Tampere University

Shanshan Wang

# Online Speaker Separation Using Deep Clustering

Faculty of Information Technology and Communication Sciences (ITC)
Master's thesis
November 2019

# Abstract

Shanshan Wang: Online Speaker Separation Using Deep Clustering
Master's thesis
Tampere University
Master's Degree Programme in Data Engineering and Machine Learning
November 2019

---

In this thesis, a low-latency variant of speaker-independent deep clustering method is proposed for speaker separation. Compared to the offline deep clustering separation system, bidirectional long-short term memory networks (BLSTMs) are replaced with long-short term memory networks (LSTMs). The reason is that the data has to be fed to the BLSTM networks both forward and backward directions. Additionally, the final outputs depend on both directions, which make online processing not possible. Also, 32 ms synthesis window is replaced with 8 ms in order to cooperate with low-latency applications like hearing aids since the algorithmic latency depends upon the length of synthesis window. Furthermore, the beginning of the audio mixture, here, referred as buffer, is used to get the cluster centers for the constituent speakers in the mixture serving as the initialization purpose. Later, those centers are used to assign clusters for the rest of the mixture to achieve speaker separation with the latency of 8 ms. The algorithm is evaluated on the Wall Street Journal corpus (WSJ0). Changing the networks from BLSTM to LSTM while keeping the same window length degrades the separation performance measured by signal-to-distortion ratio (SDR) by 1.0 dB, which implies that the future information is important for the separation. For investigating the effect of window length, keeping the same network structure (LSTM), by changing window length from 32 ms to 8 ms, another 1.1 dB drop in SDR is found. For the low-latency deep clustering speaker separation system, different duration of buffer is studied. It is observed that initially, the separation performance increases as the buffer increases. However, with buffer length of 0.3 s, the separation performance keeps steady even by increasing the buffer. Compared to offline deep clustering separation system, degradation of 2.8 dB in SDR is observed for online system.

**Keywords:** Speaker Separation, Low Latency, Deep Clustering.

# Acknowledgement

Firstly, I really appreciate the opportunity working as a research assistant, given by Prof. Tuomas Virtanen also being as my supervisor, to collaborate with Oticon project which is part of my thesis work. Under his elaborate cultivation, I am fortunate to publish a paper in ICASSP 2019, which provides a great platform where I could let other researchers know my work and also get to know others' research interests. Also, his rigorous attitude and great passion towards scientific research have set a very good example for me to follow.

Secondly, I would like to give my special thanks to Gaurav Naithani who plays a very important role during my research work. I am very fortunate to learn many technical knowledge from him such as the common Linux command, GPU computing, writing scientific papers and so on so forth. It is him who guides me in my research work and helps me to build confidence.

Also, I would like to give many thanks to the whole staff members in Audio Research Group including Joonas Nikunen, Pasi Pertilä, Mikko Parviainen, Paul Magron, Archontis Politis whom I often trouble a lot for technical problems I met, Sharath Adavanne for his guidance in Advanced Audio Processing course as well as clusters related doubts I had. Additionally, I want to express my gratitude to Soumya Ranjan Tripathy for his generous help in the usage of Pytorch platform, Inkscape, and etc.

Lastly, many thanks should be given to Tampere Univeristy and ITC department for providing such wonderful working environment and great facilities.

# Contents

# 1 Introduction

Single channel speech separation aims to recover the target speech from mixture audio signals [1]. Separating the constituent speakers in the mixture signal consisting of multiple speakers seems easy to human auditory system while challenging for machines to do this task, which is commonly called 'auditory scene analysis' or 'cocktail party problem' [2].

Speech separation has been widely used in many other field such as mobile communications [3], machine translation [4], automatic speech recognition [5], and especially for hearing aids application [6]. For example, in machine translation, given an audio signal which consists of speeches from multiple speakers and background noise, the machine will be able to 'listen', 'translate' correctly and efficiently only if the target speech is not interfered by other speakers' speech and the background noise [4].

There has been many studies to solve the 'cocktail party problem'. Before deep learning approaches, there has been model based approaches [7] and matrix factorization such as NMF [8]. Recently, deep learning methods have been widely used in signal processing related problems including source separation [9, 10]. In [11], authors proposed using deep neural networks to predict the masks in a supervised learning manner. Later, quite a few neural network structures have been studied like convolutional neural networks (CNN), LSTM, and convolutional recurrent neural networks (CRNN) [12]. However, those supervised learning approaches come into problems of speaker dependency. In other words, networks are able to separate mixture signals only if speakers are seen during the training. Unfortunately, it fails to separate when networks meet unseen speakers. For solving the speaker dependency problems, another approach named deep clustering [13] has been proposed recently to achieve speaker independency. Unlike supervised learning approaches predicting the masks, or the target speakers in a supervised manner, deep clustering method generates a higher dimensional embedding vector for each time-frequency bin, then the clustering method like k-means is used to cluster the embedding vectors into clusters. Later, CRNN has been studied [14], and better objective loss function has been also studied [15]. More recently, deep attractor networks has further been studied [16].

However, the deep clustering approach proposed in [13] is not practical for online separation system due to the fact that clusters will not be estimated until the whole sentence completes. Applications like hearing aids [6] and cochlear implants [17] require restrictive latency. It is found that the subjective disturbance is experienced by the listeners (e.g., [18]). When the latency is between 3 to 5 ms, it will be

noticed by hearing-impaired people. However, if the latency is above 10 ms, it will be unbearable [19]. Thus, an online variant deep clustering method is in need.

The contributions of this thesis are listed as follows. First of all, the usage of LSTM networks instead of BLSTM networks allows one to make online processing possible. Secondly, in the offline deep clustering system, normally analysis-synthesis window with the length of 32 ms is employed. However, in low-latency speaker separation system, 8 ms analysis-synthesis window lengths are preferred to ensure the low latency. Lastly, instead of predicting the embedding vectors for the complete sentence, the buffer is proposed to get cluster centers in the beginning of the mixture. Furthermore, estimated masks are obtained by using those cluster centers to assign clusters for the rest of the embedding vectors.

The thesis is organized as follows. Chapter 2 recaps the theoretical background including the audio basics and machine learning basics. In chapter 3, related works are discussed. Chapter 4 describes the proposed algorithms for low-latency deep clustering separation system. Evaluation part is given in the chapter 5. Finally, chapter 6 concludes the whole thesis.

# 2 Theoretical Basics

In this chapter, the general basics are discussed. First section introduces audio basics. In the latter section, basics related to machine learning are discussed.

## 2.1 Audio Basics

In this section, some basic audio knowledge are discussed. In the first subsection, spectral feature used in this thesis by calculating the short-time Fourier transform (STFT) is discussed. In the last subsection, the technique used for source separation (time-frequency (TF) masking) in this thesis is discussed.

### 2.1.1 Short-Time Fourier Transform

In this thesis, the STFT [20] features of signals are used to represent how frequencies in the signal change over time. The convenience of easily inversible and computationally fast of the STFT features makes itself widely used. The calculation of the STFT consists of three parts, segmenting, windowing and calculating discrete Fourier transform (DFT) in each frame [1]. $x(t,n)$ is denoted as the segmented, windowed signal shown as

$$x(t,n) = x(n + n_0 + tM)w_a(n), \tag{2.1}$$

where $t \in [0, T-1]$ denotes the $t^{th}$ frame, $n \in [0, N-1]$ represents the $n^{th}$ sample in a frame, $n_0$ is the $1^{st}$ sample in the $1^{st}$ frame, $M$ is the number of frame advance in samples, and $w_a(n)$ is the analysis window. The DFT is applied to each windowed frame as

$$X(t,f) = \sum_{n=0}^{N-1} x(t,n)e^{-2j\pi nf/F}, f \in [0, F-1], \tag{2.2}$$

where $X(t,f)$ denotes the spectrogram, $F$ is the number of frequency bins and $j$ is the imaginary part.

For example, a signal is given in the Figure 2.1. As the signal is shown in the time domain, it can only be seen roughly how the amplitude changes over time. However, the frequency information inside the signal is not clear. By applying the discrete Fourier transform (DFT) to the signal, the frequency information can be clearly seen as shown in the Figure 2.2. A peak at 20 Hz is shown in the spectrum. However, how the frequency changes over time is not clear yet in the spectrum. By applying the STFT, this feature can be clearly seen as shown in the Figure 2.3. We

can see that the signal has consistent frequency of 20 Hz along all the time.

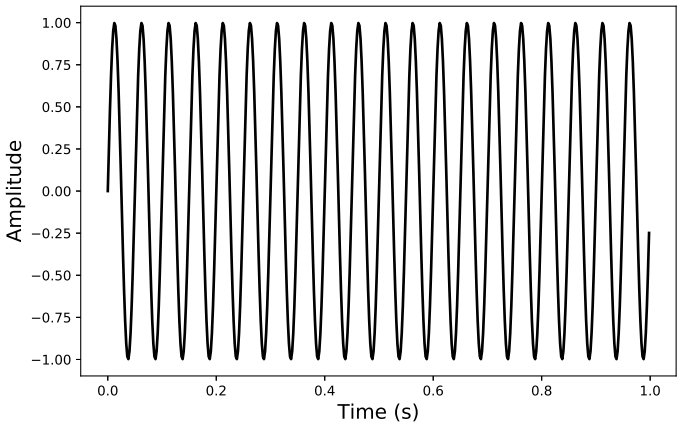


***Figure 2.1*** *An example of a signal in the time domain having only one frequency 20 Hz*

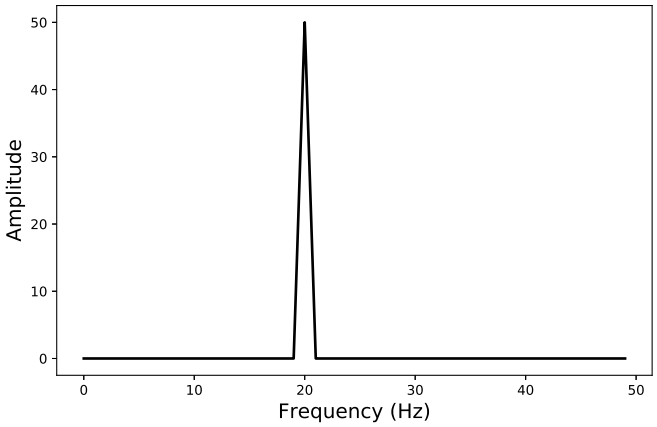


***Figure 2.2*** *DFT features of the signal above in the frequency domain, having one peak at 20 Hz*

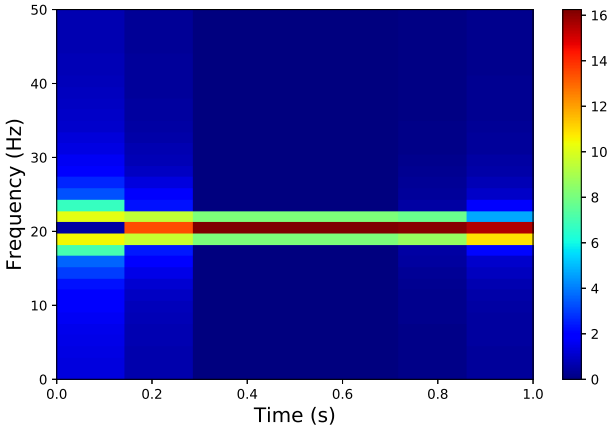


***Figure 2.3*** *The STFT features of the signal above represented in the time-frequency domain. The color depicts the amplitude of the frequency.*

### 2.1.2 Time-Frequency Masking for Source Separation

#### Ideal Binary Mask

As described above, masking is done in the time-frequency domain in order to separate the mixture. Ideal binary mask (IBM) [21] is one of the commonly used masks. An ideal binary mask tells which source dominates in the mixture spectrogram for each time-frequency bin. The $j^{th}$ source magnitude $\hat{X_j}(t, f)$ at time $t$ and frequency $f$ is defined as

$$\hat{X_j} = |X_j(t, f)|, \tag{2.3}$$

where $||$ denotes the magnitude of the spectrogram, $X_j(t, f)$ is calculated as the Eqaution 2.2. The IBM is calculated as

$$M_j(t, f) = \begin{cases} 1, & \text{if} \quad \hat{X_j} > \tau \sum_{k \neq j} \hat{X_k} \\ 0, & \text{otherwise} \end{cases}, \tag{2.4}$$

where $M_j(t, f)$ refers to the $j^{th}$ source mask at time $t$ and frequency $f$, and $\tau$ is the threshold.

For example, mixture spectrogram is shown in the Figure 2.4 (a), spectrogram of source signal 1 is shown in the Figure 2.4 (b), and spectrogram of source signal 2 is displayed in the Figure 2.4 (c). From those two source signals' spectrogram, it can be clearly seen that, at around 1.5 s, higher frequencies are dominated by source 1 and lower frequencies are occupied by source 2, which can also be seen in the binary masks of source signal 1 shown in the Figure 2.4 (d) and source signal 2 shown in the Figure 2.4 (e).

### Ideal Ratio Masks

The value of the binary mask is either 0 or 1, which is quite hard assignment for each time-frequency bin. Thus, a smooth version of the binary mask is ratio mask [22]. Instead of assigning 0 or 1 to each time-frequency bin, ratio mask assigns values between 0 and 1. The IRM is calculated as

$$M_j(t,f) = \left( \frac{\hat{X}_j^p}{\sum_k \hat{X}_k^p} \right)^v, \tag{2.5}$$

where $p$ and $v$ are the parameters for tuning the shape of the ratio mask. Specifically, when $p = 2$ and $v = 1$, it is called the Wiener filter. The same source signals' (refer to Figure 2.4 (b) and (c)) ideal ratio masks are shown in the Figure 2.5 (a) and (b). By comparing Figure 2.4 (d) and (e) with Figure 2.5 (a) and (b), respectively, it can be seen that the IRMs are smoother than IBMs.

