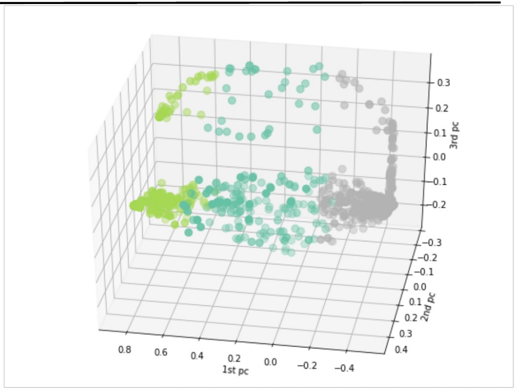
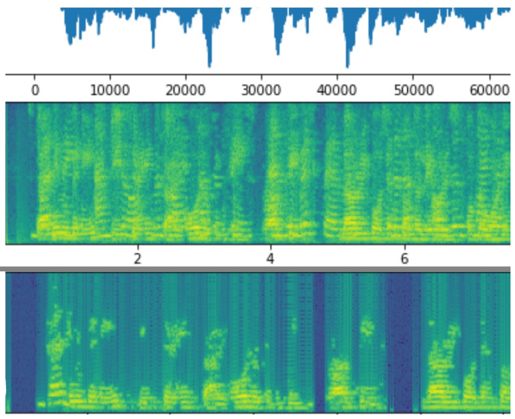
CAPSTONE PROJECT REPORT



SINGLE CHANNEL SPEAKER SEPARATION

Presented by:

DIVYANSHU RAJ
JAYASHREE DHAL
SAURABH MANRAL
SOORMA DAS
ABDUL HANNAN QAZI
MOHIT BISARIA



Praxis Business School

Capstone Project Report

Single Channel Speech Separation using Deep Clustering

Project Group Members:
Divyanshu Raj
Jayashree Dhal
Saurabh Manral
Soorma Das
Abdul Hannan Qazi
Mohit Bisaria

Supervisor:
Subhasis Dasgupta

A report submitted in fulfilment of the requirements for the Degree of PGPDS

in the

July 2019 Batch
Data Science

17th April 2020

Abstract

The problem of accoustic source separation is addressed using *Deep Clustering*, a deep learning framework. A deep neural network, using a sequence of *Bi-Directional Long Short Term Memory* networks, is utilised to produce spectrogram embeddings that are discriminative for partition labels given in training data. These embeddings are compact and are amenable to simple clustering methods. A simple clustering method such as *KMeans* can be used to decode the output embeddings. On training the model with spectrogram features containing mixtures of two speakers, and testing on the mixture of held-out set of speakers, the model was able to separate speech. However, the separation was not clean when it came to same gender speech mixtures.

Contents

List of Figures

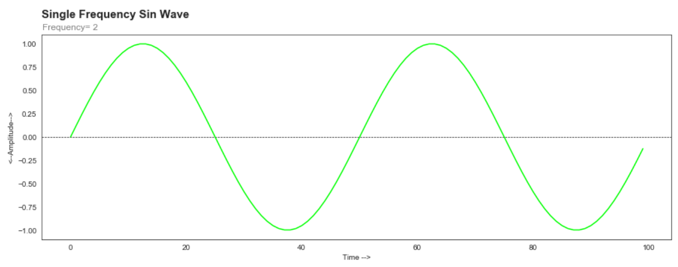
Chapter 1

Preliminaries

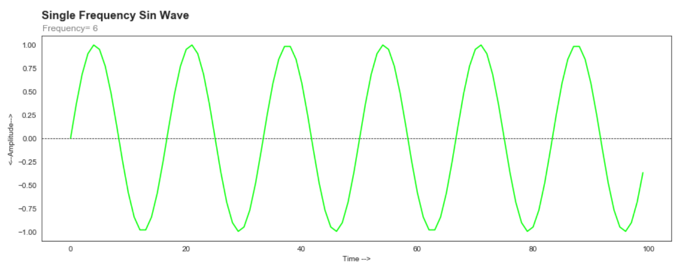
1.1 Sound Signal

A sound signal is an electronic representation of the sound waves - the basic phenomena behind the transfer of sounds from one place to another in space. The sound waves travel through vibrations in the particles of the medium between the source and destination. These vibrations add up to create the resultant sound waves. Similarly, the sound signal can be understood as a cumulative result of multiple waves, travelling at different frequencies, that add up to give the resultant signal.

