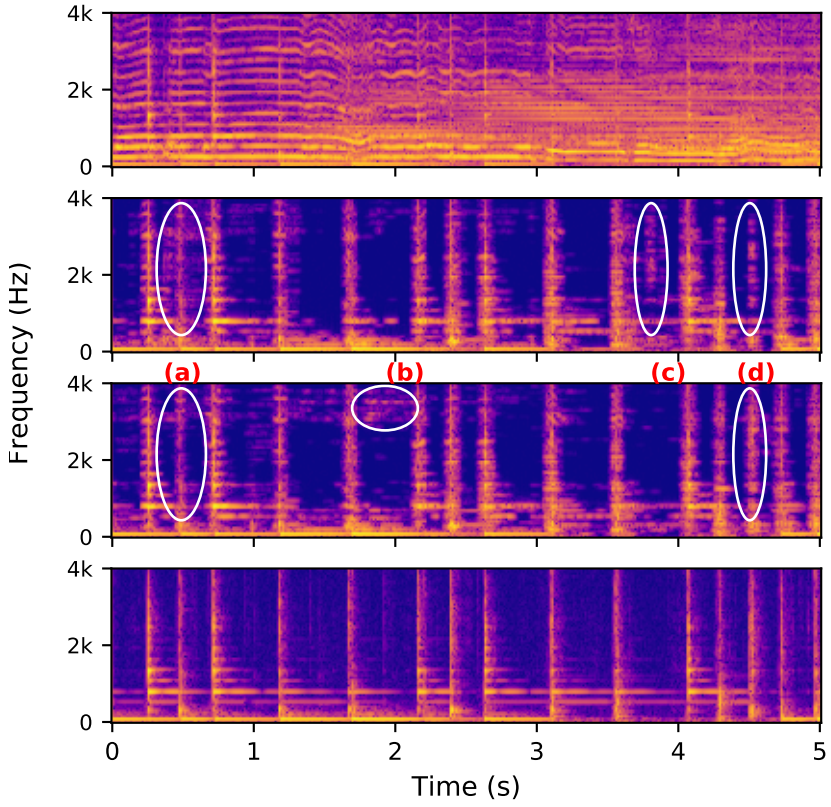


Figure 5.5: From top to bottom: Spectrograms of (i) Mixture (ii) Model 5 output (iii) Model 7a output and (iv) Clean tabla. At time (a) and (d) model 7a output has a more pronounced onset. (b) Model 7a has worse separation in some non-onset frames (c) Model 5 output has vertical spread which could be mistaken as tabla onset, model 7a has filtered this out

5.3.1 Additional comments

We also evaluated the onsets detected by the onset branch. Table 5.4 shows the accuracy of the onsets detected by the ODB . The accuracy was found to be comparable to that of the onsets obtained from the source separated tabla, and in some cases slightly better. Model 3a seems gives the best onset detection performance from the ODB.

We also report the source-to-distortion values for the tabla separation performed on the test set. Its interesting to note that introducing the additional onset loss does not

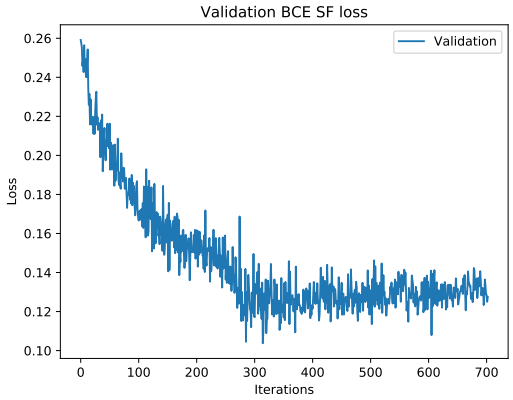


Figure 5.6: Model 7a validation BCE SF loss

cause source separation to perform worse in terms of source separation quality metrics, but it even slightly improves those metrics for some cases.

Chapter 6

Vocal source separation experiments

In this chapter we detail the vocal source separation experiments performed. The source separation performance is evaluated using the metrics described in section 2.2. While evaluating the performance in terms of a given pitch tracking algorithm, we use the pitch track obtained by applying that algorithm on the clean vocals as the ground truth pitch estimate.

Ideally, to get good vocal separation on our test set, we would like to train the *Open Unmix* architecture on a vocals only target using a Hindustani music dataset. Unfortunately, the data obtained by us was such that a significant part of the vocals already had tanpura backing. Hence, in order to make use of all the available data, the dataset was constructed such that there are two stems available ; clean tabla and vocals+tanpura. Due to this, we were unable to train the architecture on a vocals only target. Based on this, we experiment with two approaches of obtaining the vocals estimate, which are described in this chapter. The chapter starts with motivating the need for an improved vocal source separation model, followed by sections describing the two approaches we used.

