Music Source Separation

Dr. Vipul Arora
Project Supervisor

Aaditya Singh
160002

Introduction

In the general source separation problem , we are given one or more mixture signals that contain different mixtures of some original source signals. This is illustrated in the following Figure 1 where four sources, namely vocals, drums, bass and guitar are all present in the mixture. The task is to recover one or more of the source signals given the mixtures [1].

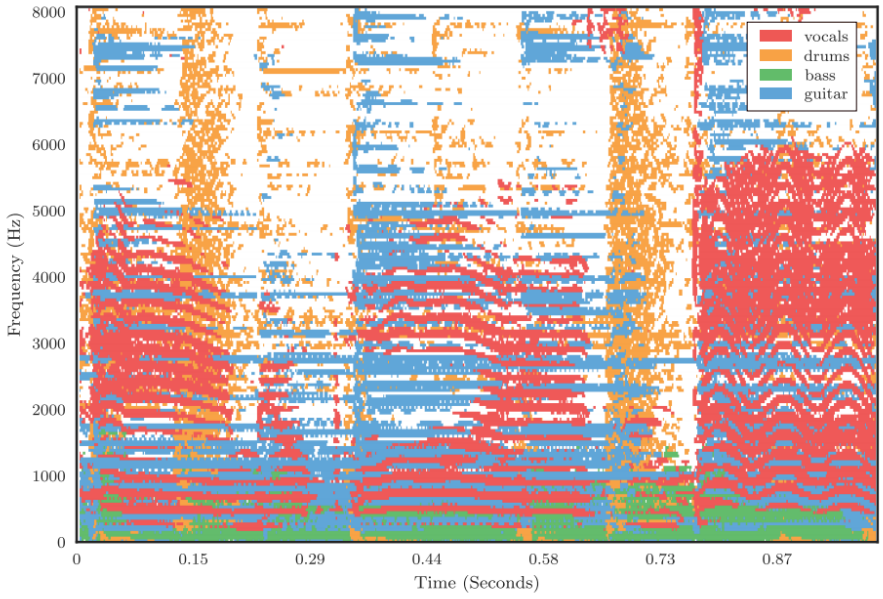


Figure 1: Representation of a music mixture in the time-frequency domain. The dominant musical source in each time-frequency bin is displayed with a different color.

Music signals have distinct characteristics that clearly differentiate them from other types of audio signals such as speech or environmental sounds. In a general sense, musical sources are often categorized as either predominantly harmonic, predominantly percussive, or as singing voice.

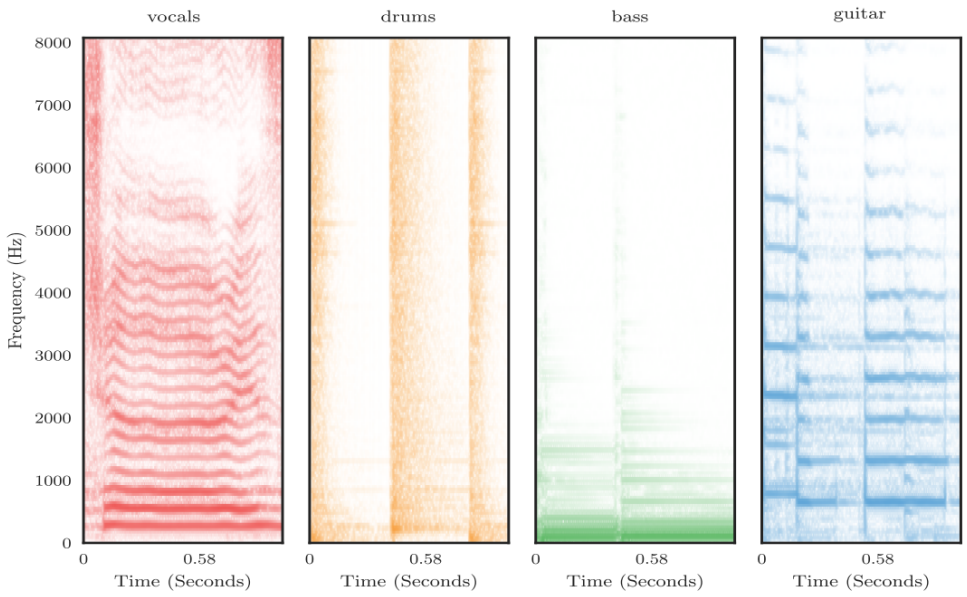


Figure 2: Magnitude spectrogram of four example music signals: vocals (left), drums (mid-left), bass (mid-right) and acoustic guitar (right). The horizontal axis represents time in seconds, and the vertical axis frequency in Hz.

Harmonic signals exhibit a clear structure composed of a fundamental frequency F0 and a harmonic series. For most instruments, the harmonics appear at multiple integers of the fundamental frequency: for a given F0 at 300 Hz, a harmonic component can be expected close to 600 Hz, the next harmonic around 900 Hz, and so on.

Harmonic sources exhibit a relatively stable behavior over time, and can typically be identified in the spectrogram as horizontal components. This can be observed in Figure 2 where a series of notes played by an acoustic guitar are displayed.

In contrast to harmonic signals, which contain only a selected number of harmonic components, percussive signals contain energy in a wide range of frequencies. Percussive signals exhibit a much flatter spectrum, and are highly localized in time with transient-like characteristics. This can be observed in the drums spectrogram in Figure 2, where clear vertical structures produced by the drums signal can be observed.

In reality, most music signals contain both harmonic and percussive elements. For example, a note produced by a piano is considered predominantly harmonic, but it also contains a percussive attack produced by the hammer hitting the strings.

Similarly, singing voice is an intricate combination of harmonic (voiced) components, produced by the vibrations of the vocal chords, and percussive-like (plosive) components including consonant sounds such as k or p where no vocal fold vibration occurs. These components are in turn filtered by the vocal cavity, with different formant frequencies created by changing the shape of the vocal cavity. As seen in Figure 2, singing voice typically exhibits a higher rate of pitch fluctuation compared to other musical instruments.

A notable property of musical sources is that they are typically sparse in the sense that for the majority of points in time and frequency, the sources have very little energy present. This is commonly exploited in Music Source Separation, and can be clearly seen for each of the sources in Figure 2.

Fundamental Frequency Estimation

We first performed the fundamental frequency estimation for the leading voice by following the method presented in the paper [2].