When it comes to signal processing, the Sinusoidal wave is the building block behind all signals. A pure Sine wave travels at a constant frequency and has a characteristic amplitude and initial phase. Following figures show how multiple sinusoidal signals having different frequencies add up to form a resultant signal with a different frequency.



All audio signals are made up of thousands of single frequency sinusoids,



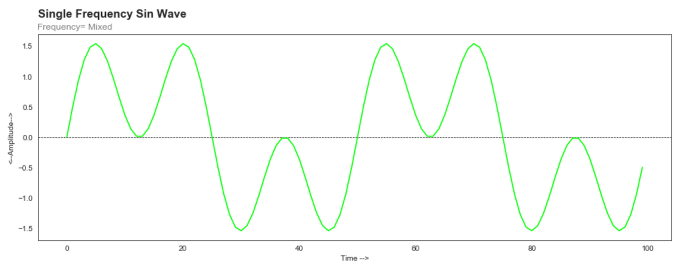


Figure 1.1: Sine Waves

whose amplitude changes with time. The resulting audio signal, thus, has a frequency that varies with time.

1.2 Fourier Transform

After visualising the time domain signal above, we observe the change of amplitude with time. In case of high frequency signal, these changes are so fast that they become unnoticeable to human ears. The time domain analysis of sound signal, thus, carries very limited information that can be used to characterise different sound samples.

One potent approach to better analyse a sound signal is to decompose it into its constituent single frequency sinusoids. Then we can analyse these individual sinusoids based on their frequency value and their magnitude in the original sound signal. This will help us characterise sound signals.

Fourier Transform is a mathematical technique that enables us to do this analysis. Fourier transforms break down a time domain signal, say $x(t)$ into its constituent frequencies in frequency domain. The Fourier transform formula is given below:

$$C(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt \tag{1.1}$$

The output of this function is a complex number, real part of which represents the frequency (ω) present in the original signal, while the imaginary part gives us the initial phase lag of the frequency.

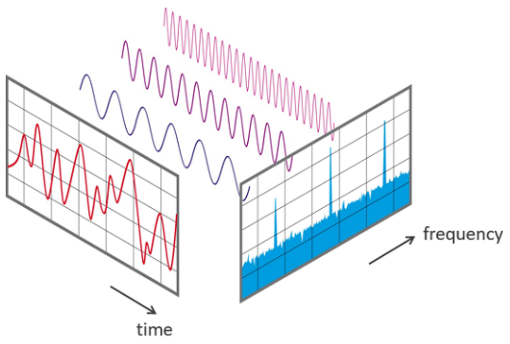


Figure 1.2: Fourier Transform

In Figure 1.2, the red wave represents a sound signal, while the three pink waves represent the constituent sinusoids of different frequencies. The blue graph is obtained after Fourier transformation and gives the magnitude of the three main frequencies in the original signal, along with magnitude of several other less significant frequencies.

An inverse Fourier transform performs just the opposite function. It takes a frequency domain representation of a signal and uses it to reconstruct the original time domain signal.

1.3 Fast-Fourier Transform

Fourier Transform provides frequency information averaged over the entire signal duration. Short-Time Fourier Transform (STFT), on the other hand, provides time-localised frequency information of a signal whose frequency changes over time. The only difference between a FT (Fourier Transformation) and STFT is that STFT uses discrete values as input, where as FT requires a continuous input. Python implementation of STFT is done using a function called FFT, over a windowed audio signal.