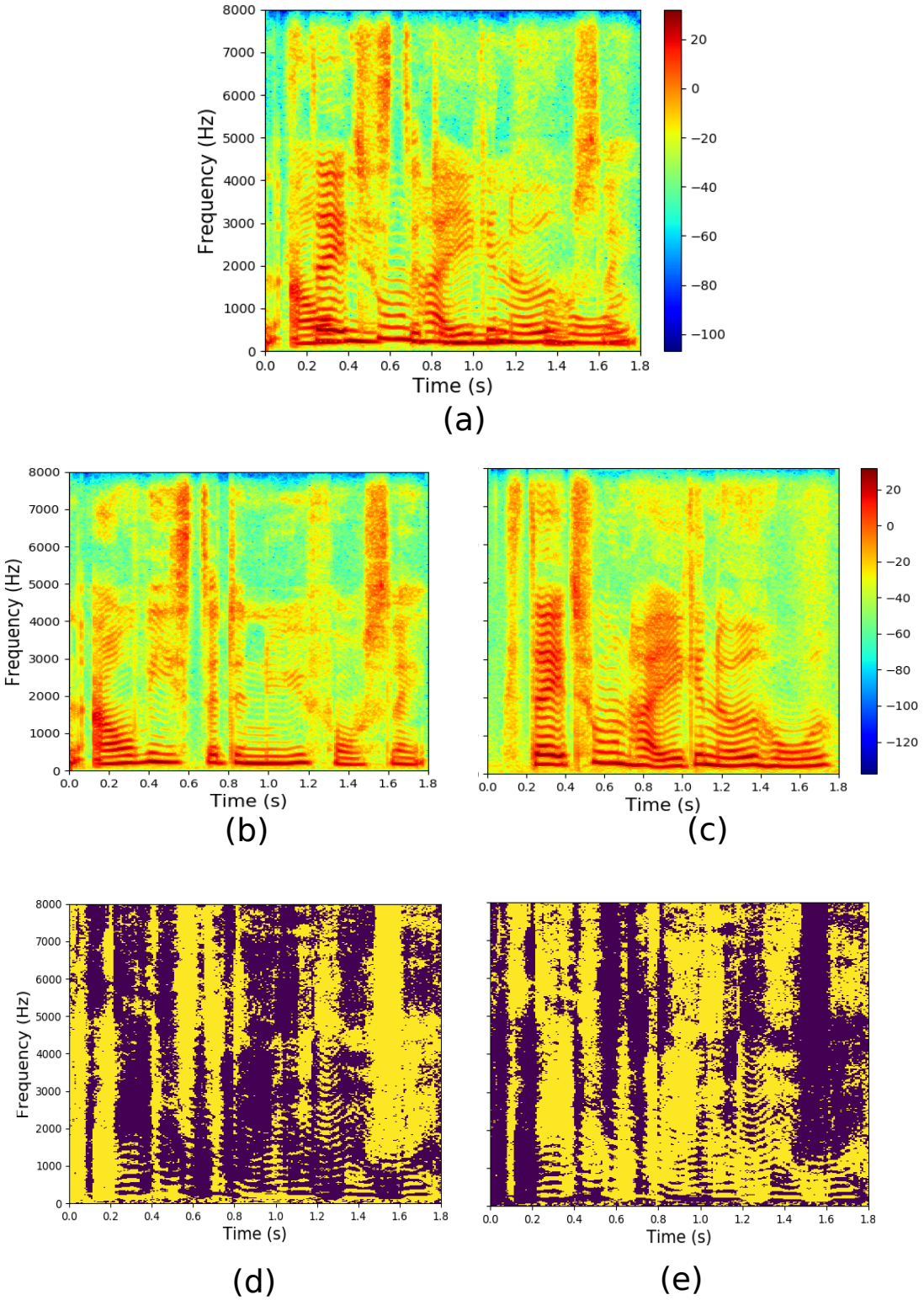


***Figure 2.4*** *Illustration of IBMs. Part (a) shows the spetrogram of the mixture, the spectrograms of source 1 and 2 are shown in the part (b), (c), respectively. Part (d), (e) display the corresponding IBMs.*

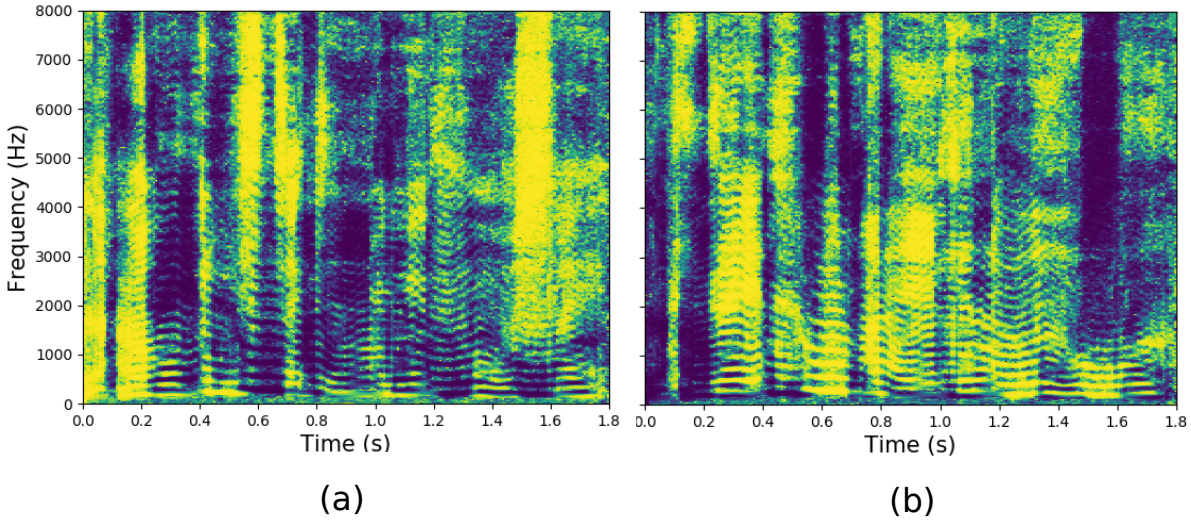


***Figure 2.5*** *Illustration of IRMs. Part (a) shows the IRM for the source signal 1, and part (b) represents the IRM for the source signal 2.*

### The Process of Time-Frequency Masking for Source Separation

The term "source" is used to refer to the clean and single component signal like one speaker's speech or one type of music component (the guitar sound). Acoustic mixtures consist of multiple sources. Hence, source separation aims to recover the constitutive sources from the mixtures. There are instantaneous mixing [23] from the sources shown as

$$x_i(n) = \sum_j s_j(n)a_{ij} + b_i(n), \tag{2.6}$$

where $s_j(n)$ denotes the $j^{th}$ source signal, $a_{ij}$ represents the gain of the $j^{th}$ source signal in the mixture $x_i(n)$, and $b_i(n)$ indicates the noise. There are also convolutional mixing [23] of source signals shown as

$$x_i(n) = \sum_j \sum_\tau s_j(n-\tau)a_{ij}(\tau) + b_i(n), \tag{2.7}$$

where $a_{ij}(\tau)$ refers to the response of the $j^{th}$ source signal to the $i^{th}$ microphone, $b_i(n)$ states the noise, and the mixture is $x_i(n)$.

Mostly, separation is processed through masking in the time-frequency domain instead of the time domain. The reasons are listed as follows, firstly, in the time domain, only how the amplitude changes over time can be visualized and the structure of the sound is not clear. However, in the time-frequency domain, the structure of the natural sound is more clear than in the time domain like where the harmonics are. Secondly, the convenience of taking the STFT to frequency domain and its

inverse short-time Fourier transform (ISTFT) back to time domain makes it handy. Thirdly, the fact that the convolution mixing in the time domain equals to the multiplication in the time-frequency domain makes computation feasible. Lastly, for the separation tasks, the less overlap in the mixture, the easier to separate the sources. In the time domain, signals are more overlapped with each other. However, in the time-frequency domain, the different sources are more sparsely distributed.

Given a mixture signal $x(n)$, firstly, the STFT features $X(t,f)$ are calculated as Equation 2.2. Secondly, mask $M_j(t,f)$ (IBM) for the $j^{th}$ source is calculated using Equation 2.4 during training process. In the testing case, masks are predicted by DNNs. The estimated spectrum $\hat{S}_j(t,f)$ of the $j^{th}$ source follows

$$\hat{S}_j(t,f) = X(t,f)M_j(t,f). \tag{2.8}$$

Further, the inverse DFT (IDFT) is taken as

$$\hat{s}_j(t,n) = \frac{1}{F}\sum_{f=0}^{F-1}\hat{S}_j(t,f)e^{2j\pi nf/F}, n \in [0, N-1], \tag{2.9}$$

where $\hat{s}_j(t,n)$ denotes the time domain signal of the $t^{th}$ frame. Finally, the complete time domain signal $\hat{s}_j(n)$ is overlap-added from the previous windowed frames as

$$\hat{s}_j(n) = \sum_{t=0}^{T-1}\hat{s}_j(t, n-n_0-tM)w_s(n-n_0-tM), \tag{2.10}$$

where $w_s(n)$ denotes the synthesis window.

The diagram of this process is shown in the Figure 2.6.

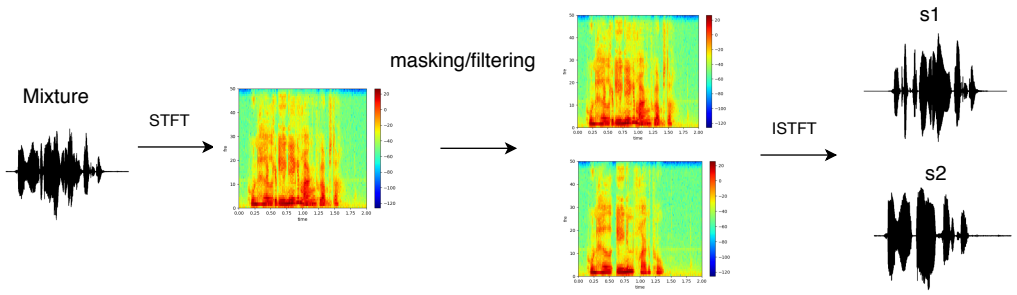


***Figure 2.6*** *Illustration of masking in the time-frequency domain. Spectral features of the mixture signal is calculated by the STFT, filtering process happens afterwards, separated source signals are reconstructed back to time domain by the ISTFT.*

## 2.2 Machine Learning Background

In this chapter, the basics related to machine learning are discussed such as the clustering, recurrent neural networks, and activation functions. Finally, one of the common problems in machine learning (overfitting) is discussed.

### 2.2.1 Clustering Method

Cluster analysis [24] is to group the data into a few clusters with high similarity in the same clusters and high dissimilarity between different clusters as shown in the Figure 2.7. Two lines separate the plane into four parts, one part groups the green data points into one cluster, one part clusters red data points into one cluster, one for black data points and another part separates the yellow data points.

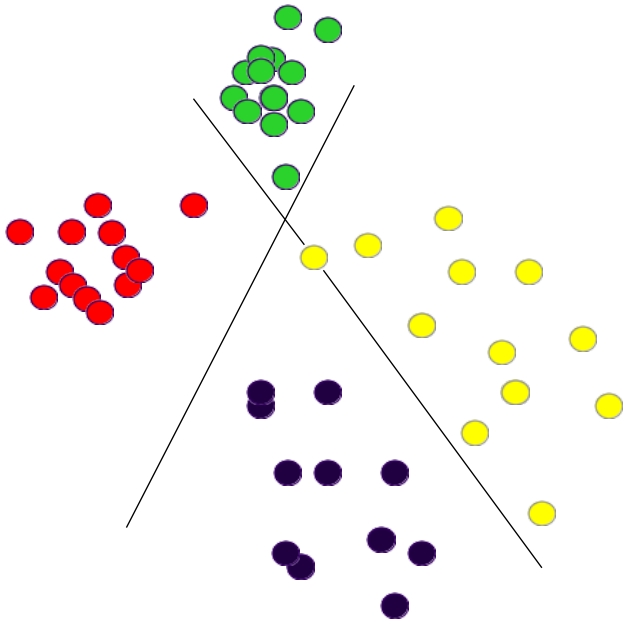

***Figure 2.7*** *Example of clustering. The green dots, red dots, black dots, and the yellow dots represent different groups.*

Clustering is an unsupervised learning approach in machine learning, which does not require the labels for each data. The mostly commonly used clustering method is k-means clustering [25], which groups the data into $k$ clusters and minimizes the distance (squared Euclidean distances) within the same cluster. Given a set of data $Z = \{z_i\}$ where $i = 1, ...m$ and $z_i$ denotes the $i^{th}$ sample, k-means tries to group $m$ observations into $k$ clusters $S = \{s_j\}$ where $j = 1, ...k$, $s_j$ denotes the $j^{th}$ cluster and $k \leq m$. It follows

$$\underset{S}{argmin} \sum_{i=1}^{k} \sum_{Z \in S_i} \|Z - \mu_i\|^2, \tag{2.11}$$

where $\mu_i$ denotes the mean of the cluster $s_i$.

### 2.2.2 Recurrent Neural Networks

Recurrent neural network (RNN) [26] is a type of artificial neural network. What makes a RNN different from feedforward deep neural networks (FDNNs) is that at the time step t, the input consists of both current input and the previous output, which takes the past information into account. Especially, the RNN is widely used in sequential data like speech [27] or robot control [28]. The common structure of a RNN is shown in the Figure 2.8. In the forward pass, the RNN follows

$$\begin{aligned} h_t &= tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh}), \\ y_t &= W_{ho}h_t, \end{aligned} \tag{2.12}$$

where $h_0 = \{0\}$. We denote the weight matrix from the input to the hidden layer as $W_{ih}$, the weight matrix from hidden layer to the next hidden layer $W_{hh}$, and the weight matrix from the last hidden layer to the final output layer $W_{ho}$.