6.1 Motivation

Table 6.1 shows the performance of both the pitch tracking algorithms applied on mixture audios of the test set as well as vocal estimates obtained by passing the test set mixtures through the Open Unmix architecture trained on the vocals target using the MUSDB18 [4] dataset. We can see that CREPE overall performs significantly better

than PYIN, because of its robustness to noise [15]. Using the pretarained *Open Unmix* model for vocals improves the performance of the PYIN and CREPE algorithms as compared to the mixture, although even these performances are objectively unsatisfactory. This sets up the need for a vocal source separation system trained specifically on Hindustani music mixtures that is able to separate vocals in such a way that pitch detection algorithms applied on these separated vocals give improved performance.

	PYIN				
	VR	FA	RPA	RCA	OA
10dB mix	0.849	0.862	0.208	0.301	0.161
15dB mix	0.834	0.847	0.334	0.425	0.196
OU-vocals (10dB)	0.837	0.780	0.611	0.630	0.401
OU-vocals (15dB)	0.834	0.809	0.595	0.617	0.296
	CREPE				
	VR	FA	RPA	RCA	OA
10dB mix	0.911	0.408	0.877	0.904	0.819
15dB mix	0.926	0.433	0.886	0.908	0.830
OU-vocals (10dB)	0.931	0.431	0.880	0.901	0.836
OU-vocals (15dB)	0.934	0.430	0.882	0.903	0.839

Table 6.1: Vocal pitch detection performance using PYIN and CREPE algorithms on mixture audio and mixture audios passed through *Open Unmix* trained on vocals target using the MUSDB18 [19] dataset. VR, FA, RPA, RCA and OA refer to voicing recall, false alarm, raw pitch accuracy, raw chroma accuracy and overall accuracy respectively.

6.2 Train *Open Unmix* on vocals+tanpura target

Our first approach was to train the *Open Unmix* architecture on the vocals+tanpura target, and use this model to obtain an estimate of the vocals+tanpura. After this the tanpura from this would be suppressed using the Noise reduction tool on Audacity.

However, the architecture was unable to train successfully on this target. This was first observed by hearing the 'vocals+tanpura' estimated by the trained model. It can further be confirmed by comparing the validation MSE plot for training on this target, with that of the tabla target (which we know has trained successfully). There are 2 main differences between the two plots (figure 6.1). First, the plot for the tabla target has a slight decreasing trend all throughout, while that of the vocals+tanpura target decreases

once at the start, but only oscillates after that, without any clear trend. Second, is the number of epochs for which the training happens. Training on the tabla target goes on for nearly 800 epochs, while that for the vocals+tanpura target goes on only for about 170 epochs. Given that we have used early stopping (patience=140) during training, training goes on only upto such a point where validation loss is improving. The tabla target training going on for 800 epochs indicates that for this duration of training the separation performance is improving on the independent validation set. For the vocals+tanpura target training stops only at 170 epochs. This along with the observation that the training MSE loss for this case is decreasing indicates that minimizing MSE on the train set is not able to improve MSE on the validation set. This leads us to the conclusion that the architecture does not train well for a combination of audio from two instruments, as it does for audio from only one instrument, and we try the next approach.