1.4 Window Signal

In order to feed discrete inputs to the FFT function, the continuous valued audio signal must be discretised into various short-time samples. For this we use a Window Signal. A window signal can be both smooth and step, based on the requirement. We have used a Hanning Window signal, which becomes '0' valued outside some specific time interval. The Hanning Window Function is multiplied with our audio signal, and the result is a discrete signal having a non-zero value for only a specific time interval.

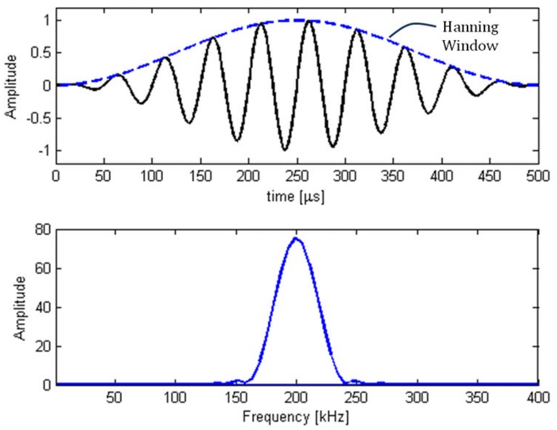


Figure 1.3: Hanning Window

Chapter 2

Auditory Scene Analysis

Auditory scene analysis seeks to identify the components of audio signals corresponding to individual sound sources in a mixture signal. The approach is to formulate a set of *elements* in the signal using an indexed set of features. These features carry multi-dimensional information about part of the signal. Elements for the audio signal are typically defined in terms of time-frequency coordinates. This is *Segmentation* approach.

Segmenatation problem can either be *class-based* or *partition-based*. In the *class-based* problem, the goal is to learn from training class labels to label *known* object classes. *Partition-based* segmentation problem, however, is more general where the task is to learn from labels of partitions, without requiring object-class labels, to segment the input. Solving the partition-based problem has the advantage that *unknown* objects can also be segmented.

2.1 Embedding

We can not feed an audio file to a ConvNet. So, we map the audio to vectors of real numbers. This mapping is known as *embedding*. In order to solve the partition-based segmentation problem for single-channel multi-speaker audio inputs, the model is made to learn embeddings for each input elements, such that the correct labelling can be determined by simple clustering methods. We focus on sigle-channel audio-domain because using segmentation as a mask, we can extract parts of the target signals that are not corrupted by other signals.

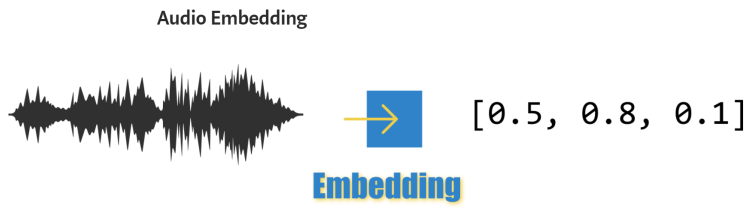


Figure 2.1: Audio Embedding

2.2 Deep Embedding for Clustering

Let x be a raw input signal (time-domain waveform) and $X_n = g_n(x)$, $n \in 1, ..., N$, a feature vector indexed by an element n. In case of audio signals, n may be a time-frequency index (t, f), where t indexes frames of the signal and f frequencies, and $X_n = X_{t,f}$ the value of the complex spectrogram at the corresponding time-frequency bin. We assume that there exists a partition of the elements n into regions which we would like to find. In case of audio-source separation, these regions could be defined as the sets of the time-frequency bins in which each source dominates. Estimating such a partition would enable us to build time-frequency masks to be applied to X_n, leading to time-frequency represenations that can be inverted to obtain isolated sources.