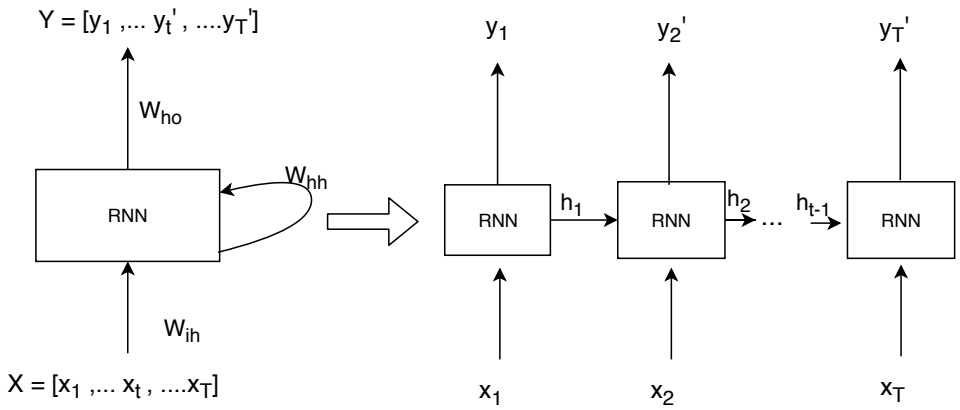


***Figure 2.8*** *Structure of RNN. The input vector is $X$, and output vector $Y$, $W_{ih}$ represents the weights from the input layer to the hidden layer, $W_{hh}$ represents the weights from the hidden layer to the hidden layer, $W_{ho}$ represents the weights from the hidden layer to the output layer.*

#### Long-Short-Term-Memory Networks

RNNs suffer from the vanishing gradients [29], which makes the model stop learning during the training process. Small gradients lead to exponentially shrinkage while large gradient cause exponentially increment. For solving this problem, long-short-term-memory networks (LSTM) have been proposed in [30]. LSTM is similar to the vanilla RNN other than it has three gates inside the hidden cell, one is input gate, one is forget gate [31], and another one is output gate. The input gate controls

what information should be inputted to the cell and what to block out, forget gate forgets the unnecessary information, finally output gate regulates the final output. The illustration how information forwards through LSTM networks is shown as

$$\begin{aligned} i_t &= \sigma(W^{xi}x_t + W^{hi}h_{t-1} + W^{ci}c_{t-1} + b^i), \\ f_t &= \sigma(W^{xf}x_t + W^{hf}h_{t-1} + W^{cf}c_{t-1} + b^f), \\ c_t &= f_t c_{t-1} + i_t tanh(W^{xc}x_t + W^{hc}h_{t-1} + b^c), \\ o_t &= \sigma(W^{xo}x_t + W^{ho}h_{t-1} + W^{co}c_t + b^o), \\ h_t &= o_t tanh(c_t), \end{aligned} \tag{2.13}$$

where $i_t$ denotes the input gate at time $t$, $f_t$ denotes the forget gate at time $t$, $c_t$ denotes the cell state at time $t$, $o_t$ denotes the output gate at time $t$, $h_t$ denotes the final hidden output at time $t$.

### Bidirectional Long-Short-Term-Memory Networks

Bidirectional long-short-term-memory networks (BLSTM) [32] is another type of RNN which is pretty much similar to LSTM other than it has both forward LSTM and backward LSTM. BLSTM network structure is shown as the Figure 2.9.

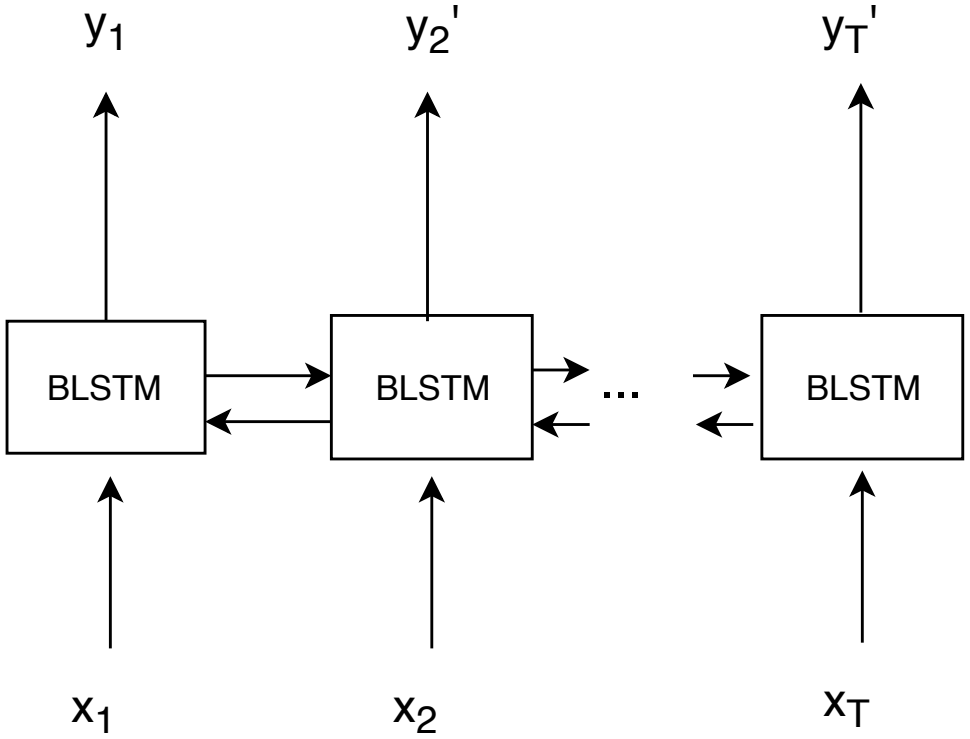


***Figure 2.9*** *BLSTM network structure. The input vector is $X$, and output vector $Y$. The BLSTM block represents the BLSTM cell, the arrow shows how information flows. It has two directions, one forward and one backward direction.*

### 2.2.3 Activation Function

Activation functions build the bridge between one layer to the next layer. It is very critical in machine learning in a sense that activation functions decide which output and how much portion of the output from the last layer should go to the next layer. It functions as a gate. The mostly common used activation functions are Sigmoid function, Hyperbolic Tangent function (tanh), and Rectified Linear units (ReLu).

#### Sigmoid Function

The formula of sigmoid function [33] is shown as

$$f(x) = \frac{1}{1 + e^{-x}}, \quad \text{where} \quad x \in R. \tag{2.14}$$

The visualization of sigmoid function is displayed in the Figure 2.10. It can be seen that the value of this function is between 0 and 1 and 0 centered shown in the orange vertical line.

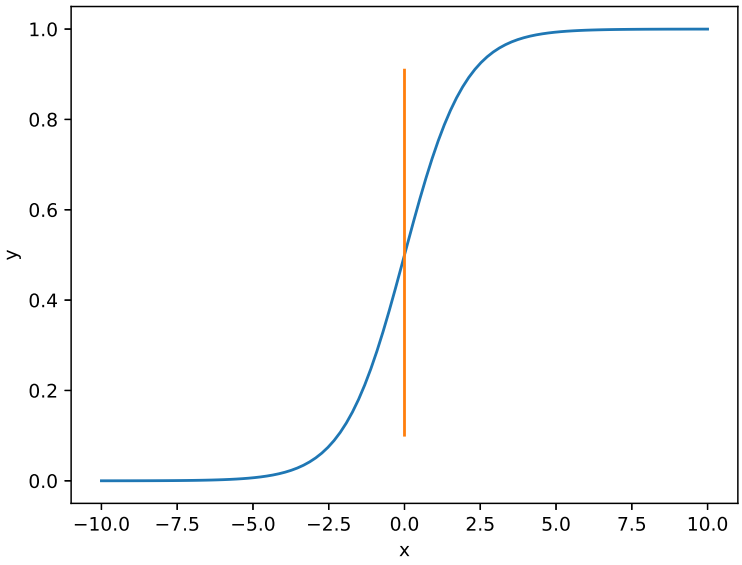


***Figure 2.10*** *Sigmoid function. The values are bounded between 0 and 1.*

### Hyperbolic Tangent Function

This non-linear function tanh [34] is bounded between -1 and 1 as shown in the Figure 2.11 and it is mathematically represented as

$$f(x) = \frac{2}{1 + e^{-2x}} - 1. \tag{2.15}$$

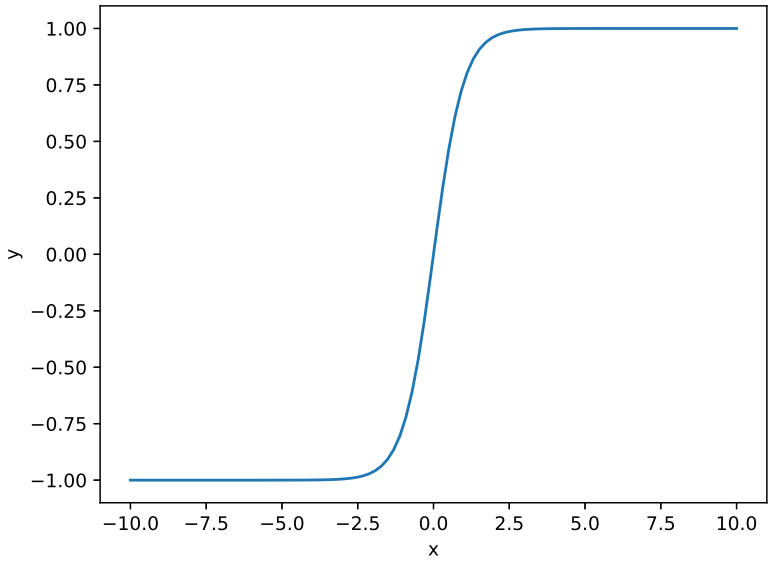


***Figure 2.11*** *tanh function where values are bounded between -1 and 1.*

### Rectified Linear units

ReLu [35] has also been used a lot especially along with convolutional neural networks. Relu is defined as follows

$$f(x) = max(0, x) \tag{2.16}$$

which takes the maximum value between 0 and its value. The vivid Figure 2.12 shows the property of Relu activation function, which only passes positive values.

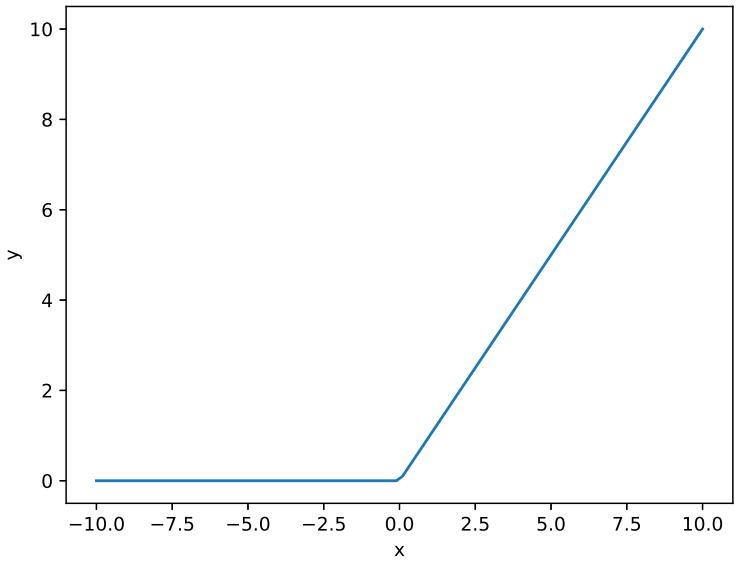


***Figure 2.12*** *Relu function where values are between 0 and infinity.*

### 2.2.4 Overfitting

In deep learning, the most common problem is overfitting problem [36] that the model fits the training data so well that it does not work on the testing data, which indicates that the model does not generalize but overfit the training data. As shown in the Figure 2.13, the red dots shows the distribution of the data. It can be clearly seen that the data are linearly increasing shown as the green line. However, if the model overfits the data, it tries to cover all the data points shown the blue line instead of trying to find the patterns behind the data and generalizing the data.

To avoid the overfitting problem, splitting the data into three parts is needed, normally, 70 percent of data is used for training, 20 percent for cross validation [37], and 10 percent for testing. On top of that, early stopping technique [38] can also be used to avoid overfitting problem. It stops the training process when the validation loss does not decrease for a certain number of the epochs (patience). Another way to solve overfitting problem is that using large amount data if the condition can be satisfied to train the networks so that it can not memorize every single data point instead of generalizing the features behind the data. Also, if there is not enough data, complicated networks should not be used.

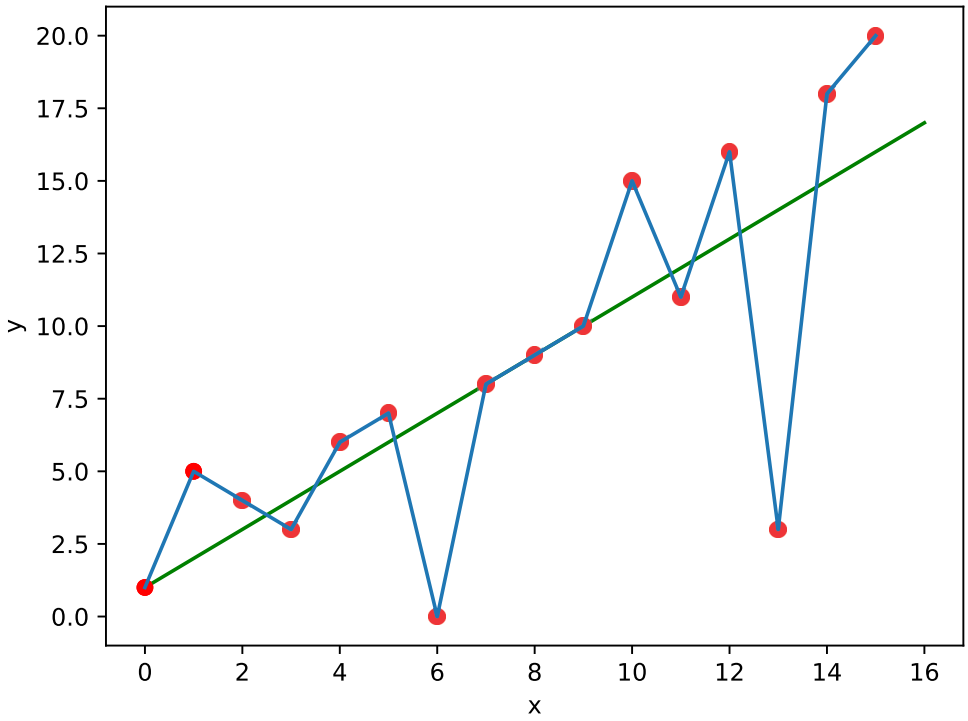


***Figure 2.13*** *Illustration of overfitting problem. The green line shows the general distribution of the red dots input values and the blue line overfits the data.*

# 3 Related Works

In this chapter, the related works are discussed. In the first section, the DNNs based methods for source separation is talked. In the next section, low latency requirements of some application are discussed. In the last section, the recent technique, deep clustering, for source separation is discussed.