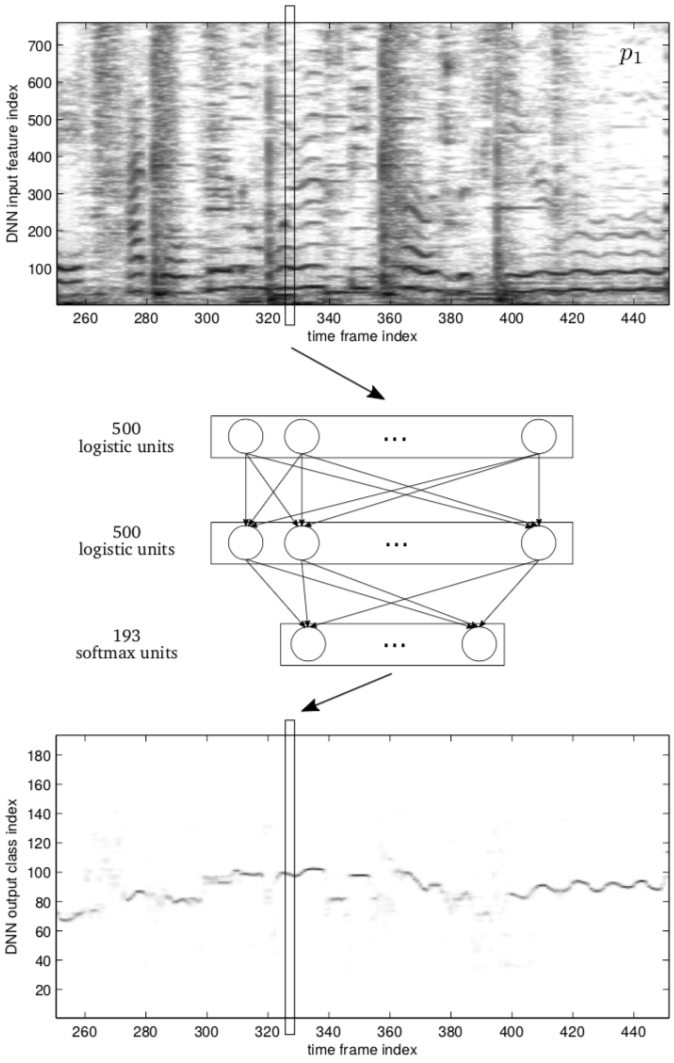


Figure 3: f0 estimation network illustration.

Figure 3 shows the network used to estimate the fundamental frequency of the leading voice. We use the logarithm of the modulus of the Short Time Fourier Transform (log-spectrogram), computed from a Hamming window of duration 64 ms with 0.75 overlap.

At each time frame, the log-spectrogram is fed into a classifier to predict one of the f0 classes which are uniformly spread between the minimum and maximum observed f0 values. This becomes a classification task, and we try to minimize the cross entropy loss (log loss) by using an optimizer such as Adam or Stochastic Gradient Descent.

Experiments and Results

We have used the MIREX-05 dataset, which has 31 western songs in various genres, as the training and validation dataset for the the architecture presented in this paper. We varied most of the hyperparameters including the window size, step size, learning rate, batch size and number of epochs.

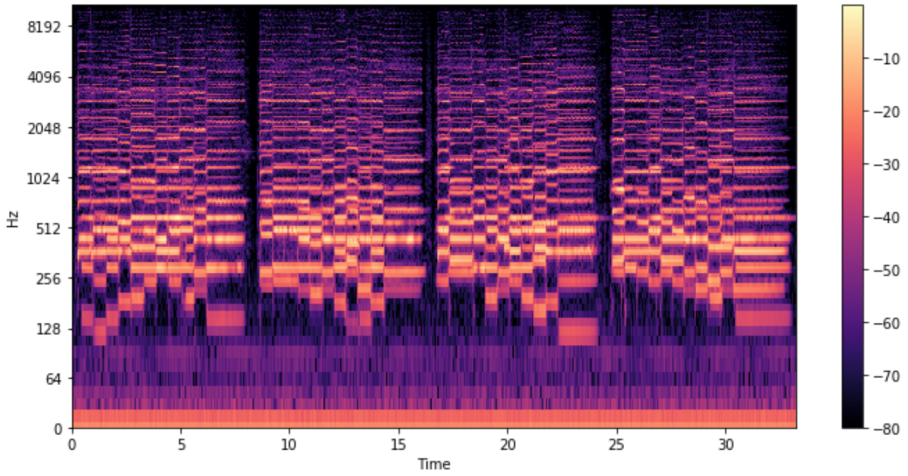


Figure 4: Magnitude spectrogram of the song *Fuer Deinen Thron* obtained by taking a window size of 64ms with 85% overlap.

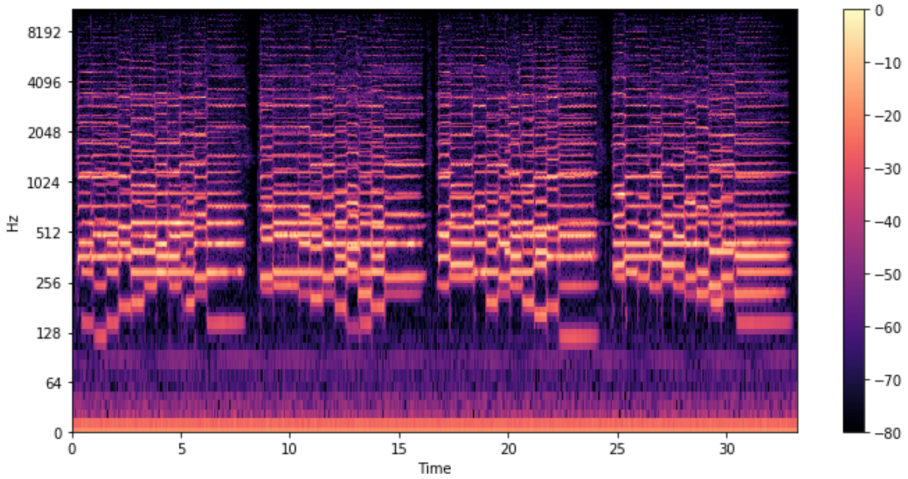


Figure 5: Magnitude spectrogram of the song *Fuer Deinen Thron* is made sharper by taking a window size of 84ms with 88% overlap.

We used a training to validation ratio of 4:1, and achieved the best training and validation classification accuracies of 48% and 92% by using a window size of 64ms, a step size of 16ms, Adam optimizer with learning rate 0.001, a batch size of 32 and 5 epochs.