To estimate the partition, we aim to find out a K-dimensional embedding $V = f_\theta(x) \in \mathbb{R}^{N \times K}$, parameterized by θ, such that performing some simple clustering in the embedding space will likely lead to a partition of $1, ..., N$ that is close to the target one. Here we base $V = f_\theta(x)$ to a deep neural network that is a global function of the entire input signal x. Thus our transformation can take into account global properties of the input, and the embeddings can be considered permuatation- and cardinality-independent encoding of the network's estimate of the signal partition.

We consider a unit-norm embedding, so that $|v_n|^2 = \sum_k v_{n,k}^2 = 1, \forall n$, where $v_n = \{v_{n,k}\}$ and $v_{n,k}$ is the value of the k-th dimension of the embedding for element n. The partition-based training requires a reference label indicator $Y = \{y_{n,c}\}$, mapping each element n to each of c arbitrary partition classes, so that $y_{n,c} = 1$ if element n is in partition c. For a training objective, we seek embeddings that enable accurate clustering according to the partition labels and which is invariant to the number and permutations of the partition labels from one training example to the next. One objective for minimization is

$$C(\theta) = |VV^T - YY^T|_W^2 = \sum_{i,j,y_i=y_j} \frac{(\langle v_i, v_j \rangle - 1)^2}{d_i} + \sum_{i,j,y_i \neq y_j} \frac{(\langle v_i, v_j \rangle - 0)^2}{\sqrt{d_i d_j}} \tag{2.1}$$

$$= \sum_{i,j,y_i=y_j} \frac{|v_i - v_j|^2}{d_i} + \sum_{i,j,y_i \neq y_j} \frac{(|v_i - v_j|^2 - 2)^2}{\sqrt{4 d_i d_j}} - N, \tag{2.2}$$

where $|A|_W^2 = \sum_{i,j} w_{i,j} a_{i,j}^2$ is a weighted Frobenius norm, with $W = d^{\frac{-1}{2}} d^{\frac{-T}{2}}$, where $d_i = YY^T 1$ is an $(N \times 1)$ vector of partition sizes i.e. $d_i = |\{j : y_i = y_j\}|$.

In the above equations, we use the fact that $|v_n|^2 = 1, \forall n$. Intuitively, this objective pushes the inner product $\langle v_i, v_j \rangle$ to 1 when i and j are in the same partition, and to 0 when they are in different partitions. We note that the first term is the objective function minimized by k-means, as a function of cluster assignments. This renders the second term to be a constant. So, the objective reasonably tries to lower the k-means score for the labeled cluster assignments as training time.

At test time, the embeddings V are computed on the test signal and the rows $v_i \in \mathbb{R}^K$ are clustered using k-means (say).

2.3 Bidirectional LSTM

Long Short Term Memory (LSTM) Architecture was an upgraded version of Recurrent Neural Network (RNN). It made long time lags accessible which was not possible with the earlier architectures because backpropagated error either blows up or decays exponentially. An LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. Each block contains one or more recurrently connected memory cells controlled by three gates - update gate, forget gate, and output gate.

Bidirectional LSTM (BLSTM) networks essentially put two independent LSTMs together. This structure allows the network to have both backward

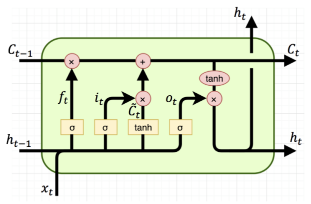


Figure 2.2: LSTM Memory Cell

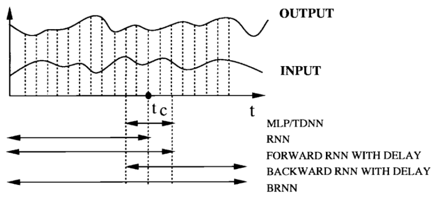


Figure 2.3: Input Data Usage

and forward information about a sequence of data at every time step. Using bidirectional network runs the inputs in two ways - one from past to future, and the other from future to the past. How this approach differs from the unidirectional network is that at any point of time, the information from both the past and the future is preserved, and the network has complete, sequential inforation about all points before and after that particular point of time. The network is free to use as much or as little of this information, as necessary.