## 3.1 DNN Based Methods for Speech Separation

In this subsection, denoising autoencoder for source separation is firstly discussed, followed by the time-frequency masking prediction for separation.

### 3.1.1 Denoising Autoencoders for Source Separation

#### Autoencoder

Autoencoder [39] is a neural network which outputs the same as the input. It consists of two parts, one is the encoder part which maps the input to the code, normally, some lower dimensional features, and another one is decoder part which decodes the code and reconstruct the features back to the input itself. It is an unsupervised learning method due to its fact that it does not require the labels for the input data. The basic autoencoder is shown in the Figure 3.1. We denote that encoder as function $\phi$, and decoder as function $\theta$, such that

$$\begin{aligned} \phi &: X \longrightarrow H \\ \theta &: H \longrightarrow X' \end{aligned} \tag{3.1}$$

and the loss function is calculated in a way such that the distance between the output of the autoencoder network and the input is minimized. The loss function is calculated as follows

$$L = \|X - X'\|^2, \tag{3.2}$$

where $X$ is the input and the $X'$ is the output of the autoencoder system.

One of the main applications of the autoencoder is dimensionality reduction [40], which is essential to many tasks like classification [41]. What we are interested most in autoencoder network is the output of the encoder part which represents some essential lower dimensional features [41] of the input, which is enough to reconstruct back the signal by the decoder. Another application of autoencoder is anomaly detection [42] which trains on the normal data. It learns the most common features of the normal data and reconstruct back by the decoder. When it encounters outliers, the model will fail to reconstruct them back, which can be used to detect the

abnormal data.

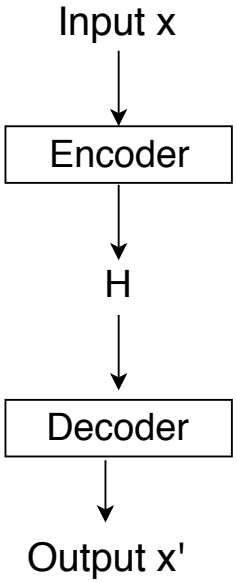


***Figure 3.1*** *Autoencoder. Input $X$ firstly goes to the encoder part to $H$, and the output $X'$ is returned back by the decoder part.*

### Denoising Autoencoder

The mechanism of denoising autoencoder is similar to autoencoder other than different rules of reconstruction [40]. For the autoencoder [41], the input is $X$, and the target is also $X$ while for denosing autoencoder, the input is noisy $X$, and the target output is clean $X$, which can be used as the denosing system.

The denosing autoencoder process is shown in the Figure 3.2. It also has two parts, one is the encoder part, a function of $\phi$, and decoder part, a function of $\theta$ such that

$$\begin{aligned} \phi &: X' \longrightarrow H \\ \theta &: H \longrightarrow X. \end{aligned} \tag{3.3}$$

The loss function is calculated to minimize the output $\hat{X}$ and the clean input $X$, shown as follows

$$L = \|X - \hat{X}\|^2, \tag{3.4}$$

where $X$ is the input and the $X'$ is the output of the denoising autoencoder system.

The simplest denoising autoencoder could be the feedforward neural networks (FNNs). However, for audio source separation problems, the input features normally are the spectrogram with one axis of spectral features and another axis of

the temporal information. FNNs autoencoders have limitations of capturing both dimensional information. Hence, convolutional denoising autoencoders have been proposed in single-channel source separation [43].

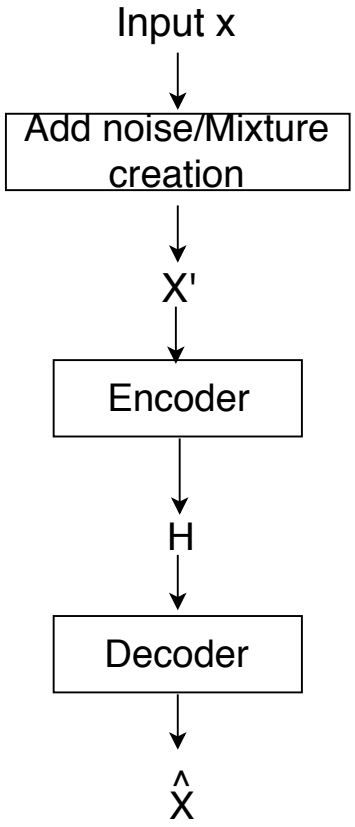


***Figure 3.2*** *Denosing Autoencoder. Input $X$ is firstly added noise to create mixtures $X'$, encoder part encodes the noisy $X'$ to the code $H$, the output $\hat{X}$ of the system is reconstructed back through the decoder part.*

### 3.1.2 Time-Frequency Mask Prediction

There are quite a few approaches to predict the time-frequency masks using DNNs [9]. For example, DNNs can be trained to predict the masks provided that the masks are known beforehand. Then after training, it is assumed that masks are well-learned to filter the mixtures during the testing time. For this approach, DNNs are trained to predict the time-frequency masks. As it is shown in the Figure 3.3, during training procedure, firstly, features are extracted form the time domain mixture signals normally by calculating its STFT. Secondly, features are inputted to the DNNs to predict the masks. Loss is calculated in a way such that the difference between the ground truth masks and the estimated masks is minimized as shown as

$$L = \|M - \hat{M}\|^2 \tag{3.5}$$

where $M$ is the groundtruth masks and $\hat{M}$ denotes that the estimated masks form the DNNs.

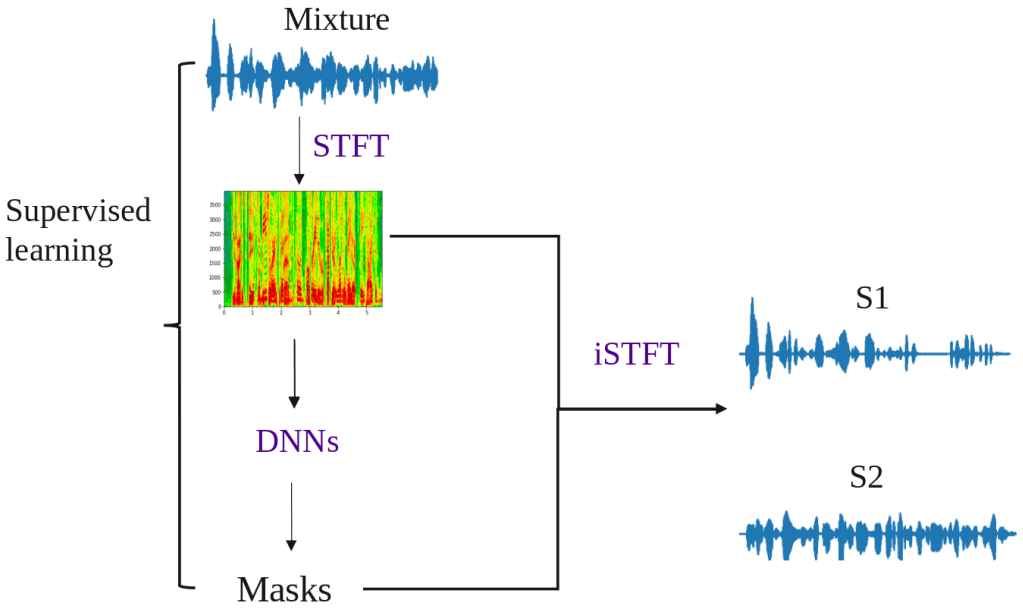


***Figure 3.3*** *Mask prediction using DNNs. Spectral features are calculated by the STFT firstly. DNNs predict the masks in a supervised manner. Separated source signals are reconstructed back to the time domain by using the ISTFT.*

## 3.2 Low-Latency Requirements

Low latency is very important for applications like hearing aids as reported in [6], cochlear implants [17]. Especially, for hearing aids, the latency is quite restrictive for the reason that sound is perceived by the listener both through the direct path also the hearing aids. As shown in the Figure 3.4, the black arrow represents the sound propagates through the direct path and the red arrow shows that the sound is also perceived by the listener through the hearing aid. Suppose the time for the sound reaches the listener through the direct path (black arrow) is $t1$, and the $t2$ denotes the time for the listener to receive the processed sound through the hearing aid. The differences between these two times is denotes as $\Delta t$ shown

$$\Delta t = |t1 - t2|. \tag{3.6}$$

Ideally, the less value of $\Delta t$ is, the better experience that the hearing-impaired listeners have. Hence, the values for the $\Delta t$ have been studied in [19], it is found that the delay of these two time as low as 3 ms to 5 ms can be noticeable, while more than 10 ms, objectionable.

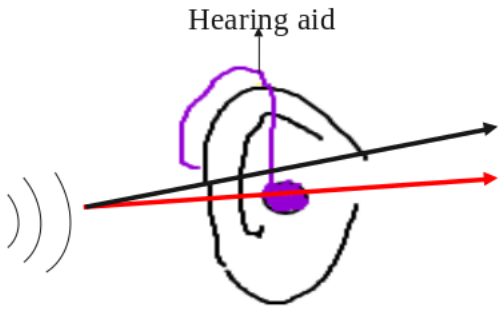


***Figure 3.4*** *Illustration of the hearing aids. The purple color shows the hearing aids device, and the black arrow shows the direct path of the sound. The red line means the sound path through the hearing aid device.*

The latency restrictions of applications like hearing aids have been discussed above. In source separation, what causes latency of the system is the synthesis window length from the algorithmic point of view. As shown in the Figure 3.5, given a mixture in the time domain, features are calculated from the chunk of mixtures windowed by the analysis window, multiplied by the FFT. Spectrum is then inputted to the DNN to get the masks. Estimated source spectrum is obtained by multiplying the mixture spectrum with the masks, which the IFFT later converts back to the time domain. Final separated sources are overlap-added with the previous windowed frames. It is noted that the latency of this scheme depends upon the synthesis window explained in the Equation 2.10.

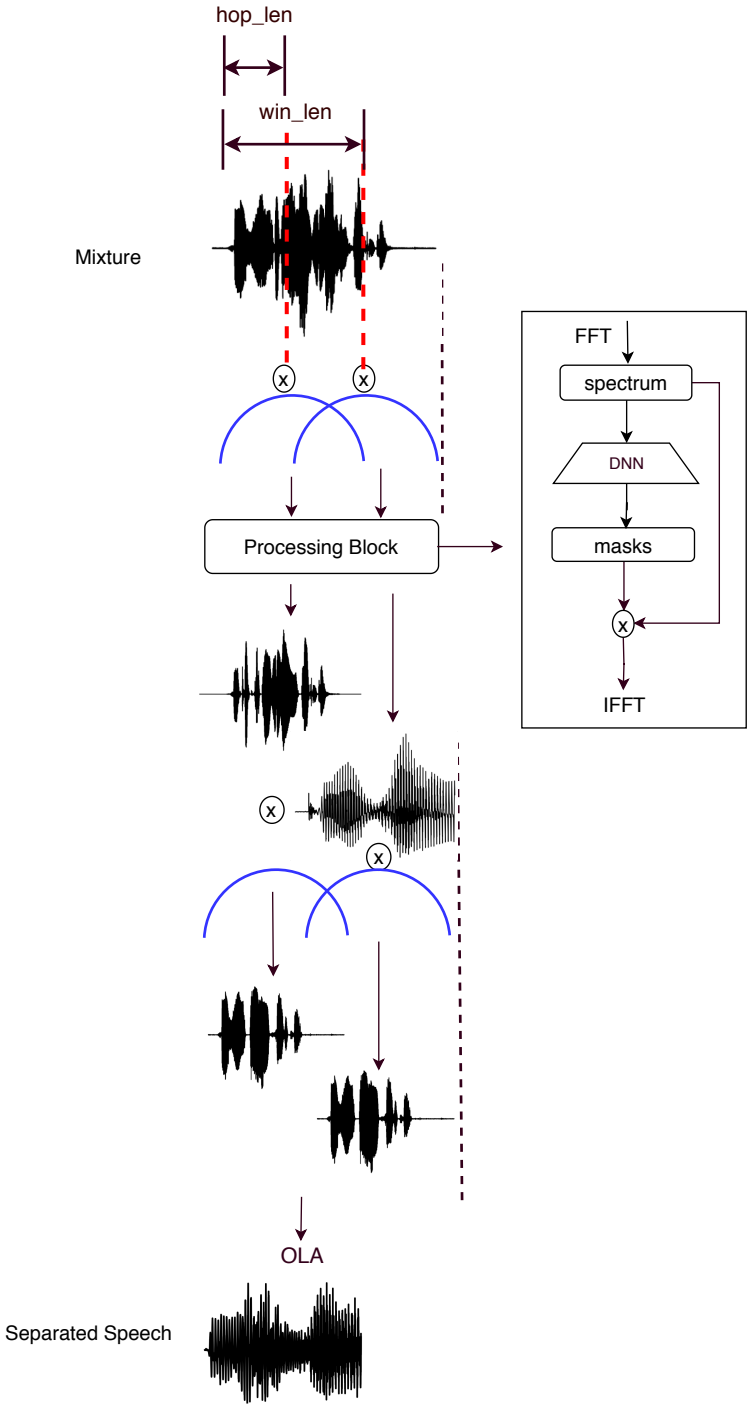


***Figure 3.5*** *Illustration of separation process. Mixture is firstly divided into a sequence of segments by overlapped analysis windows, which are multiplied by the FFT to get the spectrum features. DNNs predict the masks for the mixtures. The estimated source spectrum is obtained by masking the mixture spectrum. The IFFT converts the spectrum back to time domain. Final output is the results of windowed and overlap-added from the previous frame.* $\otimes$ *denotes the element-wise multiplication, and OLA is for overlap-add.*

## 3.3 Offline Deep Clustering for Speech Separation

For source separation, deep learning related methods have shown great improvement. However, there are also many limitation while using these methods based on supervised learning like speaker dependency. For solving the generalization problem, a method named deep clustering has been proposed in [13]. Deep clustering is a combination of supervised learning and unsupervised learning method. For the traditional method based on deep learning, the networks output the time-frequency masks or the factors used for calculating the masks in a supervised learning manner. While for deep clustering method, it generates a high dimensional embedding vector for each time frequency bins in a supervised learning manner and then a clustering method is used to cluster those embedding vectors in order to get the time frequency masks in a unsupervised way. Those embedding vectors are trained in a such way that the distance for the same speaker is minimized while maximized for the different speakers.