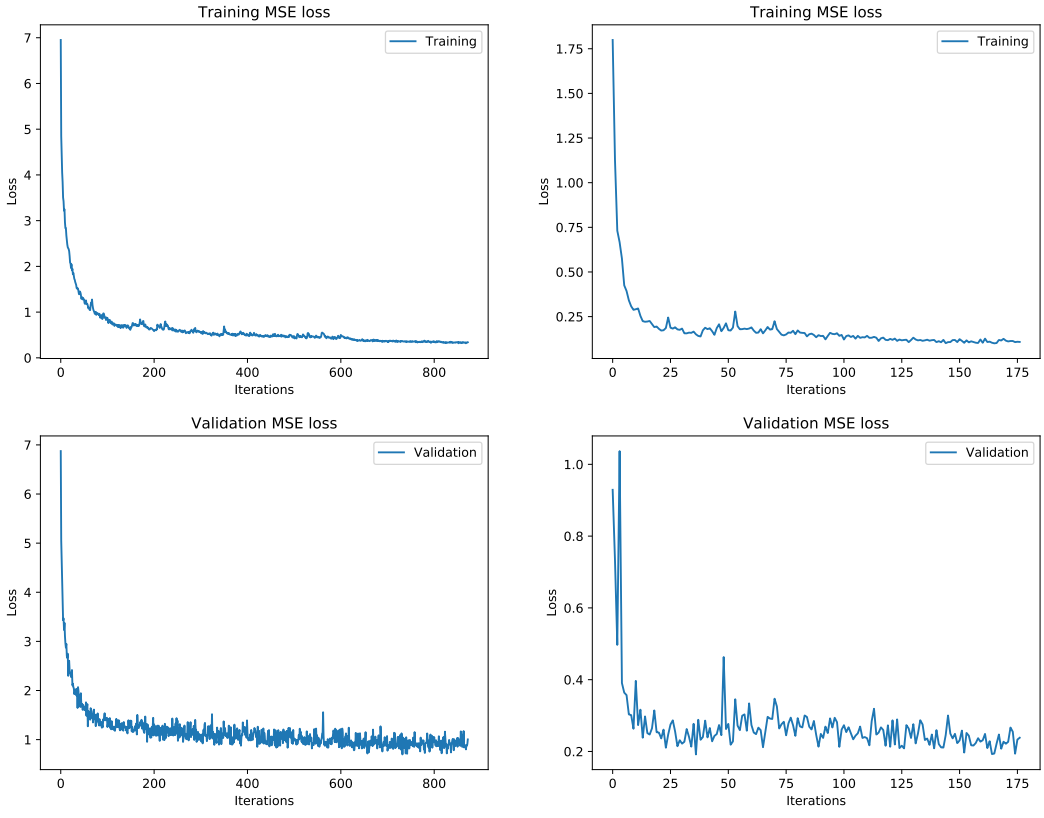


Figure 6.1: Training and validation MSE plots for *Open Unmix* trained on tabla (left) and vocals+tanpura (right) targets

6.3 Remove tabla estimate from mixture audio using Wiener filtering

In this approach, we obtain an estimate of the vocals+tanpura by 'subtracting' the tabla estimate obtained using tabla separation model 1 (see table 5.2) from the mixture using Wiener filtering. We then suppress the tanpura using the Noise Reduction tool on Audacity, to get the vocals estimate. The tool requires the user to first manually choose a section of the song that has only tanpura audio. Then, assuming the tanpura to be stationary, it suppresses the part of the spectrum that it detects to be corresponding to the tanpura, based on information from the earlier chosen section. The tool allows us to perform this suppression at various dB levels, where the level indicates the amount of volume reduction of the chosen noise to be applied. A higher level leads to more noise suppression, but could come at the cost of damaging the signal of interest.

Table 6.2 shows the performance of the pitch tracking algorithms applied on the vocal+tanpura estimates obtained by filtering out the tabla and the vocal estimates obtained by suppressing the tanpura in these mixtures at different levels of suppression. We make the following observations from these experiments.

For PYIN, the performance on the Mix-Tabla (M-T) estimate is an improvement over the mixture. The 20dB suppression estimate shows no significant change in performance as compared to the M-T, but the 30dB suppression estimate shows improved overall accuracy. This improvement in accuracy comes mainly from a lower false alarm rate. The raw pitch accuracy has in fact slightly degraded as compared to M-T. Figure 6.2 shows an excerpt of the PYIN pitch tracks of a test example. At around the 8s mark ((i) in figure 6.2) we can see that the clean vocals are unvoiced, but the M-T estimate detects that entire section as being voiced, because of the background tanpura drone. For the (M-T)30dB estimate however, we see that some part of that section is now detected as unvoiced. This kind of occurrence is common throughout the test examples, where the vocalist takes a pause between lines. In the M-T estimate, even though the vocalist has stopped singing, the harmonicity of the tanpura causes the corresponding frames to be detected as voiced. Suppressing the tanpura using the available tool leads to some of these frames being correctly detected as unvoiced. The raw pitch accuracy however does not noticeably benefit from the suppression. Points (ii) and (iv) on the plots show regions here M-T follows the clean vocal pitch track better than (M-T)30dB. Point (iii) shows a section where (M-T)30dB performs better than M-T. Upon closer observation of section (iv), we can see that in this section the vocals pitch is nearly the same as that of

	PYIN									
	10dB mix					15dB mix				
Estimate	VR	FA	RPA	RCA	OA	VR	FA	RPA	RCA	OA
Mix	0.849	0.862	0.208	0.301	0.161	0.834	0.847	0.334	0.425	0.196
M-T	0.835	0.818	0.592	0.618	0.376	0.832	0.828	0.583	0.613	0.280
(M-T)20dB	0.828	0.802	0.579	0.592	0.373	0.832	0.819	0.558	0.577	0.279
(M-T)30dB	0.827	0.589	0.540	0.548	0.432	0.819	0.692	0.527	0.546	0.350
	CREPE									
	10dB mix					15dB mix				
Estimate	VR	FA	RPA	RCA	OA	VR	FA	RPA	RCA	OA
Mix	0.911	0.408	0.877	0.904	0.819	0.926	0.433	0.886	0.908	0.830
M-T	0.928	0.480	0.883	0.906	0.827	0.932	0.478	0.886	0.909	0.831
(M-T)20dB	0.922	0.478	0.873	0.903	0.816	0.925	0.486	0.875	0.903	0.818
(M-T)30dB	0.912	0.489	0.852	0.891	0.790	0.914	0.499	0.854	0.892	0.791