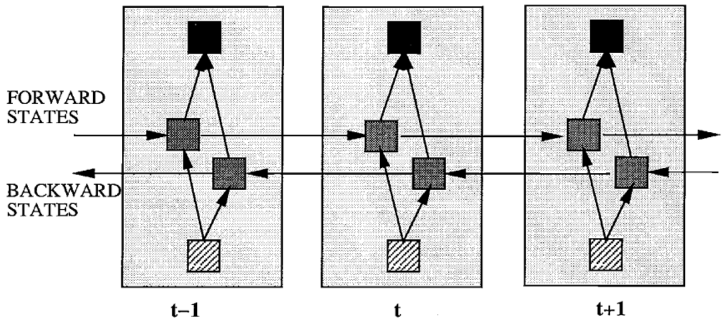


Figure 2.4: General Structure of a Bidirectional LSTM

2.4 Usage of BLSTM in Speaker Separation

Generally, sounds, words, and even whole sentences that at first mean nothing are found to make sense in the light of future context, and we as human beings make full use of it. Therefore, usage of BLSTM seems only consequential for the monoaural single channel speaker separation problem. The only assumption about the input data is that it can be divided into finitely long segments, and that it has negligible effect on the data itself.

Chapter 3

Model Implementation

3.1 Data Collection

The data set used is the TSP data set. The TSP data set consists of the voices of 12 men and 12 women. The voice content is Harvard sentence. Most of the 24 people are native speakers (Canadians). In Harvard sentence, there are 72 lists in total, and each list has 10 sentences. Each of these people reads 6 lists, so it is 12 men who read 6 lists, which is 7210 phrase sounds. 12 women have no one reading 6 lists, so this is also 7210 voices. We are using the data reorded at 8k Hz.

The wav files in the TSP data set are organized by speakers. For example, all sentences read by MA, on behalf of Male A form a folder, FA stands for Female A, and CA stands for Child A. $FE26\_09.wav$ stands for Female E list-26 sentence-09. In general, it is recommended to use FK, FL, MK, ML as the test set, and the rest (excluding CA CB) as the training set.

For *testing* purposes, we recorded the speech of different students of our class and mixed the speech of two students at a time. The mixed speech was passed into the trained model and the segregated speech was obtained.

3.2 Neural Network

The shape of the input data is ($batch\_size$ = 128, FRAMES_PER_SAMPLE = 100, NEFF = 129), which is $(batch\_size, time\_step\_size, input\_vec\_size)$.

There are 4 bidirectional LSTMs. The output of LSTM for each layer is $(outputs\_fw, outputs\_bw)$. The shape of $outputs\_fw$ and $outputs\_bw$ are the same. The output of the first layer is all concat so the input of the BLSTM after it is ($batch\_size = 128$, FRAMES_PER_SAMPLE $= 100, hidden = 600$).

After the 4−layer BLSTM, there is a fully connected neural network whose output is ($batch\_size$, EMBBEDDING_D * NEFF). Then $tanh$ activation is performed, and the final reshape ($batch\_size, NEFF = 129$, EMBBEDDING_D $= 40$) is the final output of the network.

3.3 Training Process

The first step is to organize the data, put the wav files of same speaker in the same directory, divide the training set and test set, and organize them according to required format. Then the audio files are converted to a pkl file. Input is read from the pkl file, divided into batches and then sent to the neural network.

Once the model is trained, we use it on our test data. For testing, we take one of the 2−*speaker* mixes recorded by the students of our class. We feed the *mixed* audio sample to the *trained* model, which generates an *embedding vector*. We apply KMeans clustering on the obtained embedding vector. These embedding vectors are nothing but *learned feature transformations*.