Given a mixture signal in the time domain $x(n)$, firstly, features are extracted by calculating the magnitude of its STFT. Secondly, features are inputted into BLSTM networks to get embedding matrix $V$. During the training phase, the loss is calculated in a way such that the difference between the embedding affinity matrix $VV^{\mathsf{T}}$and the ideal binary affinity matrix $YY^{\mathsf{T}}$ is minimized shown as

$$L = \|VV^{\mathsf{T}} - YY^{\mathsf{T}}\|^2. \tag{3.7}$$

As shown in the Equation 3.7, $\mathsf{T}$ denotes the transpose of the matrix. $V$ is the embedding matrix with the dimension of $(T * F, D)$, where $T$ denotes the time with units of $s$, $F$ represents the frequency with the unit of $Hz$ and $D$ is the dimension of the embedding (here D is 40). $Y$ is the IBMs with dimension of $(T * F, C)$, where C denotes the number of the speakers in the mixtures (here 2 is used). If the loss calculation is based on the Equation 3.7, the computation would be costly as the dimensions of $VV^{\mathsf{T}}$ and $YY^{\mathsf{T}}$ are quite high which makes training process difficult. Hence, a better representation of the loss function [13] is

$$L = \|V^T V\|^2 - 2\|V^T Y\|^2 + \|Y^T Y\|^2. \tag{3.8}$$

During the loss calculation as only those bins with magnitude greater than the VAD threshold (here, we set the threshold to 40 dB with respect to the maximum magnitude), voice active detection (VAD) technique is used to mask out those inactive time-frequency bins. In this way, the computation burden is decreased again for accelerating the training process. The flowchart of deep clustering method has been shown in the Figure 3.6 and Figure 3.7.

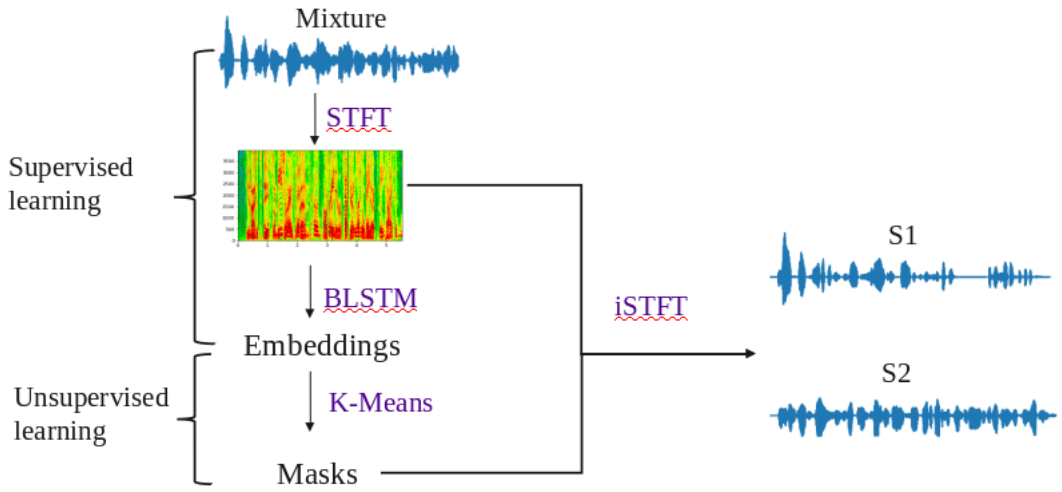


***Figure 3.6*** *Framework of Deep Clustering method. Spectral features of the mixture is calculated by the STFT are inputted into the BLSTM networks. Embedding vectors for each time-frequency bins are outputted from the networks. Networks are trained in a supervised manner. Clustering methods are used to cluster the embedding vectors into masks in an unsupervised manner. Finally, the separated source signals are converted back to the time domain by the ISTFT.*

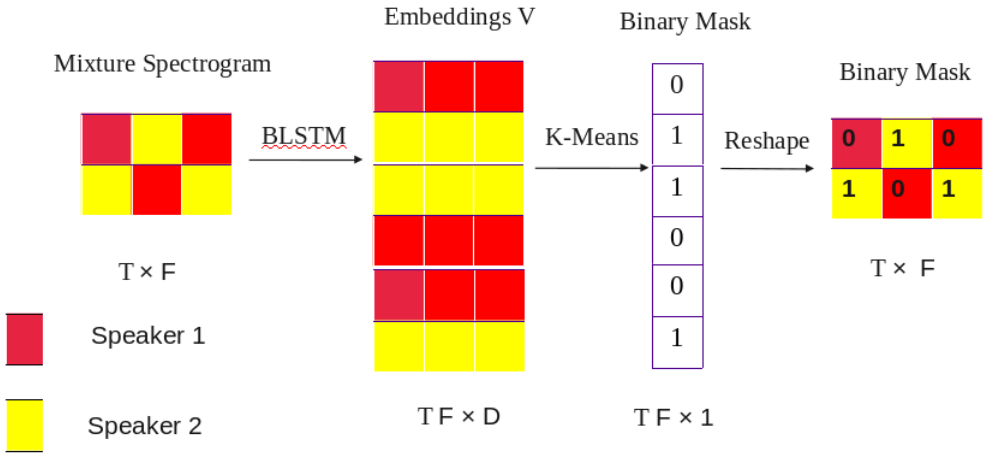


***Figure 3.7*** *Illustration of Deep Clustering method. The red blocks in the time-frequency bins represent the bins where the speaker 1 is dominant, and the yellow blocks shows the bins where the speaker 2 is dominant.*

In the testing phase, given a mixture in the time domain $x(n)$, firstly, features are extracted by calculating the magnitude of its STFT as the training process, then features are inputted to the well-trained deep clustering networks to get the embedding vectors. VAD is also used to mask out those inactive time-frequency bins. Finally, k-means clustering method is applied to the embedding vectors by assigning value 1 to one cluster and 0 to another cluster. In this way, the estimated binary masks are obtained. After obtaining the time-frequency masks, those masks are applied to the mixture spectrogram and the ISTFT is used to reconstruct back to the time domain signals.

# 4 Low-Latency Deep Clustering for Speech Separation

Deep clustering method has shown great improvement for source separation. However, the usage of BLSTM networks and larger window length 32 ms make it difficult for applications like hearing aids. Thus, a low-latency version of deep clustering method is proposed in this thesis. In order to make deep clustering method cooperate with low latency, three main changes [44] have been done explained as follows.

Firstly, the networks should allow online processing by the usage of LSTM networks instead of BLSTM networks shown in the Figure 4.1. Due to the fact that BLSTM networks consist of two parts, one is forward LSTM and another one is backward LSTM. One has to wait until the whole utterance finishes, which makes online processing not practical. Secondly, shorter window length is required as low as 10 ms, here we are using 8 ms window length shown in the Figure 4.2.

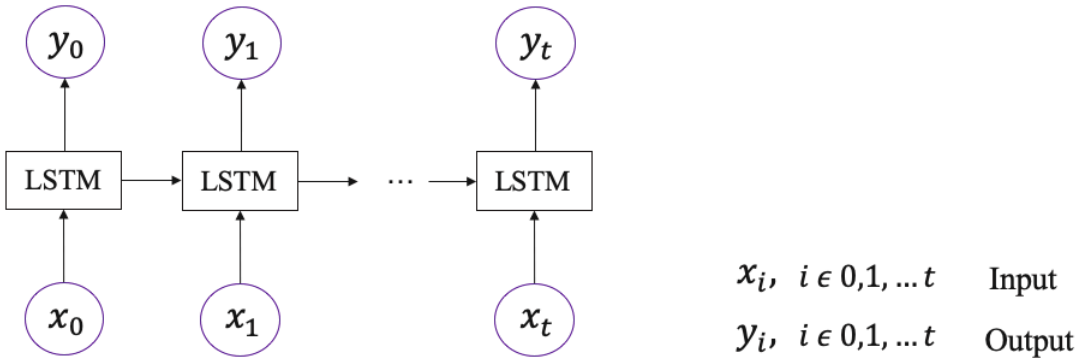


***Figure 4.1*** *LSTM networks show how the information flows through it.*

Thirdly, a speaker model in short time is also needed. For offline deep clustering method, it outputs the embedding vectors for the complete utterance, then a clustering method is applied to the embedding vectors and get the time-frequency masks by assigning different clusters. In order to make this method cooperate with low latency algorithm, here we proposed to get the embedding vectors only from the beginning of the mixture referred as buffer length, then clustering method like k-means is used to cluster those embedding vectors from the buffer duration to the cluster centers. Furthermore, those cluster centers are used to assign the clusters for the rest of the mixture in order to get the time-frequency masks.

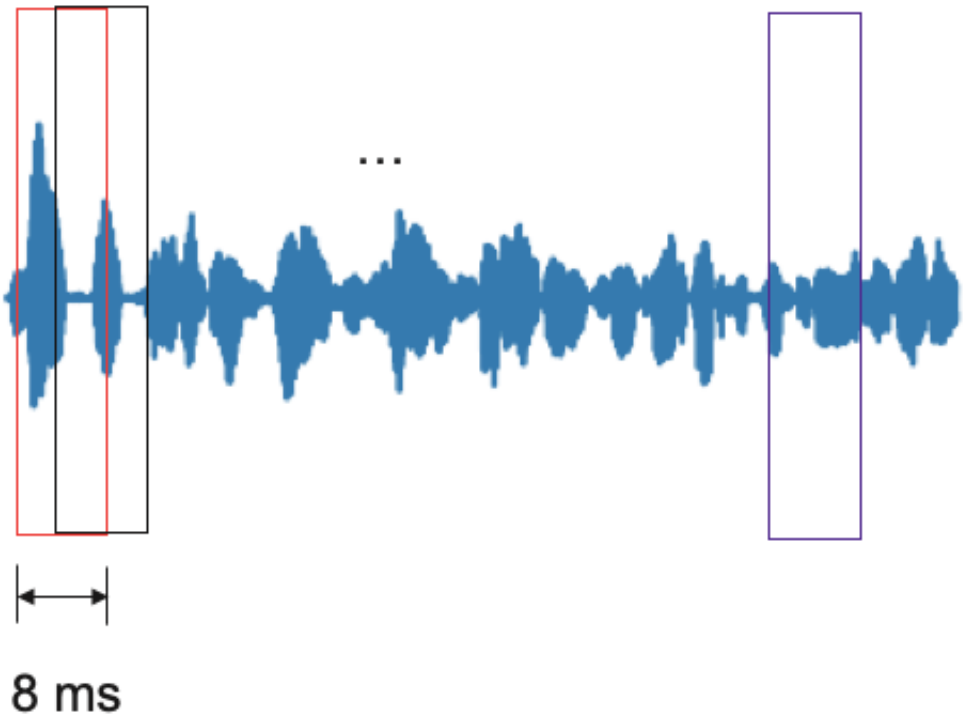


***Figure 4.2*** *The first block represents the first frame which is 8ms, and the second block having the same length 8 ms overlaps with the first block with 4 ms.*

Given a mixture $x(n)$ in time domain, firstly, the As shown in the Figure 4.3,features are extracted from the beginning of the mixture, for example, the first 0.3 s, by calculating the magnitude of its STFT. Secondly, features from the buffer length are inputted to the well-trained networks and output the embedding vectors. Thirdly, k-means is used to cluster the embedding vectors to get the cluster centers as shown in the Figure 4.3, the yellow group represents one cluster and the black cross is the center of the yellow group. Similarly, the red group displays another cluster and the green cross is the cluster center of the red group. Finally, for the rest of the mixture, features are extracted by calculating the magnitude of its STFT and inputted to the networks to get the embedding vectors as how it is done for the buffer length. Those centers from the buffer length are used to assign the clusters for the rest of the embedding vectors and get the estimated time-frequency masks.

It should be noted that in order to get the reasonable cluster centers from the beginning of the mixture, both speakers should be active during the buffer length, which is not the always case in practical. Thus, silence in each mixture is removed in order to cooperate with buffer length idea, which indicates that there will be more overlapped signals and make the separation challenging. Additionally, in order to keep the testing material the same for the different buffer length, it is proposed by using clustering utterance from the same speaker pair as the testing utterance to

get the cluster centers and centers are used to predict the time-frequency masks for the testing utterance. The details of those method are explained in the section 5. The flowchart of online deep clustering method for speech separation is shown in the Figure 4.4.