Non-Negative Matrix Factorization

We wanted to learn about unsupervised learning algorithms for the separation of sound sources, and the common approaches for this purpose are either based on non-negative matrix factorization (NMF) or probabilistic latent component analysis (PLCA). We learned about NMF from the paper [3]. and how it can be used for source separation from the paper [4].

NMF algorithms tries to approximate a non-negative matrix V as the product of two non-negative matrices W and H. The significance of the approximation is that each column of V can be rewritten column by column as $v \sim Wh$, where v and h are the corresponding columns of V and H. In other words, each data vector v is approximated by a linear combination of the columns of W, weighted by the components of h. Therefore W can be regarded as containing basis vectors that can linearly approximate the data in V.

The paper [3] presents two algorithms for NMF based on iterative updates of W and H, which minimize the cost function (Euclidean or Kullback-Leibler divergence) on each update. The Euclidean distance between two non-negative matrices V and WH is

$$||V - WH||^2 = \sum_{ij} ||V_{ij} - (WH)_{ij}||^2 \tag{1}$$

The iterative updates for W and H for minimizing this Euclidean distance are

$$H_{a\mu} \leftarrow H_{a\mu} \frac{W^T V}{(W^T W H)_{a\mu}} \qquad\qquad W_{ia} \leftarrow W_{ia} \frac{(V H^T)_{ia}}{(W^T W H)_{ia}} \tag{2}$$

The Kullback-Leibler divergence between two non-negative matrices V and WH is

$$D(V||WH) = \sum_{ij} V_{ij} log(V_{ij}/(WH)_{ij}) - V_{ij} + (WH)_{ij} \tag{3}$$

The iterative updates for W and H for minimizing this Kullback-Leibler divergence are

$$H_{a\mu} \leftarrow H_{a\mu} \frac{\sum_i (W_{ia} V_{i\mu})/(WH)i\mu}{\sum_k W_{ka}} \qquad\qquad W_{ia} \leftarrow W_{ia} \frac{\sum_\mu (H_{a\mu} V_{i\mu})/(WH)i\mu}{\sum_\nu H_{a\nu}} \tag{4}$$

Source Separation

The paper [4] presents an unsupervised learning algorithm for MSS based on a signal model, where the magnitude spectrum vector $\mathbf{x_t}$ in frame t is modeled as a linear combination of basis functions $\mathbf{b_j}$ and their time-varying gains $g_{j,t}$, t = 1,...,T with T being the number of frames. This can be written as

$$\mathbf{x_t} = \sum_{j=1}^{J} g_{j,t} \mathbf{b_j} \tag{5}$$

The time-domain signals and their complex spectra sum linearly. However, the phase spectra of natural sounds are very unpredictable and the estimation of the frame-wise phases of the sources would make the model too complex. Moreover, the human auditory perception is quite insensitive to phase. Therefore, the linear addition of the complex spectra is approximated as a linear addition of magnitude spectra. The obsrvation vector and the basis functions are magnitude spectra in this paper. The equation (5) can be rewritten using the matrix notation as

$$\mathbf{X} = \mathbf{BG} \tag{6}$$

where $\mathbf{X} = [\mathbf{x_1},...,\mathbf{x_T}]$, $\mathbf{B} = [\mathbf{b_1},...,\mathbf{b_T}]$, and $[\mathbf{G}]_{j,t} = g_{j,t}$. In this paper, we try to estimate the sources after knowing only the observation matrix $\mathbf{X}$, and and matrices $\mathbf{B}$ and $\mathbf{G}$ are estimated without any prior knowledge which is specific to the sources.

Each component is modeled using a fixed magnitude spectrum, which is non-negative by definition. The basis functions are thus restricted to be entry wise non-negative and additive, meaning that the gains are also restricted to be non negative. Therefore, we can apply NMF to decompose $\mathbf{X}$ into the product of $\mathbf{B}$ and $\mathbf{G}$.

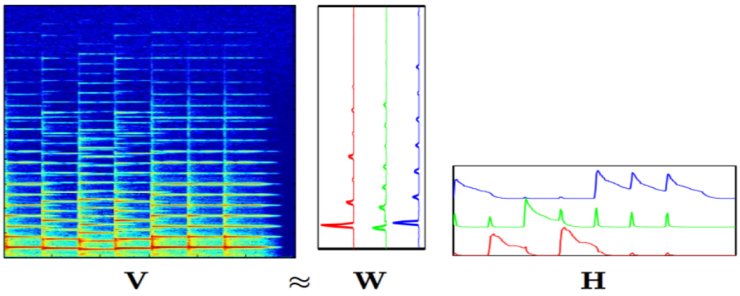


Figure 6: The magnitude spectrum $\mathbf{V}$ is expressed as a linear combination of the basis vectors in $\mathbf{W}$ and their weights for each time frame in $\mathbf{H}$

.

An additional constraint is imposed on the matrix $\mathbf{G}$ for the temporal continuity criterion to increase the robustness of the separarion. Temporal continuity of the components is measured by assigning a cost to large changes between the gains $g_{j,t}$ and $g_{j,t-1}$. The cost function for the temporal continuity is given by

$$c_t(\mathbf{G}) = \sum_{j=1}^{J} \frac{1}{\sigma_j^2} \sum_{t=2}^{T} (g_{j,t} - g_{j,t-1})^2 \tag{7}$$

Estimation of $\mathbf{B}$ and $\mathbf{G}$ is done by minimizing a cost function which is the weighted sum of the reconstruction error term $c_r(\mathbf{B}, \mathbf{G})$ and the temporal continuity term $c_t(\mathbf{G})$.

$$c(\mathbf{B}, \mathbf{G}) = c_r(\mathbf{B}, \mathbf{G}) + \alpha c_t(\mathbf{G}) \tag{8}$$

In the estimation algorithm, the matrices $\mathbf{B}$ and $\mathbf{G}$ are first randomly initialized with random positive values and then alternatively updated with multiplicative update rules. In the cost function (8), $\mathbf{B}$ only affects the reconstruction error term $c_r(\mathbf{B}, \mathbf{G})$ for the minimization of which the NMF update rules can be used. The update rules for $\mathbf{B}$ is given by

$$\mathbf{B} \leftarrow \mathbf{B}. \times \frac{\frac{\mathbf{X}}{\mathbf{BG}}\mathbf{G^T}}{\mathbf{1}\mathbf{G}^T} \tag{9}$$