A time-domain speech signal is constructed based on time-frequency masks for each speaker. The time-frequency masks for each source speaker were obtained by clustering the row vectors of embedding V , where V was outputted from the proposed model for each segment, similarly to the training stage. The number of clusters corresponds to the number of speakers in the mixture.

The earlier mentioned objective function minimizes the distances between embeddings of elements within a partition, while maximizing the distances between embeddings for elements in different partitions. Partition label is derived by observing the *spectral dominance patterns*.

3.4 Results

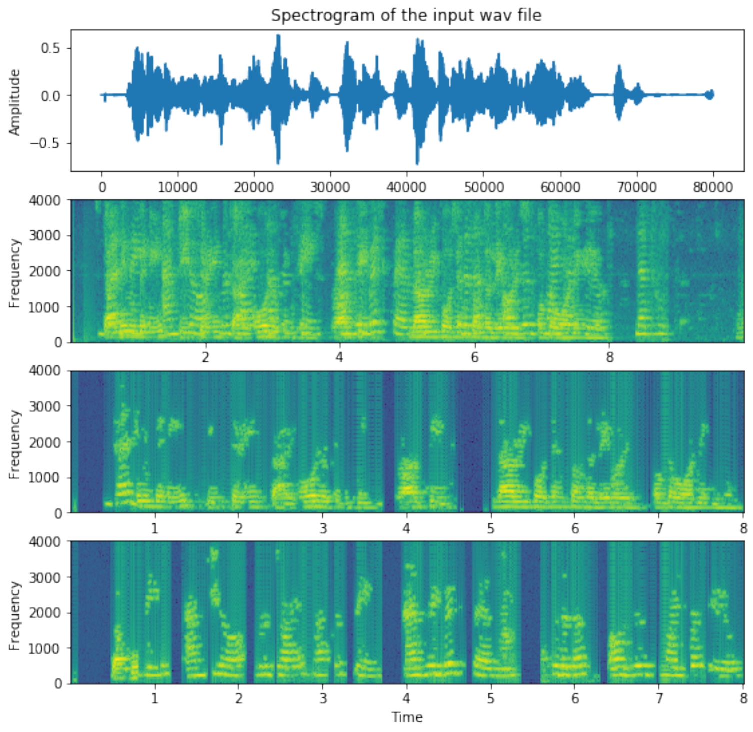
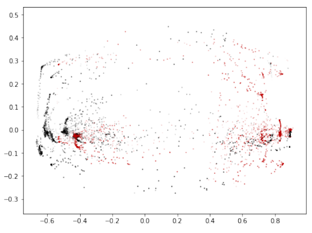


Figure 3.1: Log Spectrogram of test mix signal and of separated outputs

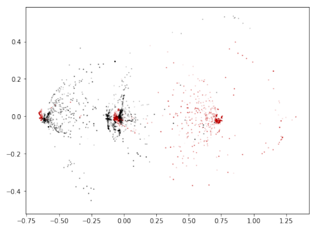
In Figure 3.1, the topmost plot is the input `.wav` file plotted in time domain. Below this plot are the spectrogram plots: first spectrogram plot is that of the the input `.wav` file, the second and the third ones are those of the final output audio files obtained after clustering.

The frequencies of the speakers or the pitch are identified with the brighter yellow columns present in the spectrum.

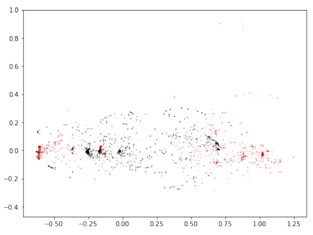
We can clearly observe the well-marked clusters formed in this case. The last spectrogram corresponds to the segregated Female voice which is marked by the presence of a sequence of brighter yellow columns compared to a dull yellow columns of the second-last spectrogram corresponding to the male voice.