Table 6.2: Pitch detection performance of PYIN and CREPE algorithms when applied on different vocal estimates. M-T refers to Mix-Tabla which is the estimate obtained by wiener filtering the tabla estimate from the mixture. (M-T)xdB refers to the vocal estimate obtained by applying a tanpura suppression of xdB on the M-T estimate.

the tanpura drone (see pitch detected in section (i) by Mixture-Tabla, where the vocalist is slient). For this region it can be hypothesised that the suppression also suppresses vocal harmonics, leading to poor pitch tracking. On observing the spectrograms (figure 6.4) and listening to the audio at instants (ii) and (iii), it is difficult to identify why one estimate performs better over the other in these regions. Average statistics indicate there is not a significant difference between M-T and (M-T)30dB in terms of raw pitch accuracy, but suppression does slightly damages it.

Since CREPE is more robust to noise it still gives a reasonably good estimate in the presence of other sources. We can see in figure 6.3 that for all three cases shown, the pitch track follows the clean vocals pitch track quite well. The performance of M-T is a slight improvement over the mixture. Suppressing the tanpura worsens overall accuracy. We hypothesise that the reason for this is the following. The CREPE algorithm because of its robustness to noise, is able to ignore voicing due to the tanpura leading to a significantly low false alarm rate even on the mixture. Hence, suppressing tanpura does not further help with that. Raw pitch accuracy gets degraded slightly on suppression, causing the overall accuracy to go down. This could be happening as a result of the vocal

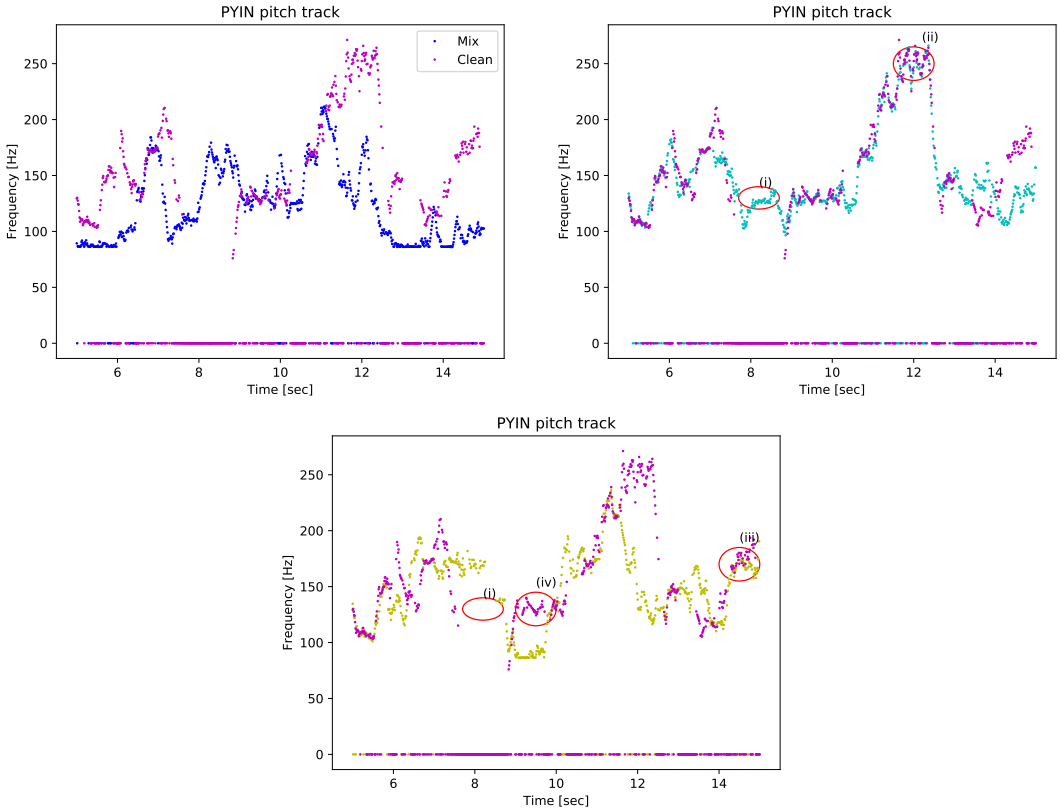


Figure 6.2: PYIN pitch tracks of an excerpt from a test example of (a) Mixture (b) Mixture-Tabla (c) (M-T)30dB compared with pitch track of clean vocals. (M-T)30dB refers to the track obtained by suppressing the tanpura of the Mixture-Tabla track at 30dB level.

harmonics being damaged during the tanpura suppression. An attempt was made to verify this from the spectrogram (figure 6.4) and audio, but because of the slight amount of degradation in performance, it was difficult to conclusively find a region where this could be observed.