The gradients of the reconstruction error term $c_r(\mathbf{B}, \mathbf{G})$ and the temporal continuity term $c_t(\mathbf{G})$ with respect to $\mathbf{G}$ are given by

$$\nabla c_r(\mathbf{B}, \mathbf{G}) = \mathbf{B}^T(\mathbf{1} - \frac{\mathbf{X}}{\mathbf{BG}}) \tag{10}$$

$$[\nabla c_t(\mathbf{G})]_{j,t} = 2T\frac{2g_{j,t} - g_{j,t-1} - g_{j,t+1}}{\sum_{i=1}^{T} {g_{j,i}}^2} - T\frac{2g_{j,t}\sum_{i=2}^{T}(g_{j,i} - g_{j,i-1})^2}{(\sum_{i=1}^{T} g_{j,i}^2)^2} \tag{11}$$

The gradient is written as a subtraction of element-wise non-negative terms as $\nabla c(\mathbf{BG})$ = $\nabla c^+(\mathbf{BG})$ - $\nabla c^-(\mathbf{BG})$ where $\nabla c^+(\mathbf{BG})$ = $c_r^+(\mathbf{B}, \mathbf{G})$ + $\alpha c_t^+(\mathbf{G})$ and $\nabla c^-(\mathbf{BG})$ = $c_r^-(\mathbf{B}, \mathbf{G})$ + $\alpha c_t^-(\mathbf{G})$. Here the element-wise positive terms of the gradients of reconstruction error cost and temporal continuity cost are given by

$$\nabla c_r^+(\mathbf{B}, \mathbf{G}) = \mathbf{B}^T\mathbf{1} \qquad \nabla c_r^-(\mathbf{B}, \mathbf{G}) = \mathbf{B}^T\frac{\mathbf{X}}{\mathbf{BG}} \tag{12}$$

$$[\nabla c_t^+(\mathbf{G})]_{j,t} = \frac{4Tg_{j,t}}{\sum_{i=1}^{T} {g_{j,i}}^2} \tag{13}$$

$$[\nabla c_t^-(\mathbf{G})]_{j,t} = 2T\frac{g_{j,t-1} + g_{j,t+1}}{\sum_{i=1}^{T} {g_{j,i}}^2} + \frac{2Tg_{j,t}\sum_{i=2}^{T}(g_{j,i} - g_{j,i-1})^2}{(\sum_{i=1}^{T} g_{j,i}^2)^2} \tag{14}$$

The update rule for $\mathbf{G}$ is given by

$$\mathbf{G} \leftarrow \mathbf{G}. \times \frac{\nabla c_r^-(\mathbf{B}, \mathbf{G})}{\nabla c_r^+(\mathbf{B}, \mathbf{G})} \tag{15}$$

Limitations

There is no reliable method for the automatic estimation of the number of components, thus it has to be set manually. In practice, a large number of components can be used, which are then clustered to sound sources. Moreover, we do not know which component belongs to which source, therefore supervised clustering cannot be used. In the paper [4], the unsupervised clustering methods which try to create the component cluster for each source were tested but deteriorated the results. The preferred method was automatic clustering, where original signals before mixing were used as reference for clusters.

Probabilistic Latent Component Analysis

We wanted to learn about more meaningful decompositions of the magnitude spectrogram, so we read about Probabilistic Latent Component Anaysis (PLCA) and how it can be used for source separation from the thesis [5]. We discuss about PLCA in this section.

If $\mathbf{V}$ represents the magnitude spectrogram of a given song snippet, the random variable f represents the frequency index and t represents the time frame. Then PLCA allows us to characterize the joint distribution $P(f,t)$ as

$$P(f,t) = \sum_z P(z)P(f|z)P(t|z) \tag{16}$$

This is a symmetrical decomposition of $P(f,t)$ obtained by considering both f and t dimensions as features. We can instead have a different decomposition by treating the two dimensions differently.

$$P(f,t) = P(t)\sum_z P(f|z)P(z|t) \tag{17}$$

Latent Variable Model

Consider a random process characterized by the probability $P(f)$ of drawing a feature unit f in a given draw. Let the random variable f take values from the set 1,...,F. $P(f)$ is unknown, what we observe instead is the feature counts, that is the number of times f is observed after repeated draws.

We can approximate $P(f)$ by using a normalized set of counts. Assume $P(f)$ comes from K hidden distributions or latent factors, The distributions are selected according to their relative probabilities which remain the same for an experiment. We wish to characterize these hidden distributions.

Let $P(f|z)$ be the probability of observing feature f given z, which represents the index of the hidden distribution being considered. The probability of choosing the z^{th} distribution in the t^{th} experiment is $P_t(z)$

$$P_t(f) = \sum_z P(f|z)P_t(z) \tag{18}$$

Parameter Estimation

Let V_{ft} represents the feature count of f in the t^{th} experiment, which is known. We want to estimate $P_t(z)$ and $P(f|z)$. We use a maximum likelihood formulation of the problem, and the standard procedure of solving it in latent variable models is the Expectation-Maximization Algorithm. For the E-step, we obtain a posteriori probability for the latent variable as

$$P_t(z|f) = \frac{P_t(z)P(f|z)}{\sum_z P_t(z)P(f|z)} \tag{19}$$

For the M-step, we obtain the re-estimation equations

$$P(f|z) = \frac{\sum_t V_{ft}P_t(z|f)}{\sum_f \sum_t V_{ft}P_t(z|f)} \tag{20}$$

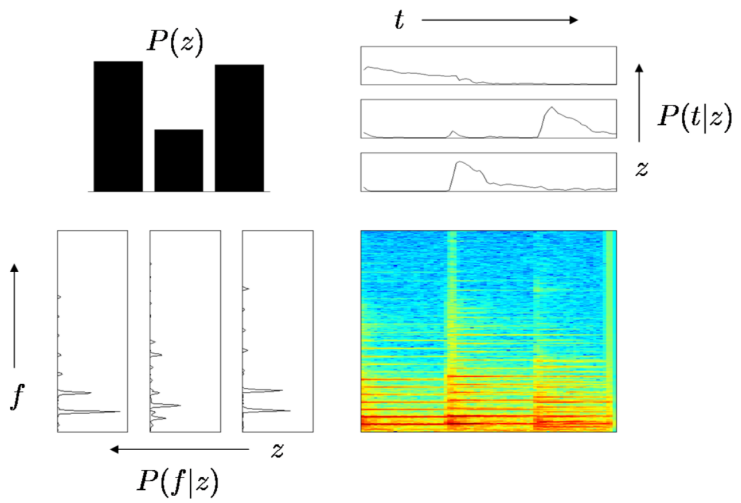


Figure 7: Illustration of PLCA applied on a spectrogram of three piano notes (bottom-right). The top-right panel shows the extracted time marginals and the bottom-left panel shows the extracted frequency marginals. The top-left plot shows the mixture weights $P(z)$

.

and

$$P_t(z) = \frac{\sum_f V_{ft} P_t(z|f)}{\sum_z \sum_f V_{ft} P_t(z|f)} \tag{21}$$

The parameters $P(f|z)$ and $P_t(z)$ are randomly initialized and re-estimated using the above equations until a termination condition is met.