(a) Mixed test signal

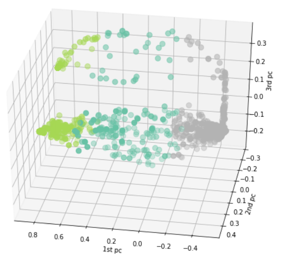


(b) Output 1

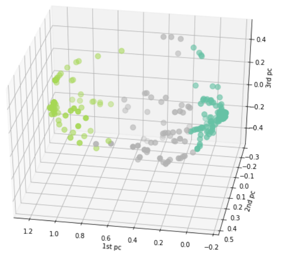


(c) Output 2

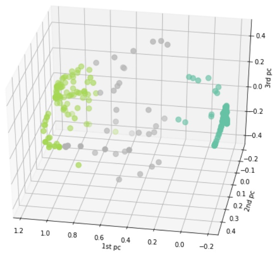
Figure 3.2: 2D Scatter plots of 3 Principal components



(a) Mixed test signal



(b) Output 1



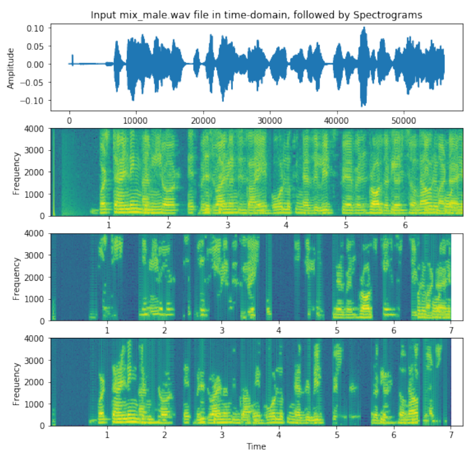
(c) Output 2

Figure 3.3: 3D Scatter plots of 3 Principal components

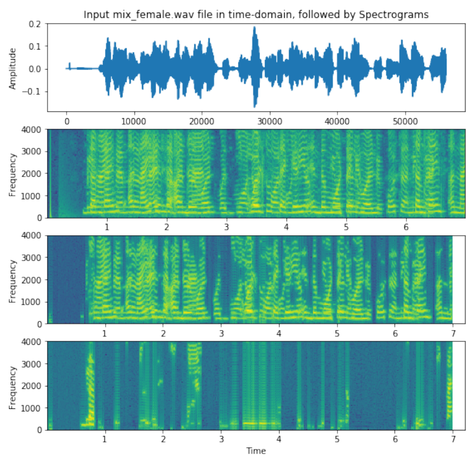
Figures 3.2 and 3.3 show the `planar` and the `3-Dimensional` scatter plots of the 3 Principal components obtained after clustering the `embedding vectors` of the test audio file and the two output files.

Chapter 4

Further Work



(a) mix male



(b) mix female

Figure 4.1: Log Spectrogram of ill-clustered output samples

We observed that our model was effective in segregating the audio voices very clearly for a `male-female` mix. However, separation was not clean when we considered `male-male` or `female-female` mixes.

Figure 4.1 clearly shows that the model was not able to form clearly demarcated clusters when the audio mix belonged to people of same gender. Further work needs to be done to sharpen the model to achieve desired speech separation for same gender audio mix samples.

References

[1] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, pp. 31–35.

[2] M. Cooke and D. P. Ellis, "The auditory organization of speech and other sources in listeners and computational models," *Speech communication*, vol. 35, no. 3-4, pp. 141–177, 2001.

[3] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *2014 22nd International conference on pattern recognition*, IEEE, 2014, pp. 1532–1537.

[4] M. Cooke, J. R. Hershey, and S. J. Rennie, "Monaural speech separation and recognition challenge," *Computer Speech and Language*, vol. 24, no. 1, pp. 1–15, 2010.

[5] Y. Wang, K. Han, and D. Wang, "Exploring monaural features for classification-based speech segregation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 270–279, 2012.