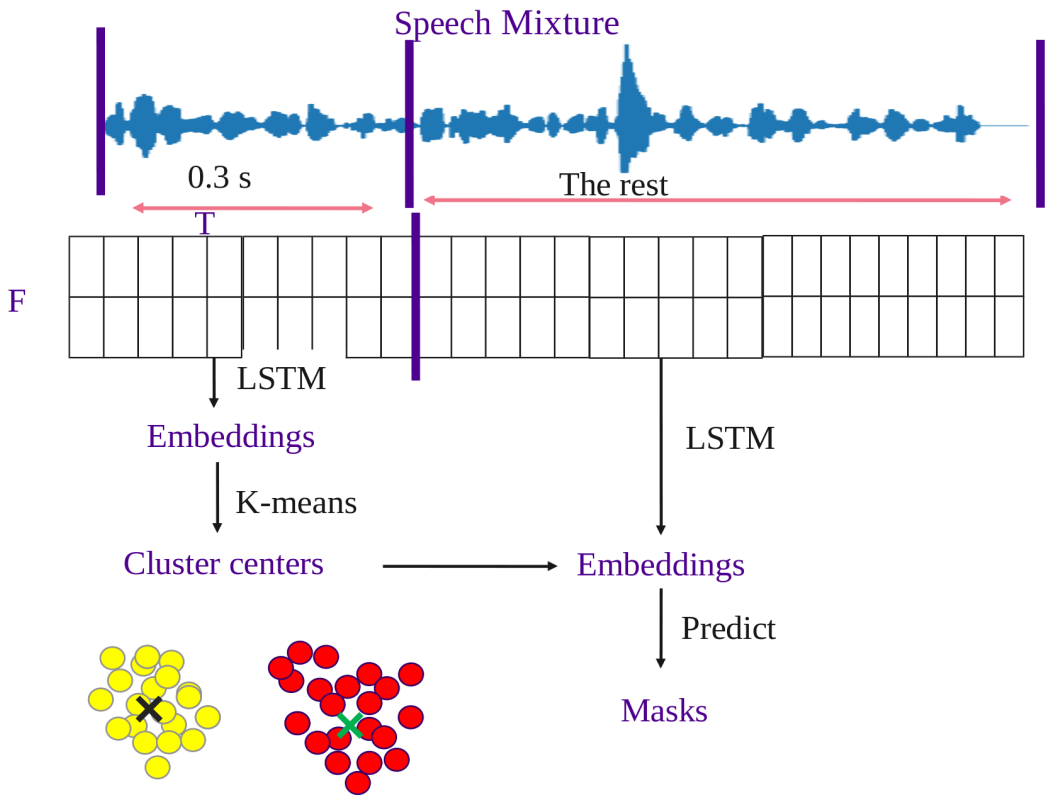


***Figure 4.3*** *Illustration of low-latency deep clustering method. The first 0.3s in the mixture are firstly extracted features by the STFT. LSTM networks output the embedding vectors. K-means clusters the embeddings into two clusters. Centers from those two clusters are used to predict the masks for the rest of the mixture.*

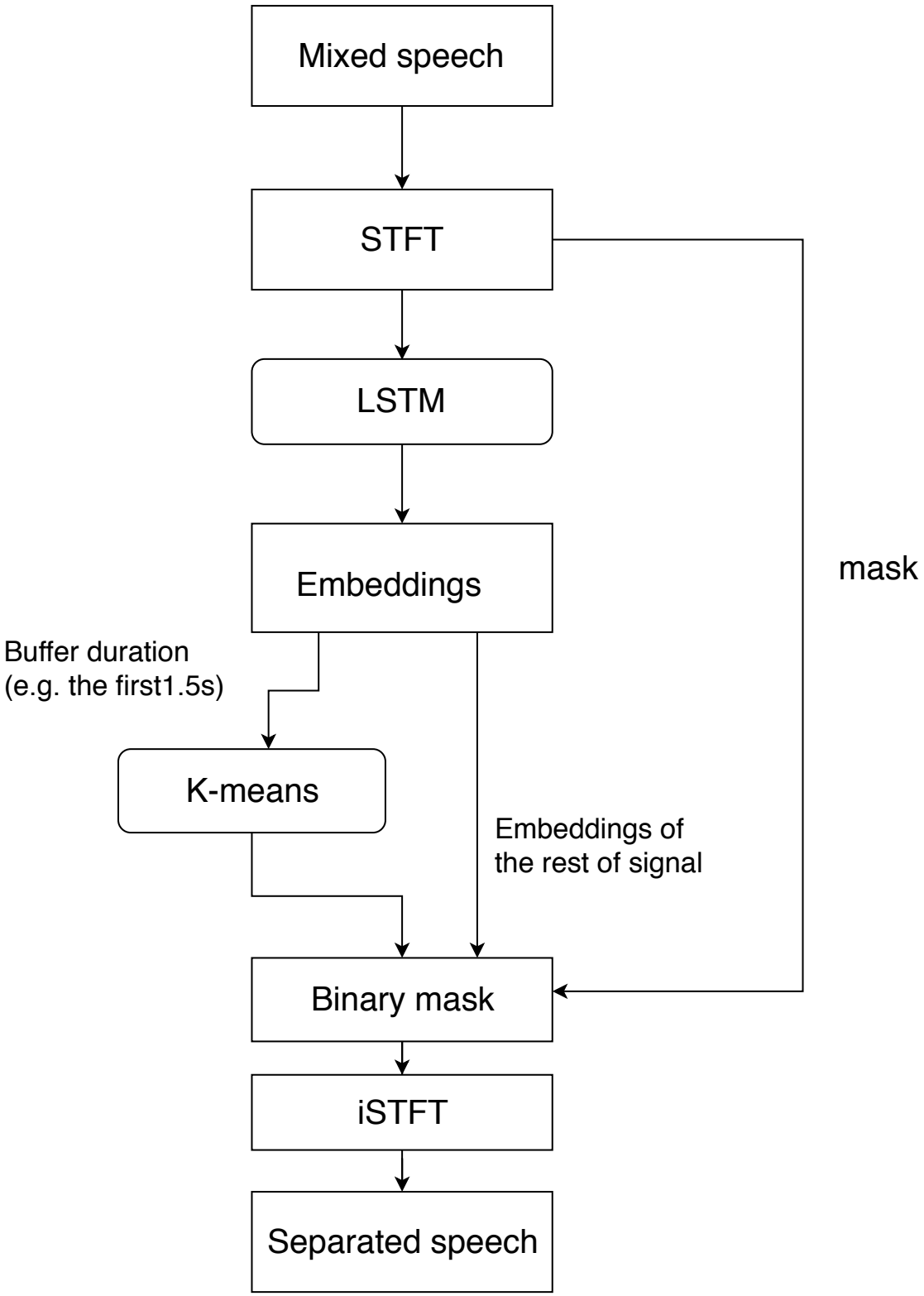


***Figure 4.4*** *The block diagram of the proposed low-latency deep clustering method.*

# 5 Evaluation

In this chapter, low-latency deep clustering method is evaluated. In the section, dataset used in this thesis is introduced. In the next section, the metrics for evaluation are discussed. The experiment setup is discussed in the third section. In the last section, the results are displayed.

## 5.1 Data

In this section, it consists of three subsections, the first one tells about the structure of the dataset. In the next subsection, preparing the data is introduced. The creation of the data is discussed in the last subsection.

### 5.1.1 Dataset

The dataset used to evaluate this separation system is Wall Street Journal Corpus (wsj0) [45]. Data inside the si_tr_s folder are used to develop the system and the data inside si_dt_05, si_et_05 folders are used to evaluate the system. For the training dataset mentioned above, there are altogether 101 speakers including 50 male speakers and 51 female speakers. For the testing dataset, there are 18 speakers including 10 male speakers and 8 female speakers. One should note that, those 18 speakers used for testing the system are not seen during the developing the system, which is meant to test if the separation system is speaker-independent or not.

### 5.1.2 Data Preparation

Firstly, The wsj0 dataset is NIST SPHERE format-wv1 format, for the later convenience of the usage, all the data are firstly converted to WAV format by using toolbox sph2pipe. Secondly, the data is resampled to 8 kHz from 16 kHz for the purpose of reducing the computational burden.

### 5.1.3 Data Creation

The script used for generating 2-speaker mixtures is the same as in [13] for fair comparison. For the training data, 20,000 mixtures are randomly selected from different speakers in *si_tr_s* folder containing around 30 hours audio material. Similarly, for the cross validation data, 5,000 mixtures are created from the same speakers as the training dataset containing around 7.5 hours audio material. Finally, for the testing data, 3,000 mixtures are created from *si_dt_*05, *si_et_*05 folders containing 18 speakers which are different from the training dataset lasting 5 hours.

On top of that, another form of the same testing data is created by grouping the 3,000 mixtures into 306 groups according to the speaker pair information. The latter version of testing data is used to evaluate the low-latency deep clustering system. Additionally, in order to investigate the effect of different buffer duration, silence in the beginning of the cluster utterances is removed ensuring that both speakers are active during the buffer. Furthermore, in this way, two cluster centers can be formed so that they can be used to assign clusters for rest of each time-frequency bin in the rest of the mixture.

It is noted that, firstly, before mixing them together, all the data have been normalized using a function called 'actilev' in voicebox, MATLAB. Secondly, a random positive value ranging from 0 to 2.5 of gain (dB) is added to speaker 1. Then the corresponding negative value of gain is added to another speaker to form a mixture. This ensures mixtures are created at the range of signal-to-noise ratio (SNR) from -2.5 dB to 2.5 dB.

Additionally, after creating the mixture, all the sources and the mixtures are scaled by the same scaling factor, which ensures that the absolute value of the signal is no more than 1. Lastly, there are two ways of trimming the signals to the same length, one is that keeping the maximum duration of the signals and pad zeros to those signals which are not. Another way is to keep the minimum duration of the signal and trim the rest to the same length. In this thesis, the minimum length method is taken. One should note that in this way, there will be more overlaps compared to taking the maximum length. The following Figure 5.1 has shown the details of how the data is created.

## 5.2 Evaluation Metrics

For evaluating the performance of the system, BSS-EVAL toolbox [46] is used. It consists of three metrics which are source-to-distortion ratio (SDR), source-to-interference ratio (SIR), source-to-artifact ratio (SAR).

SDR is defined as

$$SDR := 10 * \log 10 \frac{||S_{target}||^2}{||e_{interf} + e_{noise} + e_{artif}||^2}. \tag{5.1}$$

SIR is defined as

$$SIR := 10 * \log 10 \frac{||S_{target}||^2}{||e_{interf}||^2}. \tag{5.2}$$

SAR is defined as

$$SAR := 10 * \log 10 \frac{||S_{target} + e_{interf} + e_{noise}||^2}{||e_{artif}||^2}. \tag{5.3}$$

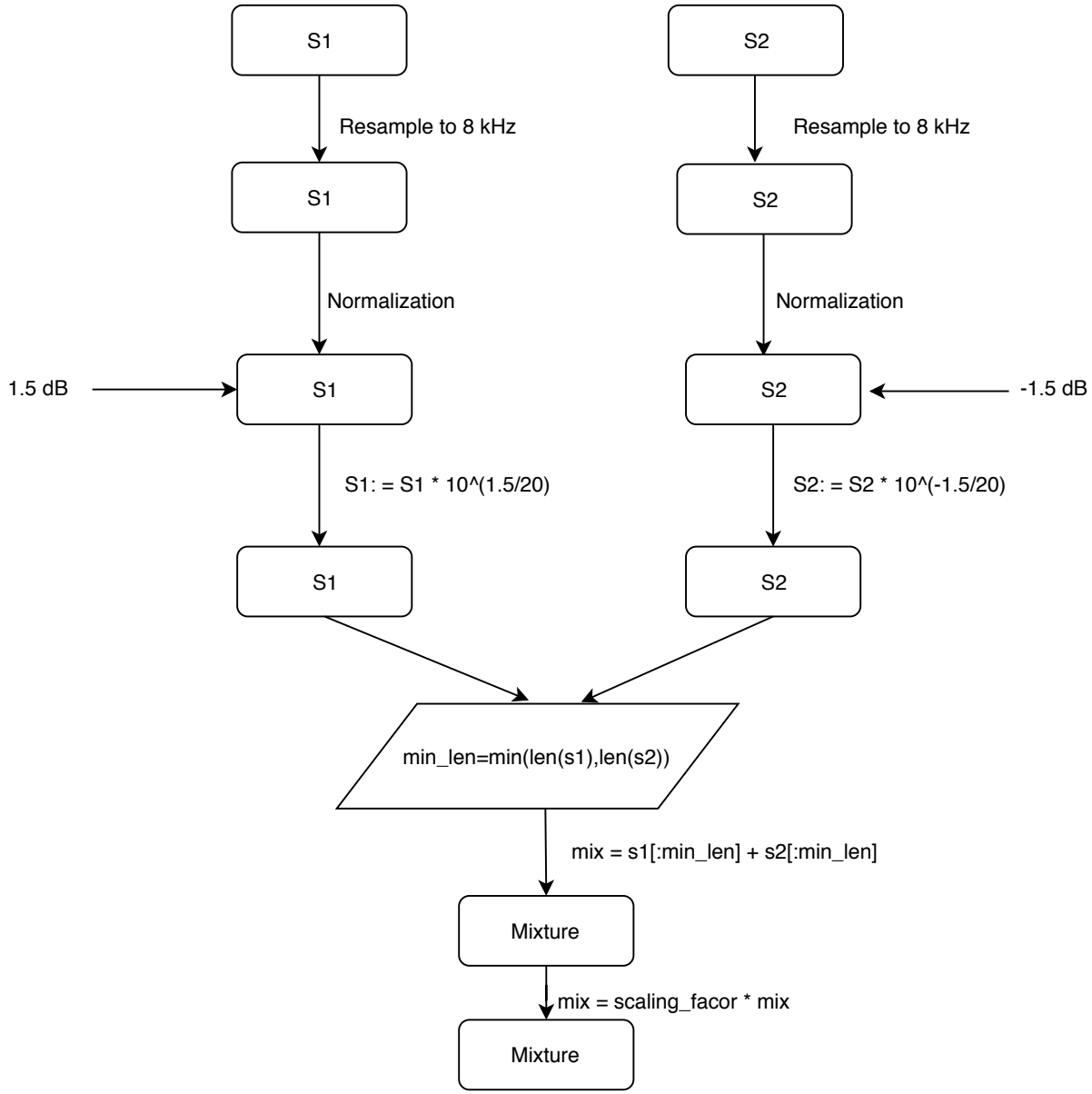


***Figure 5.1*** *The diagram of data creation. All data are firstly resampled to 8 kHz and normalized. Random number of SNR varying from -2.5 to 2.5 dB is added to the mixtures. Mixture is created from two different sources with the length of the minimum over two sources.*

From the equations above, it is clearly seen that the higher value these three metrics are, the better separation performance of the system has. In order to evaluate the separation system properly, a baseline metric which is the initial SDR (0.1dB) showing the result without any separation system.

## 5.3 Experiment Setup

In this thesis, three experiments have been conducted in order to analyze 1) the effect of different neural networks which are BLSTM vs LSTM for the offline deep clustering system, 2) the usage of the different window length, which are $32ms$ and $8ms$ window length for the offline deep clustering system, 3) the effect of different buffer duration for low-latency deep clustering system.