Source-Filter based Probabilistic Latent Component Analysis

For obtaining an even better semantic interpretation which can lead to an enhanced modeling, we read about the Source-Filter based PLCA from the thesis [6]. In the Source-Filter model, each pitched sound is modeled as consisting of spectral peaks at the integer multiples of its F0 value, known as the harmonics, and their envelope is shaped by a band pass filter.

Figure 8 shows the graphical model for source-filter based PLCA. The spectrogram is decomposed with the help of discrete latent variable p,s,z,a,f, which can take N_p,N_s,N_z,N_a,N_f values, respectively. The pitch values are given at each time t, which are indexed by p. The source models are indexed by s, each of which has spectral dictionaries indexed by z.

Each dictionary component comprises of the weights of N_a band pass filters, indexed by a. This band-pass filter bank consists of triangular magnitude response filters, with centers distributed in a geometric progression on a linear scale. This is used to model the non-linear human hearing behavior.

Source-Filter based PLCA allows us to characterize the joint distribution $P_t(f)$ as

$$P_t(f) = \sum_{p,s,z,a} P_t(f|p,a) P_t(p) P_t(s|p) P_t(z|p,s) P(a|s,z) \tag{22}$$

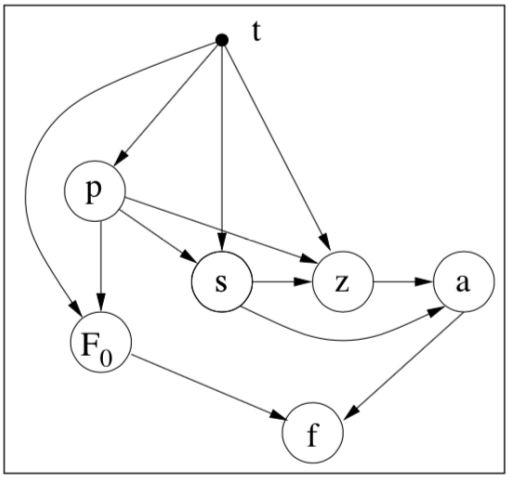


Figure 8: Graphical model for source-filter based PLCA

.

where $P_t(f|p,a)$ represents the spectrum of the output of the ath filter, at time t and pitch p. It is modeled using the source-filter model as

$$P_t(f|p,a) = \frac{e(f|F_0^{p,t})h(f|a)}{\sum_f e(f|F_0^{p,t})h(f|a)} \tag{23}$$

Here $e(f|F_0^{p,t})$ is a fixed spectrum consisting of the Gaussian harmonic peaks placed at the integer multiples of the given $F0$ associated with the pitch index p at time t. $h(f|a)$ is the magnitude transfer function of the ath triangular band pass filter.

$P_t(p)$ can be interpreted as the contribution fraction of the source with pitch p to the whole spectrum. $P_t(s|p)$ is the probability that it is the sth source which corresponds to the pitch p at the time instant t. $P_t(z|p,s)$ is the probability of the zth dictionary component of source s, whose spectral envelope is formed by $P(a|s,z)$, in the form of coefficient of ath band-pass filter.

PLCA Generative Learning

The PLCA decomposes the signal into various components according as can be seen in Equation (22). The decomposition takes place with a goal that the model closely represents the observed spectra. This is accomplished using Expectation Maximization (EM) algorithm which minimizes the Kullback-Leibler (KL) divergence between $P_t(f)$ and $V(f,t)$ normalized suitably. The likelihood

$$\mathcal{L}_G = \sum_{f,t} V(f,t) \log P_t(f) \tag{24}$$

is maximized by sequentially iterating the E and M steps, subject to the constraints that the probabilities sum to unity. In the E-step, the current model probabilities are used to find the posterior joint distribution of the latent variables.

$$P_t(p,s,z,a|f) = \frac{P_t(f|p,a)P_t(p)P_t(s|p)P_t(z|p,s)P(a|s,z)}{P_t(f)} \tag{25}$$

where $P_t(f)$ is given by Equation (22).

In the M-step, the model probabilities are updated, so as to maximize the likelihood function, as

$$P_t(p) = \frac{\sum_{f,s,z,a} V(f,t)P_t(p,s,z,a|f)}{\sum_p \sum_{f,s,z,a} V(f,t)P_t(p,s,z,a|f)} \tag{26}$$

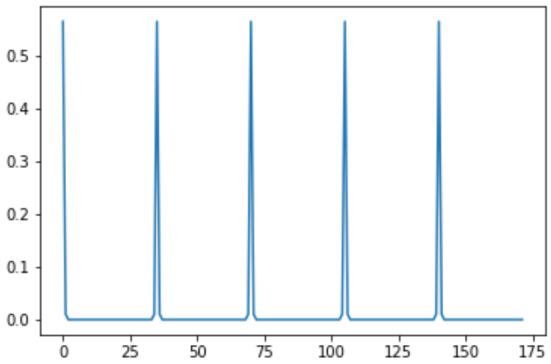
$$P_t(s|p) = \frac{\sum_{f,z,a} V(f,t) P_t(p,s,z,a|f)}{\sum_s \sum_{f,z,a} V(f,t) P_t(p,s,z,a|f)} \tag{27}$$

$$P_t(z|p,s) = \frac{\sum_{f,a} V(f,t) P_t(p,s,z,a|f)}{\sum_z \sum_{f,a} V(f,t) P_t(p,s,z,a|f)} \tag{28}$$

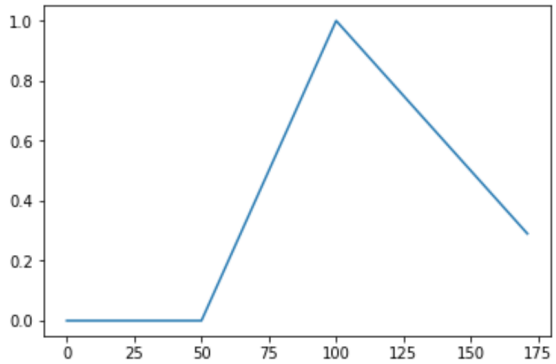
$$P(a|s,z) = \frac{\sum_{f,t,p} V(f,t) P_t(p,s,z,a|f)}{\sum_a \sum_{f,t,p} V(f,t) P_t(p,s,z,a|f)} \tag{29}$$

Experiments and Results

Bach10 [7] dataset consists of the fundamental frequency of every source which is being played at each time frame. Using this information, we can create the basic building blocks $P_t(f|p,a)$ as shown in the following figures.