From these experiments we conclude that the available tanpura suppression is not sophisticated enough to obtain a vocals estimate that gives good pitch detection performance. Future work on this problem should focus on restructuring the dataset to have a vocals only stem, and to train the *Open Unmix* architecture using it.

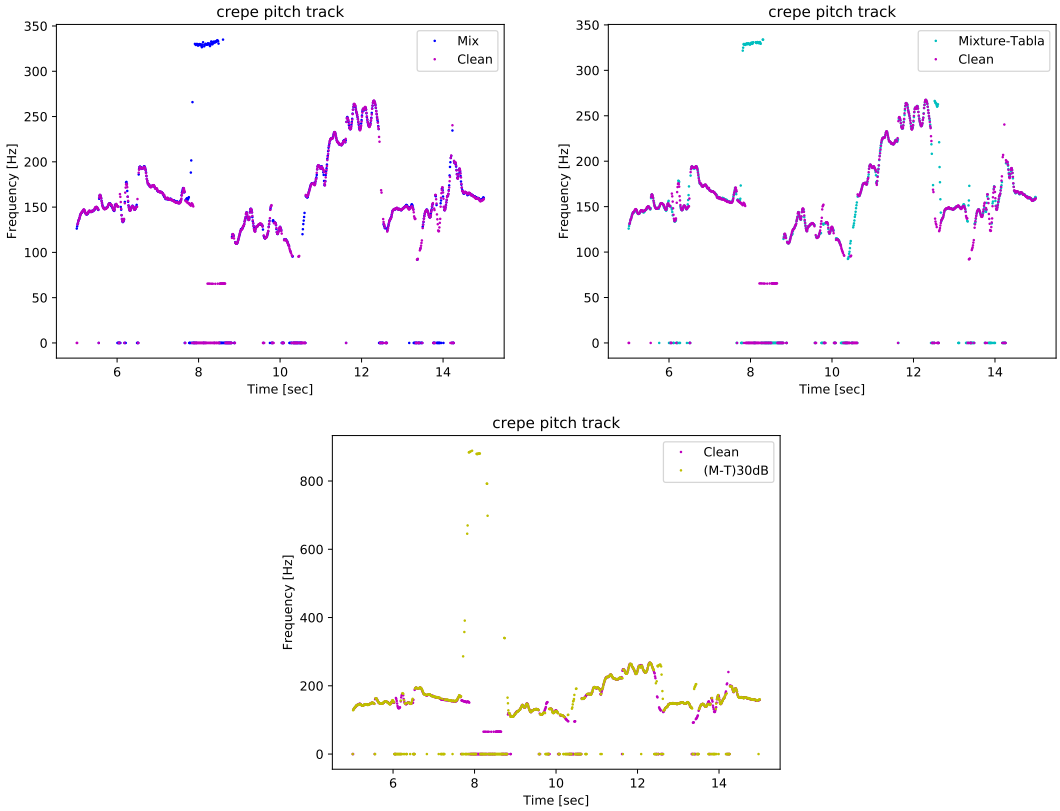


Figure 6.3: CREPE pitch tracks of excerpt from test example of (a) Mixture (b) Mixture-Tabla (c) (M-T)30dB compared with pitch track of clean vocals. (M-T)30dB refers to the track obtained by suppressing the tanpura of the Mixture-Tabla track at 30dB level.