For the experiment setup 1) BLSTM networks, firstly, features are extracted from time domain signals by computing the STFT using librosa library [47]. Hanning window function is used and the window length for this experiment is $32ms$, which has 256 samples. Hop length having $8ms$, 64 samples, is used. After extracting its STFT, a decibel version is computed as

$$features := 20 * \log 10|STFT(x)|. \tag{5.4}$$

Secondly, voice active detection (VAD) technique is used during calculating the loss function, which ensures the deep clustering system not to assign the embedding vectors for those inactive time-frequency bins. VAD is computed on the base of equation above as follows

$$VAD := feature > (max(feature) - Threshold), \tag{5.5}$$

here, threshold is set to 40 dB. Thirdly, the normalized features are sent to the neural networks by subtracting its mean and dividing its standard deviation as follows

$$normalized\_features := \frac{features - mean(features)}{std(features)}. \tag{5.6}$$

Finally, Keras [48] framework and tensorflow [49] are used for implementing this system. However, a pytorch version [50] of this system is also implemented. Here, we mainly talk about the keras version. The network consists of 4 layers BLSTM with 600 units in each layer, followed by a time-distributed Dense layer having the number of the units which is the product of embedding dimension, here, 40 is used, and the number of frequency bins, 129 is used. Hyperbolic tangent (tanh) activation function in this dense layer is used. Additionally, L2 normalization is applied to each embedding vector for the purpose of making each embedding vector to unit norm. Furthermore, sequence length of 100 is set up. During training, 100 epochs are set to train the networks, however, in order to avoid overfitting problems, early stopping [38] technique is utilized by monitoring the validation loss. The patience is set to 30 epochs, which indicates that if the validation loss does not decrease for consecutive 30 epochs, then the training process will be stopped. The networks

structure, number of the trainable parameters have been shown in Table 5.1.

***Table 5.1*** *network structure and parameters*

| **Layers (Type)** | **Output Shape** | **Param #** |
|---|---|---|
| InputLayer | (None,100,129) | 0 |
| Bidirectional LSTM 1 | (None,100,1200) | 3504000 |
| Bidirectional LSTM 2 | (None,100,1200) | 8644800 |
| Bidirectional LSTM 3 | (None,100,1200) | 8644800 |
| Bidirectional LSTM 4 | (None,100,1200) | 8644800 |
| TimeDistributed Dense 4 | (None,100,5160) | 6197160 |
| Total Params: 35,635,560 | | |
| Trainable Params: 35,635,560 | | |
| Non-trainable Params:0 | | |

From the Table 5.1, we can seen that the networks are quite heavy and there are 35,635,560 parameters to be trained. After each epoch, the training loss and the validation loss are saved to find the best model which minimize the validation loss. The following Figure 5.2 depicts how both training loss and validation loss go with the epochs. From the loss shown in the Figure 5.2, we can see that early stopping technique is used during training because it stopped at around 58th epoch instead of waiting up to 100 epochs. On top of that, we can also see that both training and validation loss converge at around 10th epoch.

In order to analyze the effect of factor 1), another the experiment setup is implemented. The only difference from the setup 1) is that the networks consist of 4 layers of LSTM instead of BLSTM. Other parameters are set to the same as the first setup. Factor 2) is analyzed by another experiment. On top of the second experiment setup, a shorter window length having 8 ms, 64 samples, 4 ms hop length, 32 samples, is used while extracting the features from the time domain signals. Additionally, the sequence length is set to 200, which makes $200 * 4 = 800ms$. The reason for doubling the sequence length is because in the setup 2 , $100 * 8 = 800ms$ is inputted to the network during training. In this way, factor 2) can be compared precisely and fairly. Other parameters are kept the same as the setup 2.

For analyzing factor 1) and 2), same testing setup is used. In the test case, given a mixture, firstly, features and the VAD are extracted as the way how training process

does, and then inputted to the well-trained networks to get embedding vectors. Then the embedding vectors are multiplied by its VAD so that those inactive time-frequency bins are not assigned any embedding vectors. Furthermore, k-means in scikit-learn [51] is used to cluster the embedding matrix to 2 clusters. The estimated masks are obtained by assigning one cluster to 1 and the other cluster to 0. Finally, masks are applied to the spectrogram of the mixtures with its own phase information and time domain signals are reconstructed back using the ISTFT in librosa [47].

For the factor 3), it has the same setup as the third setup during training, however, it has different testing setup. What differs from the offline testing setup most is that k-means clustering method is applied to the first few seconds in the beginning of each mixture (buffer) to get clusters centers instead of clustering on the complete signal and those centers are used to assign the clusters for the rest of the mixture. As it is described in the data section, for testing the online deep clustering system, 306 groups of data are formed according to the information of the speaker pair. While testing, for each testing utterance, a clustering utterance is randomly selected from the same group, as it is shown in Figure 5.3, and the silence in the mixture signal to trimmed so that 2 clusters can be formed during the buffer, as shown in Figure 5.4. Those centers are used to assign the clusters for the the testing utterance. In this way, the effect of the different buffer length can be precisely investigated. The detailed parameters of these three networks can be seen from Table 5.2.

***Table 5.2*** *networks and feature parameters for offline and online deep clustering experiments.*

| | **BLSTM** | **LSTM** | **low-latency DC** |
|---|---|---|---|
| Window length | 32 ms | 32 ms | 8 ms |
| Hop length | 8 ms | 8 ms | 4 ms |
| Sequence length | 100 | 100 | 200 |
| Window function | | Hanning | |
| Sampling frequency | | 8 kHz | |
| FFT size | | 256 | |
| Number of layers | | 4 | |
| Number of units in each layer | | 600 | |
| Embedding dimension | | 40 | |

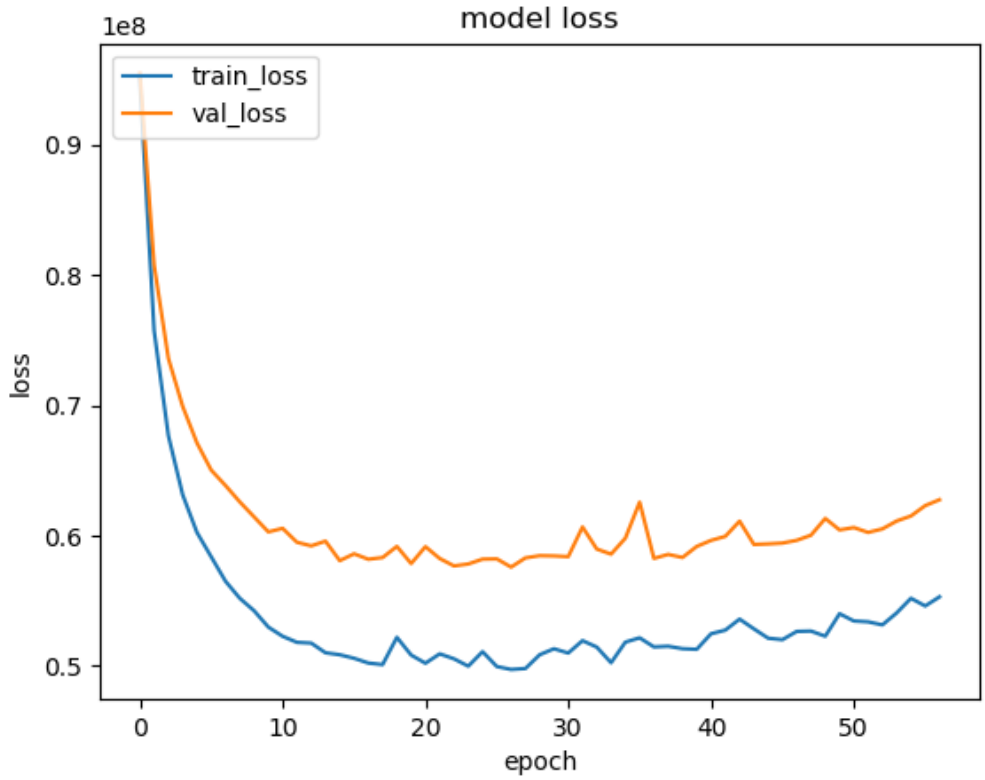


***Figure 5.2*** *The loss of the BLSTM networks, where the orange line depicts the validation loss and the blue line shows the training loss.*

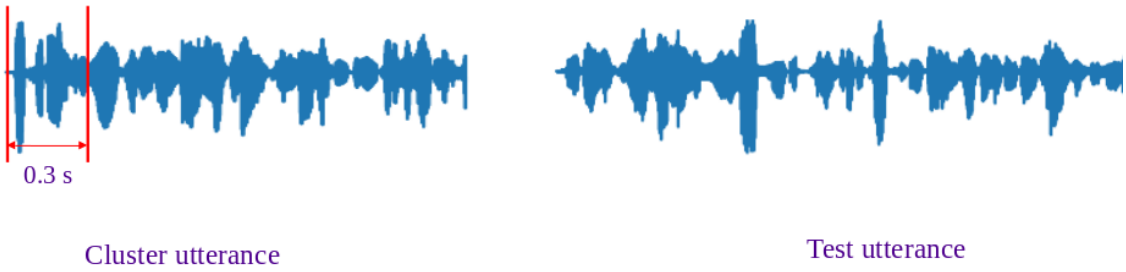


***Figure 5.3*** *Online testing setup explanation*

## 5.4 Results and Discussion

The average of SDR, SIR, and SAR are calculated from the 3,000 test mixtures. The results are seen from the Table 5.3. From the first line of the Table 5.3 which corresponds to experiment setup 1, it can be seen that the highest performance is achieved with 7.9 dB in SDR, 15.6 dB in SIR, and 9.2 dB in SAR. The violin plots are seen from the Figure 5.7. From the second line of the Table 5.3 regarding to experiment setup 2, it can be concluded that by only changing the network from BLSTM to LSTM, the performance dropped 1 dB in SDR.

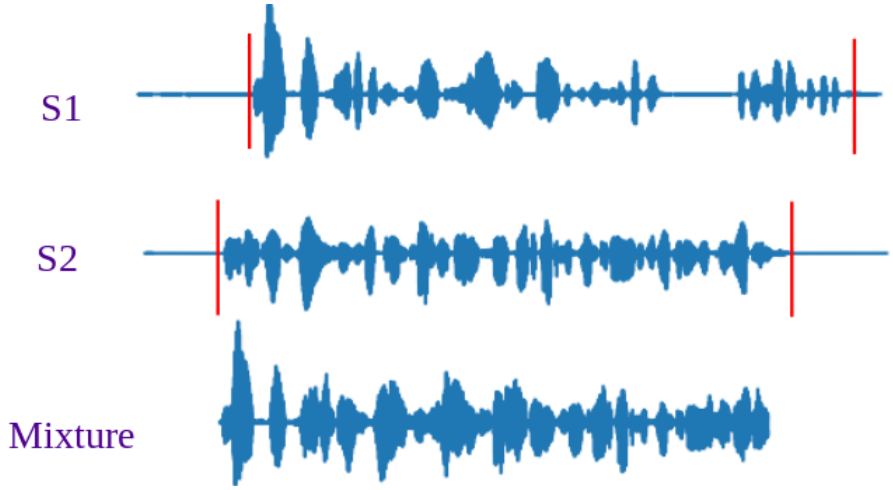


***Figure 5.4*** *The mixture is formed from the trimmed two different source signals.*

For the experiment setup 3, both offline test setup and online test setup have been conducted. Additionally, from the third line of the Table 5.3, we can seen that by shortening the window length to 8 ms from 32 ms, another 1.1 dB drop in SDR is observed compared to the second experiment setup. Figure 5.5 shows an example of the spectrogram of the 32 ms window length, it can be clearly seen that, the larger window length, better resolution and fine structure of the spectrogram has, which explains the dropped performance.

Finally, for the online deep clustering system with 1.5 s buffer, from the last of the Table 5.3, another 0.7 dB in SDR is found compared to the second last line of the Table 5.3. On top of that, the effect of the buffer length is investigated and the results are seen from the Figure 5.6. It can be concluded from the Figure 5.6 that separation performance increases with buffer duration increases. However, it stops at the 0.3 s and after that the performance keeps steady, which indicates that a relative good separation performance can be achieved from quite as short as 0.3 s buffer.

One example of the low-latency deep clustering method for separation is shown in the Figure 5.8. As it is shown, the separated sources are very close to the groundtruth.