(a) $e(f|F_0^{p,t})$ for $F0$ = 35 and $\sigma = 0.5$



(b) $h(p|a)$ centered at f_r = 50 with r = 2

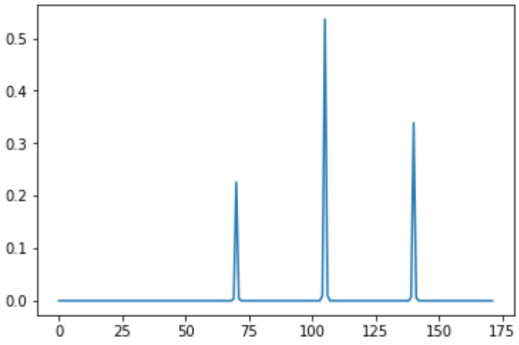


Figure 9: $P_t(f|p,a)$ for $F0$ = 35, $\sigma = 0.5$,f$_r$ = 50 and r = 2

.

As PLCA is an iterative procedure, it takes a significant amount of time for training even on GPU's. In order to reduce the training time, we have considered the spectrogram bins only upto 2048 Hz instead of the original 11025 Hz. This results in the loss of higher order harmonics, but the fundamental and many of the lower order harmonics are still retained so the overall quality of the melody is not compromised. We have chosen f_r = 50 and r = 2, from which we obtain N_a = 7 for having the last peak of the triangular filter around 2048 Hz. The rest of the chosen hyperparameters are σ = 0.5, N_s = 4, N_p = 4 and N_z = 3.

Reconstruction

After running the iterative updates of PLCA for a desired number of iterations (5 in our case), we first create $P_t(f|s)$ by using Equation (22).

$$P_t(f|s) = \sum_{p,z,a} P_t(f|p,a)P_t(p)P_t(s|p)P_t(z|p,s)P(a|s,z) \tag{30}$$

Then $V(f,t|s)$ is reconstructed as

$$V(f,t|s) = \frac{V(f,t)P_t(f|s)}{P_t(f)} \tag{31}$$

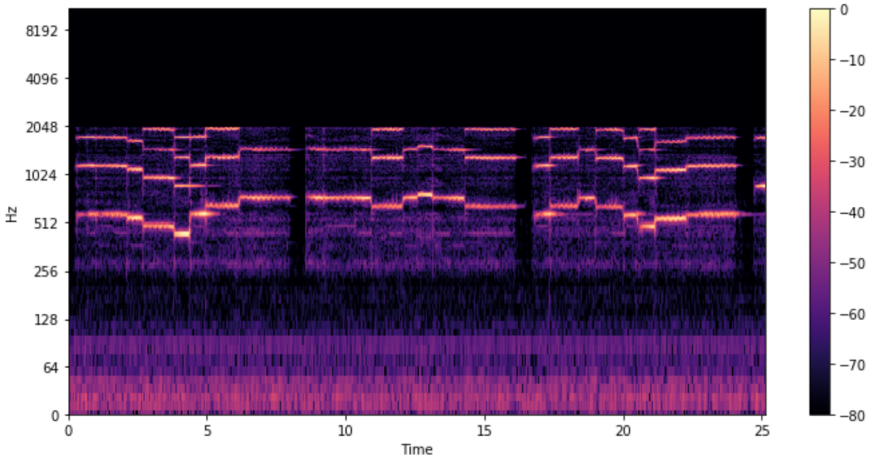


Figure 10: Actual Violin spectrogram for the song *Fuer Deinen Thron*

.

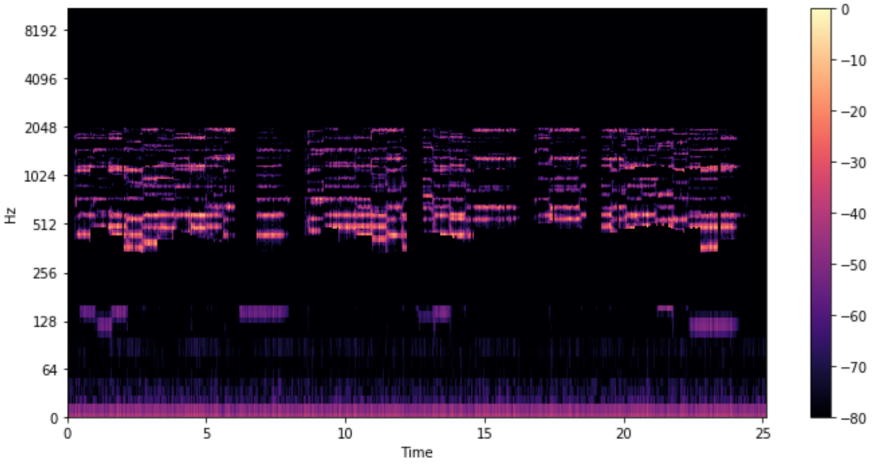


Figure 11: Reconstructed Violin spectrogram for the song *Fuer Deinen Thron*

.

Although the reconstructed spectrogram shown in Figure 10 seems to be quite different from the actual spectrogram shown in Figure 11, it was able to capture the peaks and the shape of the spectrogram around 512 Hz, 1024 Hz and 2048 Hz.

Music Source Identification

PLCA assumes the magnitude spectrogram $V(f,t)$ of the polyphonic signal to be a histogram of *energy quanta* piled up in the time-frequency bins, indexed by t and f respectively, and generated by an underlying probability distribution function (pdf) $P_t(f)$. It further assumes that the magnitude spectrum of polyphonic audio is the linear sum of the magnitude spectra of individual source signals.

We have tried two unique and independent approaches to introduce non-linearity in the source-filter based model for PLCA for handling the task of Music Source Identification. In Music Source Identification, we try to estimate which sources are being played after knowing the fundamental frequencies of the instruments (but not the instruments) at each time frame.

Thus, the pitch values are deterministic at each time frame t, which are indexed by p. At a given time frame t and pitch value p, the source index s needs to be determined. Hence, this becomes a classification task where we want to predict $P_t(s|p)$ for each s for all t and p. We try to minimize the cross entropy loss (log loss) between the actual and the predicted probabilities of the sources by using an optimizer such as Adam or Stochastic Gradient Descent.