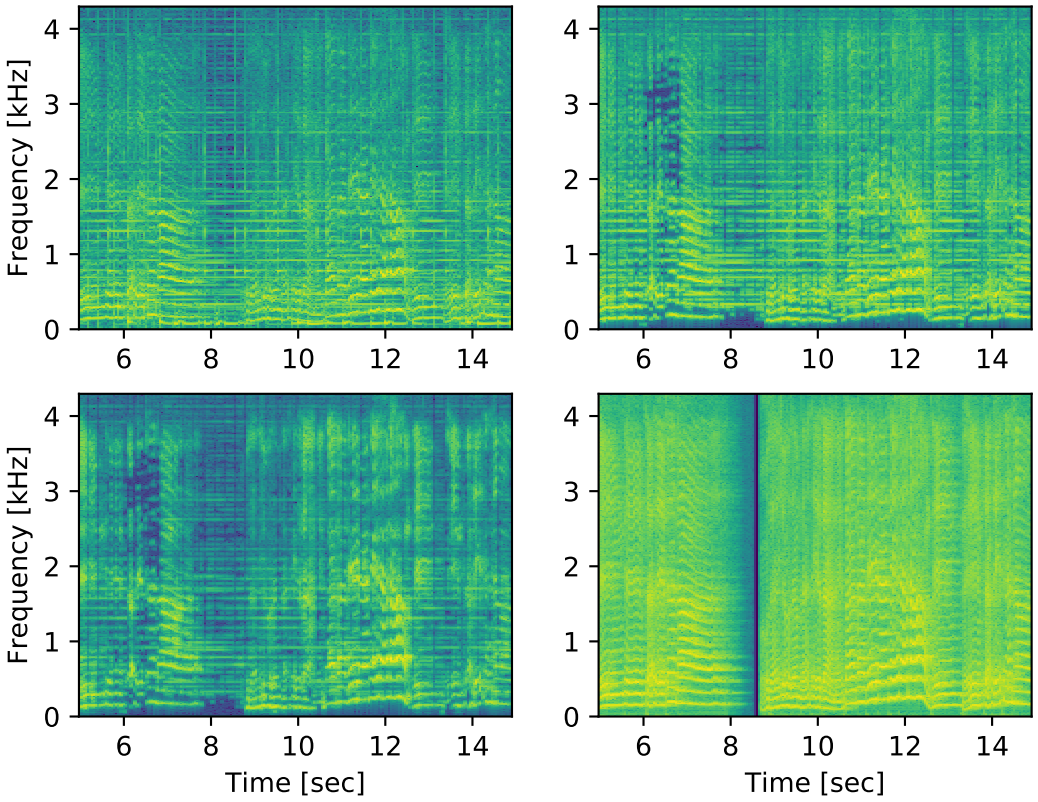


Figure 6.4: (Clockwise from top-left) Spectrograms of (a)Mixture (b)Mixture-Tabla (c)Clean vocals (d)(M-T)30dB

Chapter 7

Potential applications: Music complexity analysis

The idea of complexity in Hindustani classical music is not very thoroughly understood in terms of objective metrics that can be directly extracted from the audio of a performance. That is not to say however, that an objective notion of complexity does not exist for this kind of music. Trained professionals would almost unanimously be able to distinguish between an amateur vocalist's rendition of a raga from that of an experienced singer. They would also be able to tell whether a particular performance of a composition is "good" or "bad".

There has been a fair bit of research about complexity in the context of Western music, but little work has been done to extend these concepts to Hindustani classical music.

In this chapter there will be a discussion on different facets of musical complexity and the numeric measures associated with them. This will also include discussions on how these measures could apply in our specific concept, or conversely what they could miss out on

7.1 Facets of complexity

A given melody could be considered complex for a variety of reasons. It could be because the sequence of notes in the harmony. It could also be because of the timing relative to the bar that these notes are played. In literature some objective notions for these aspects have been defined. They come under broadly 3 categories [2] :-

1. Pitch-related variables

2. Rhythm-related variables

3. Structure related variables

7.1.1 Pitch-related variables

These variables try to capture the complexity present in the notes and the sequence of notes that make up a melody :-

1. **Entropy of pitch class distribution:** 12 pitch classes considered for the case of western music. Weighted by notes duration to account for more perceptual salience of longer notes. This weighting done using model proposed in [30].

2. **Entropy of interval distribution:** Interval is difference in pitch between successive notes. There are 25 components in this (unison, +- in semitones). Finding the entropy of distribution gives an understanding of the first order distribution of the pitch.

For Western music it is enough to consider only MIDI notes to describe complexity. For Hindustani classical music however, we would like to also be able to characterise the transitions between notes, vibrato around notes and other details which form an integral part of the vocals We would like to come up with measures that use not just the MIDI notes but the entire pitch track.

Having the ability to obtain the pitch track of the vocals (often the primary melodic instrument in Hindustani music) from a concert audio, would be of great help in being able to accurately obtain such measures.

7.1.2 Rhythm-related variables

These variables try to capture the complexity that arises due to varying note duration and the timing of them being played :-

1. **Entropy of note duration of melody:** This involves first classifying note duration into discrete categories and the finding entropy of this distribution

2. Rhythmic variability can be captured by finding standard deviation of log of note durations

3. **Note Density:** This is given as the number of notes played per second.