Deep Probabilistic Framework-I

Equation (30) represents $P_t(f|s)$ as a linear combination of the basis building blocks $P_t(f|p,a)$ and their weights $P_t(z|p,s) * P_t a|s,z)$. We rewrite Equation (30) as

$$P_t(f|s) = \sum_{p=0}^{N_p} (\sum_{z,a} P_t(f|p,a)P_t(z|p,s)P(a|s,z))P_t(s|p)P_t(p) \tag{32}$$

At a given t and p, $F0$ is deterministic. We recall that $e(f|F_0^{p,t})$ is a fixed spectrum consisting of the Gaussian harmonic peaks placed at the integer multiples of the given $F0$ associated with the pitch index p at time t. We observe that the Gaussian harmonic peaks $e(f|F_0^{p,t})$ rarely intersect each other for different values of $F0$ when they are non-zero, as illustrated in the Figure 12

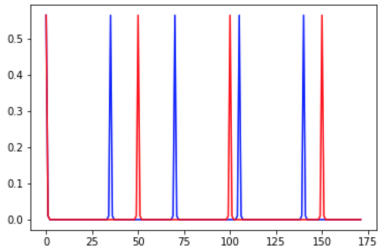


Figure 12: $e(f|F_0^{p,t})$ with σ = 0.5 for $F0$ = 35 and 50 (blue and red, respectively)

.

We can rewrite the Equation (32) by multiplying $P_t(f|p,a)$ on both sides

$$P_t(f|s) * P_t(f|p,a) = \sum_{p=0}^{N_p} P_t(f|p,a) * (\sum_{z,a} P_t(f|p,a)P_t(z|p,s)P(a|s,z))P_t(s|p)P_t(p) \tag{33}$$

Using the above observation, we approximate the Equation (34) by setting the terms where p are unequal to be zero.

$$P_t(f|s) * P_t(f|p,a) = P_t(f|p,a) * (\sum_{z,a} P_t(f|p,a)P_t(z|p,s)P(a|s,z))P_t(s|p)P_t(p) \tag{34}$$

Thus, for a given t and p, the Equation (35) gets reduced to

$$P_t(f|s) = (\sum_{z,a} P_t(f|p,a)P_t(z|p,s)P(a|s,z))P_t(s|p)P_t(p) \tag{35}$$

We have tried to use a neural network to estimate the weights associated with the basic building blocks $P_t(f|p,a)$, and then obtain $P_t(s|p)$ as

$$P_t(s|p) = \sum_f \frac{P_t(f|s)}{\sum_a P_t(f|p,a) * weights} \tag{36}$$

We first run a couple of PLCA iterative updates to roughly estimate $P_t(f|p,a)$ and $P_t(f|s)$. At each time frame t, and for a given pitch value p, $P_t(f|s)$ is a 2d matrix of shape $N_s * N_f$ and $P_t(f|p,a)$ is a 2d matrix of shape $N_a * N_f$. We feed $P_t(f|p,a)$ into a convolutional neural network to estimate a vector of shape $N_f * 1$, and then take the ratio of $P_t(f|s)$ and this vector to obtain a vector of shape $N_s * 1$ which represents $P_t(f|s)$ for every source index s.

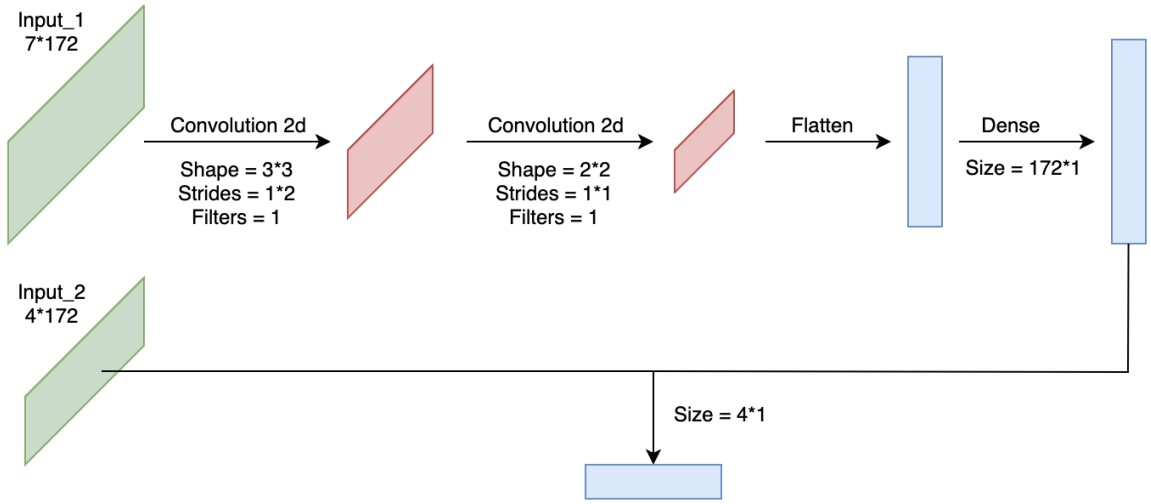


Figure 13: CNN architecture for estimating $P_t(s|p)$, with N_a = 7, N_f = 172 and N_s = 4. Here Input_1 is $P_t(f|p,a)$ and Input_2 is $P_t(f|s)$.

We chose the size of the kernels and the strides such that the height and the width reduces significantly at each convolutional layer. The larger number of parameters are caused by a large N_f, and therefore we have used just a single filter at each layer for a lesser number of parameters.

Experiments and Results

We kept most of the hyperparameters same as in our implementation of source-filter based PLCA to avoid dimensional mismatch, including the window size of 84ms, step size of 10ms with N_f = 172, N_a = 7, N_s = 4 and N_z = 3. For training the convolutional neural network, we have used Adam optimizer with a learning rate of 0.0001 and a batch size of 32 for 10 epochs. We split the training and validation in a 4:1 ratio, with 8 training songs and 2 validation songs.

With this configuration, we achieved the best training and validation accuracy of 26% and 25% respectively. For our reference, we also tried to replace $P_t(f|s)$ with the normalized spectrograms of the sources. This led to the best training and validation accuracy of 73% and 70%, which shows that there is much scope of improvement in the reconstructed probabilities.