7.1.3 Structure-related variables

These variables try to capture the complexity from a broader perspective of the structure of the entire song :-

1. **Tonal Ambiguity:** This measure distinguishes how stable a pitch class is in a given key. It uses key profile values [31] corresponding to each pitch class with respect to the key of the score. Key profile values give in a sense the distribution of pitch classes within a given key. This could be extended to the concept of ragas for Hindustani classical music instead of keys. The way ragas are defined is a bit more complicated than keys as it not only specifies the notes but also transitions between notes.

2. **Self similarity of melodic contour:** This measure deals with melodic device of repeating melodic segments or phrases. Calculated after normalising pitch contour (mean pitch = 0, standard deviation = 1) .

$$S(t) = \frac{\int_0^T \rho(R)dt}{T} \tag{7.1}$$

a R is autocorrelation of contour, ρ is half wave rectification, T is total time.

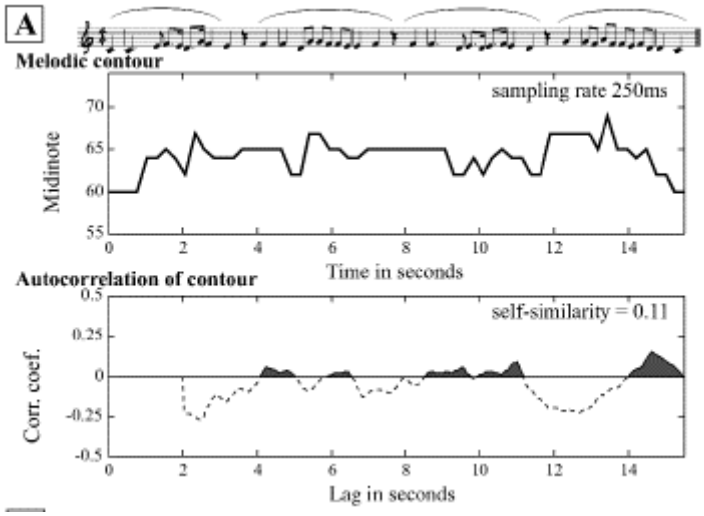


Figure 7.1: Calculation of self-similarity [2]

Chapter 8

Conclusion and Future work

8.1 Conclusion

The aim of this thesis was to perform source separation of tabla and vocals from Hindustani vocal concert audios, to aid in performing improved transcription of both these components. We first perform an exhaustive literature survey of the different approaches used in tackling the source separation problem, specifically in the context of music. We then present the BSS eval [3] metrics, a set of objective measures used to evaluate source separation performance, and also propose to use vocal and tabla transcription accuracy measures as an additional way of evaluating performance. Based on the survey, we find that data driven techniques have shown the best source separation performance in recent times [7]. Hence, we choose the state-of-the-art *Open Unmix* [1] as our baseline architecture.

Next, we present a detailed study of this architecture, describing its layers, input and output representations, the training procedure used and the hyperparameters of interest. We find that this architecture trained on a western music dataset (MUSB18 [4]), is not able to perform satisfactory source separation on Hindustani vocal mixtures, because of the difference in characteristics of the music. The absence of a source separation dataset specifically for Hindustani music led us to synthesise one from audios collected from various sources. We develop a test set that is representative of realistic vocal concert audios. The training and validation sets are synthesised with the aim of efficiently utilising the limited available data.

We first focus on the tabla source source separation task. We use our created dataset to train the *Open Unmix* architecture for the tabla target. Tabla separated by this trained model gives improved onset detection and stroke classification performance as

compared to performance obtained on mixture audio as well as tabla separated by available trained pecussion separation model (*Open Unmix* trained on drums target using MUSDB18). To further improve the tabla separation performance, we propose to use a multi-task learning (MTL) approach.

We present a short survey on MTL approaches used to tackle problems such as tempo and beat estimation [25] and speaker source separation [26], and model our MTL approach based on these. We propose a modification of the *Open Unmix* architecture by introducing an additional branch to perform tabla onset detection, and perform experiments by varying the weighting of the losses from the two branches, training procedure and loss function used. We find that this architecture trained such that the source separation branch is trained with MSE only upto 4kHz, along with the onset detection branch (ODB), gives improved tabla separation in terms of transcription accuracy measures. The ODB also gives very good onset detection performance.

For vocal separation, we first attempt to train *Open Unmix* on the vocals+tanpura target, but find that the model does not train for multiple targets. We then use wiener filtering to 'subtract' the obtained tabla estimate from the mixture to obtain a vocals+tanpura estimate. Applying tanpura suppression on this using the noise reduction tool in Audacity, we get an estimate of the vocals. We find that suppression improves overall accuracy of the PYIN algorithm, by more correctly identifying unvoiced frames. The performance of the CREPE algorithm is found to be slightly degraded after tanpura suppression.

Last, we present a literature survey on musical complexity analysis, which is an application that would greatly benefit from improved vocal and tabla transcription on mixture audio.

8.2 Future Work

Future work in source separation for Hindustani music should focus on :-

- Further building up the dataset by adding more solo vocal audios, and reconstructing the dataset to allow training on a vocals only target.
- Constructing a similar dataset, replacing the vocals with instruments such as sitar as the harmonic lead.
- Expanding current dataset by also adding harmonium.

- Employing the multi-task learning approach for vocal separation, where the additional branch performs pitch tracking.
- Improving tabla source separation in terms of improved stroke classification. One approach could be developing a multi-task architecture, where the additional branch learns to perform stroke classification.

Bibliography

[1] F.-R. Stoter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix - a reference implementation for music source separation," *Journal of Open Source Software*, 2019.

[2] T. Eerola, T. Himberg, P. Toiviainen, and J. Louhivuori, "Perceived complexity of western and african folk melodies by western and african listeners," *Psychology of Music*, vol. 34, no. 3, pp. 337–371, 2006.

[3] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[4] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "Musdb18-hq - an uncompressed version of musdb18," Aug. 2019.

[5] A. Liutkus, F.-R. Stöter, Z. Rafii, D. Kitamura, B. Rivet, N. Ito, N. Ono, and J. Fontecave, "The 2016 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings* (P. Tichavský, M. Babaie-Zadeh, O. J. Michel, and N. Thirion-Moreau, eds.), (Cham), pp. 323–332, Springer International Publishing, 2017.

[6] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models." Late-Breaking/Demo ISMIR 2019, November 2019. Deezer Research.

[7] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, D. FitzGerald, and B. Pardo, "An overview of lead and accompaniment separation in music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307–1335, 2018.

[8] E. Vincent, H. Sawada, P. Bofill, S. Makino, and J. P. Rosca, "First stereo audio source separation evaluation campaign: data, algorithms and results," in *International Conference on Independent Component Analysis and Signal Separation*, pp. 552–559, Springer, 2007.

[9] S. Dixon, "Onset detection revisited," in *Proceedings of the 9th International Conference on Digital Audio Effects*, vol. 120, pp. 133–137, Citeseer, 2006.

[10] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *Proceedings of the 39th IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6979–6983, IEEE, 2014.

[11] P. Chordia, "Segmentation and recognition of tabla strokes," in *6th International Society for Music Information Retrieval Conference (ISMIR 2005)*, 2005.

[12] O. K. Gillet and G. Richard, "Automatic labelling of tabla signals," in *4th International Society for Music Information Retrieval Conference (ISMIR 2003)*, 2003.

[13] K. Narang and P. Rao, "Acoustic features for determining goodness of tabla strokes.," in *ISMIR*, pp. 257–263, 2017.

[14] A. Srinivasamurthy, A. Holzapfel, K. K. Ganguli, and X. Serra, "Aspects of tempo and rhythmic elaboration in hindustani music: A corpus study," *Frontiers in Digital Humanities*, vol. 4, p. 20, 2017.

[15] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 161–165, IEEE, 2018.

[16] M. Mauch and S. Dixon, "pyin: A fundamental frequency estimator using probabilistic threshold distributions," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 659–663, IEEE, 2014.

[17] J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gómez, and J. P. Bello, "An analysis/synthesis framework for automatic f0 annotation of multitrack datasets.," in *ISMIR*, pp. 71–78, 2017.

[18] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, Citeseer, 2014.

[19] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017.

[20] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," 2017.

[21] Z.-C. Fan, Y.-L. Lai, and J.-S. R. Jang, "Svsgan: Singing voice separation via generative adversarial network," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 726–730, IEEE, 2018.

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[24] S. Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[25] S. Böck, M. E. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, 2019.

[26] Y. Tu, J. Du, Y. Xu, L. Dai, and C.-H. Lee, "Speech separation based on improved deep neural networks with dual outputs of speech features for both target and interfering speakers," in *The 9th International Symposium on Chinese Spoken Language Processing*, pp. 250–254, IEEE, 2014.

[27] C. S. Doire and O. Okubadejo, "Interleaved multitask learning for audio source separation with independent databases," *arXiv preprint arXiv:1908.05182*, 2019.

[28] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 771–775, IEEE, 2020.

[29] M. A. Rohit and P. Rao, "Acoustic-prosodic features of tabla bol recitation and correspondence with the tabla imitation," in *Interspeech*, 2018.

[30] R. Parncutt, "A perceptual model of pulse salience and metrical accent in musical rhythms," *Music Perception: An Interdisciplinary Journal*, vol. 11, no. 4, pp. 409–464, 1994.

[31] C. L. Krumhansl and E. J. Kessler, "Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys.," *Psychological review*, vol. 89, no. 4, p. 334, 1982.