Deep Probabilistic Framework-II

Equation (27) expresses $P_t(s|p)$ as linear combination of the product of $V(f,t)$ and $P_t(p,s,z,a|f)$. We want to establish a non-linear relationship as

$$P_t(s|p) = \mathcal{F}(\sum_f V(f,t)P_t(p,s,z,a|f)) \tag{37}$$

where $\mathcal{F}$ is a non-linear function mapping real numbers to real numbers, which we would expect a neural network to learn. We have summed over f as N_f is the largest dimension and we want to reduce the number of parameters in the network. Furthermore, as the information in the adjacent time-frequency bins are related, we have preferred to use a convolutional neural network to learn the non-linear relationship.

We first run a couple of PLCA iterative updates to roughly estimate $P_t(p,s,z,a|f)$. At each time frame t, and for a given pitch value f, $\sum_f V(f,t)P_t(p,s,z,a|f)$ is a 3d matrix of shape $N_a * N_s * N_z$ which we feed into a convolutional neural network to estimate a vector of shape $N_s * 1$ which represents $P_t(f|s)$ for every source index s.

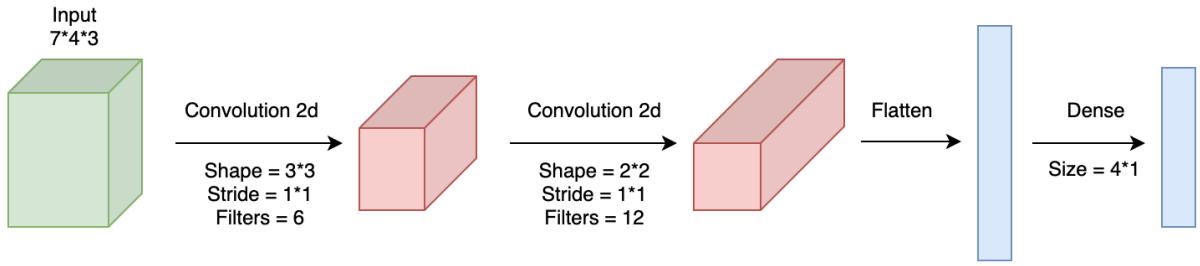


Figure 14: CNN architecture for estimating $P_t(s|p)$, with N_a = 7, N_s = 4 and N_z = 3.

.

We chose the size of the kernels such that the height and the width reduces significantly at each convolutional layer, and chose the number of filters such that the channels approximately double after each each convolutional layer. This is a standard practice adopted in visual recognition because the deeper layers presents more details and hence more filters are required.

Experiments and Results

We kept most of the hyperparameters same as in our implementation of source-filter based PLCA to avoid dimensional mismatch, including the window size of 84ms, step size of 10ms with N_a = 7, N_s = 4 and N_z = 3. For training the convolutional neural network, we have used Adam optimizer with a learning rate of 0.0001 and a batch size of 32 for 10 epochs. We split the training and validation in a 4:1 ratio, with 8 training songs and 2 validation songs. With this configuration, we achieved the best training and validation accuracy of 62% and 61% respectively.

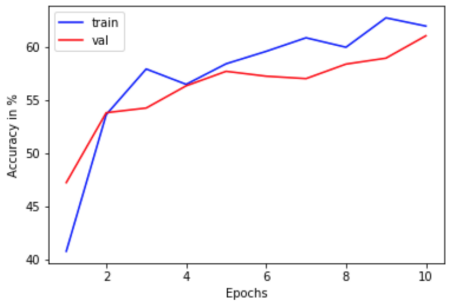


Figure 15: Plot of Training and Validation Accuracy versus Epochs.

.

Conclusion

In this undergraduate research project, We first learned about the fundamental properties of music, and implemented a neural network presented in the paper [2] to estimate the fundamental frequency of the leading voice in the music signal. We realized that our implementation is more of a brute force approach to do so, and for our future implementations we decided to shift to more meaningful representations of the music signal.

We then learned about unsupervised learning algorithms for the separation of sound sources, which are either based on non-negative matrix factorization (NMF) or probabilistic latent component analysis (PLCA). We first learned about NMF and the iterative update rules from the paper [3] and then learned about a method presented in the paper [4] to apply NMF for the task of sound source separation. We then discussed the major limitations of this method; there is no way to either determine the number of components or to know which components belong to which sound source.

To overcome these limitations, we learned about PLCA and the iterative update rules from the thesis [5]. For a better semantic representation and enhanced modeling, we learned about source-filter based PLCA from the thesis [6]. We then implemented the source-filter based PLCA for a better understanding and obtained the results for sound source separation. From our implementation and its results, we conclude that the source-filter based PLCA model works well, but the updates are slow due to large time-frame and frequency-frame dimensions even after dimensionality reduction.

For the task of music source identification, we have proposed two unique and independent deep learning based methods. These methods essentially use PLCA to obtain useful but rough semantic information's from the music signal, and then use neural networks to eliminate the linearity which is inherent in PLCA. We have shown through our results that one of the methods works well even when it is trained on a small dataset, and the other method has the potential to do so if the informations obtained from PLCA are accurate.

In the future, we would prefer to devise an end to end pipeline which performs fundamental frequency estimation for the different sound sources, then uses them for music source identification and finally performs music source separation. The proposed methods in this project will surely be an essential part of this pipeline.

In the end I would like to express my deep gratitude towards **Dr. Vipul Arora**, my mentor and project supervisor who supported me throughout the project and without whose guidance this work wouldn't have been possible.

References

[1] Musical Source Separation: An Introduction

https://hal.inria.fr/hal-01945345/document

[2] Singing Voice Melody Transcription Using Deep Neural Networks

https://pdfs.semanticscholar.org/027e/cd18db48bc49d4211f6281b321b9b34dfa56.pdf?_ga=2.137627190.702119950.1554731680-52286691.1554731680

[3] Algorithms for Non-negative Matrix Factorization

http://papers.nips.cc/paper/1861-algorithms-for-non-negative-matrix-factorization.pdf

[4] Monaural Sound Source Separation by Nonnegative Matrix Factorization With Temporal Continuity and Sparseness Criteria

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4100700

[5] Latent Variable Framework for Modeling and Separating Single-Channel Aousti Soures

http://shashanka.net/stuff/dissertation.pdf

[6] Analysis of Pitched Polyphonic Music for Source Transcription

https://drive.google.com/file/d/0By8wZfM49Y2ScC1vc2lVX0I1c1U/view

[7] Bach10 Dataset —-A Versatile Polyphonic Music Dataset

http://music.cs.northwestern.edu/data/Bach10_Dataset_Description.pdf