***Table 5.3*** *Evaluation metrics (dB) of different system setup*

| | **Window length** | **Hop length** | **SDR** | **SIR** | **SAR** |
|---|---|---|---|---|---|
| BLSTM | 32 ms | 8 ms | 7.9 | 15.6 | 9.2 |
| LSTM | 32 ms | 8 ms | 6.9 | 14.5 | 8.4 |
| LSTM | 8 ms | 4 ms | 5.8 | 13.6 | 7.2 |
| Online LSTM (1.5s buffer) | 8 ms | 4 ms | 5.1 | 12.6 | 6.7 |

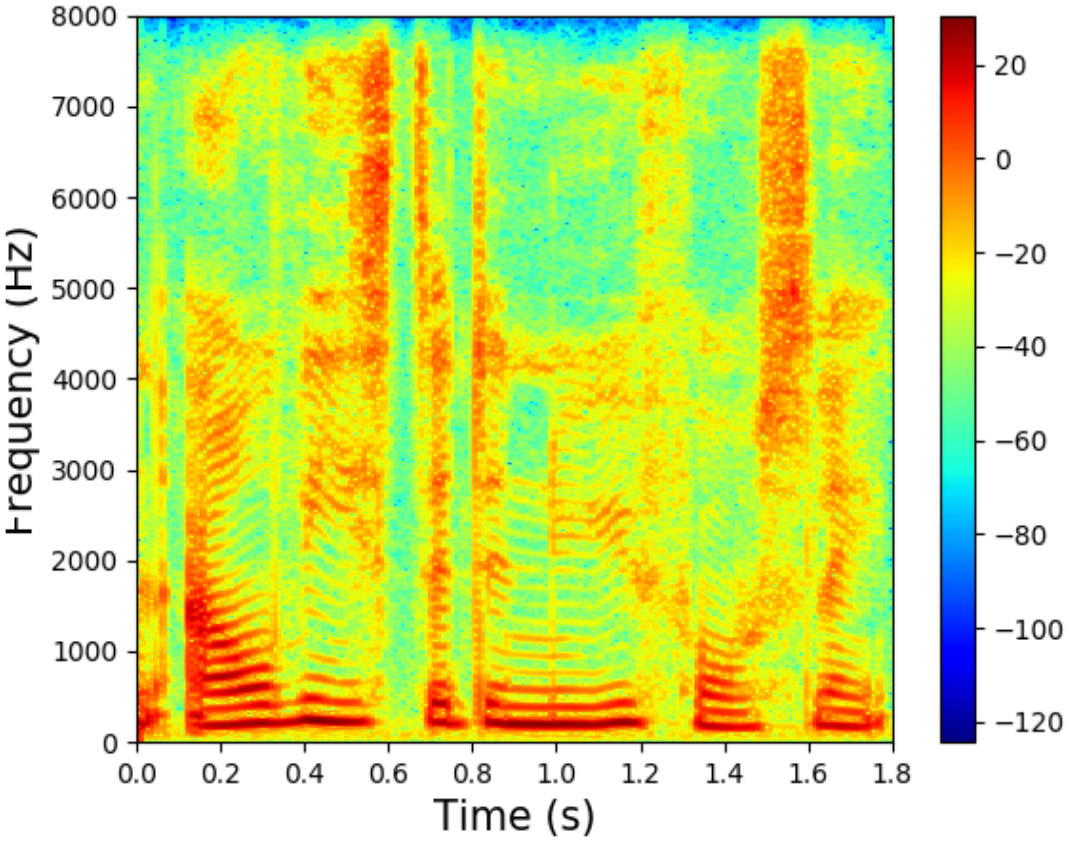


***Figure 5.5*** *An example of spectrogram of a signal with 32 ms window length*

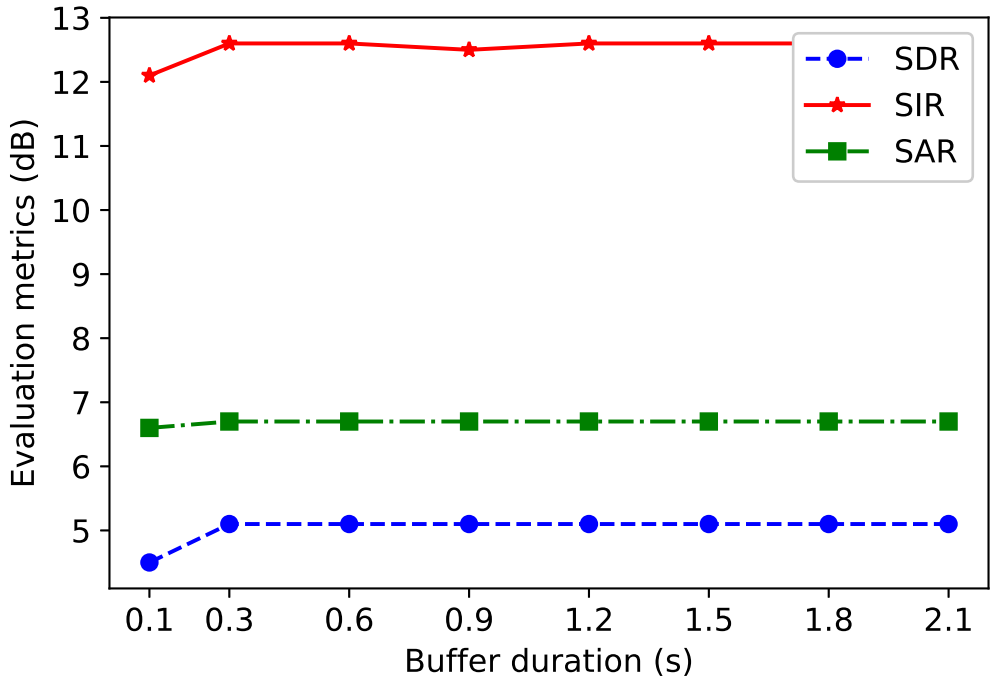


***Figure 5.6*** *Evaluation metrics with different buffer length. The blue line depicts the SDR metric (dB), the red line shows the SIR metric (dB), and the green line represents the SAR metric (dB).*

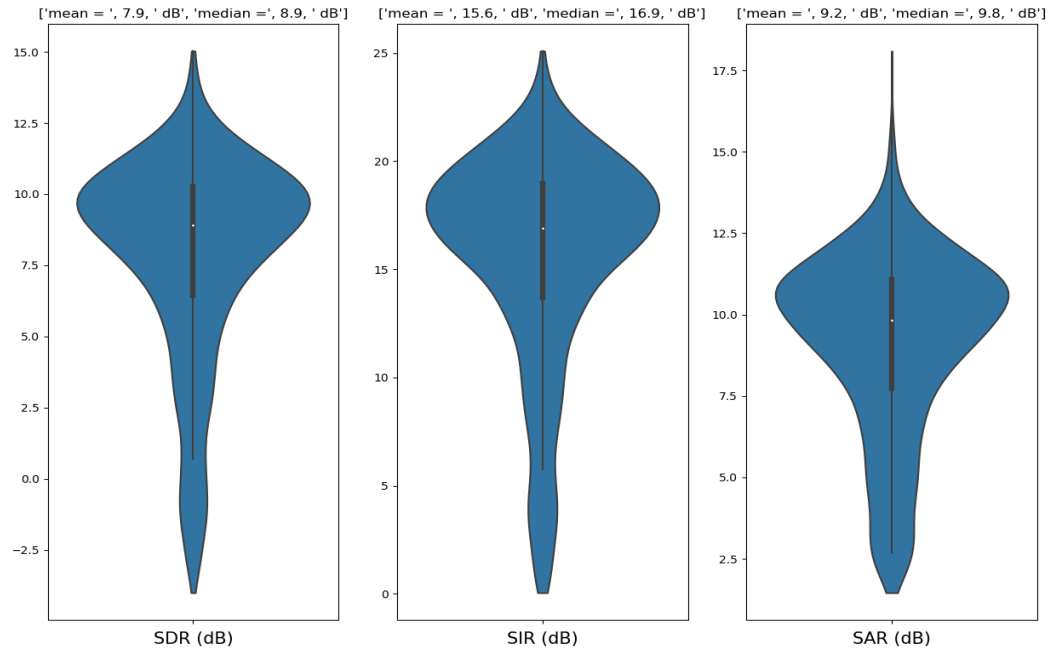


***Figure 5.7*** *The distribution of the result for BLSTM networks. The left one represents the distribution of SDR metric, SIR is shown in the middle panel, and SAR is displayed in the right side.*

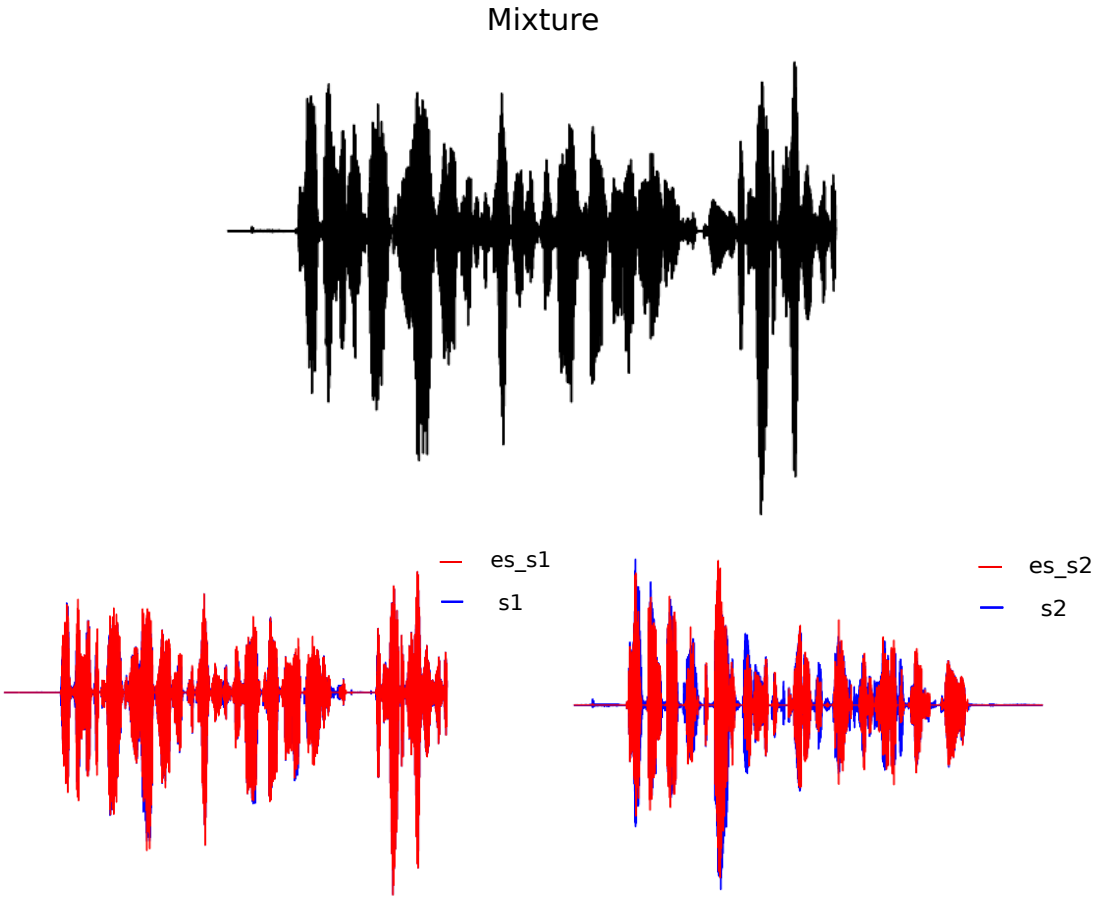


***Figure 5.8*** *Examples of online DC method. The upper part shows the mixture in the time domain. In the second row, the left most depicts the estimated source$_1$ shown in the red line and the corresponding groudtruth source$_1$ shown as the blue line. The right most represents the estimated source$_2$ shown as the red line and the groudtruth source$_2$ shown as the blue line.*

# 6 Conclusions and Future Works

In this thesis , a low-latency variant for speech separation based on deep clustering method is proposed. Specifically, three different network setups are investigated in order to analyze three factors, the usage of different networks, window length, and buffer length. The first experiment setup consists of 4 Layers of BLSTM with larger window length, the second setup is formed by 4 layers of LSTM with larger window length (32 ms) features, and the last one studies the 4 layers of LSTM with shorter window length (8 ms) features.

It is found that separation performance achieves the highest by using BLSTM networks and 32 ms window length. Additionally, 1 dB drop in SDR is noticed by only changing networks from BSLTM to LSTM while keeping other parameters the same, which indicates that the future information is crucial to the separation. Furthermore, it is found another 1.1 dB drop in SDR by shortening the window length to 8 ms from 32 ms compared to the experiment setup 2. From this, it can be explained that the spectrogram of signals with the shorter window length have poorer frequency resolution compared to the signals with larger window length. Finally, for the online deep clustering separation system, it is discovered that the separation performance improves as the buffer length increases and keeps steady after 0.3 s. This tells that after a buffer of 0.3 s, the speaker separation system is able to operate with the latency of 8 ms.

As for the future work, there are many things which are worth of exploration. For example, shown as the experiment results, the separation performance on the same gender mixtures is worse than the cross gender mixtures. Regards to this, it is necessary to investigate on improving the performance of the same gender. Additionally, performance of low-latency separation system is still far behind the offline separation system so it is worth of study on improving the performance of online system. For example, various frequency resolution can be tried out to improve the online DC separation performance. Furthermore, as deep clustering method is speaker-independent, language independent, experiments like applying it to some other language dataset or even some other more universal source separation problems can also be studied.

# References


[1] E. Vincent, T. Virtanen, and S. Gannot, *Audio source separation and speech enhancement*. John Wiley & Sons, 2018.

[2] A. S. Bregman, *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[3] K. Nagahama and S. Matsui, "Sound source separation device, speech recognition device, mobile telephone, sound source separation method, and program," Feb. 7 2012. US Patent 8,112,272.

[4] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein, "Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation," *arXiv preprint arXiv:1804.03619*, 2018.

[5] J. Barker, R. Marxer, E. Vincent, and S. Watanabe, "The third 'chime'speech separation and recognition challenge: Dataset, task and baselines," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 504–511, IEEE, 2015.

[6] L. Bramsløw, "Preferred signal path delay and high-pass cut-off in open fittings," *International Journal of Audiology*, vol. 49, no. 9, pp. 634–644, 2010.

[7] S. T. Roweis, "One microphone source separation," in *Advances in Neural Information Processing Systems 13* (T. K. Leen, T. G. Dietterich, and V. Tresp, eds.), pp. 793–799, MIT Press, 2001.

[8] T. Virtanen, "Monaural Sound Source Separation by Non-Negative Matrix Factorization with Temporal Continuity and Sparseness Criteria," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, March 2007.

[9] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1562–1566, 2014.

[10] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 708–712, 2015.

[11] J. Chen and D. Wang, "Dnn based mask estimation for supervised speech separation," in *Audio source separation*, pp. 207–235, Springer, 2018.

[12] G. Naithani, T. Barker, G. Parascandolo, L. Bramsløw, N. H. Pontoppidan, and T. Virtanen, "Low latency sound source separation using convolutional recurrent neural networks," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 71–75, 2017.

[13] 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)*, pp. 31–35, IEEE, 2016.

[14] J. Park, J. Kim, H. Choi, and M. Hahn, "Convolutional recurrent neural network based deep clustering for 2-speaker separation," in *Proceedings of the 2018 2nd International Conference on Mechatronics Systems and Control Engineering*, pp. 103–106, ACM, 2018.

[15] Z.-Q. Wang, J. Le Roux, and J. R. Hershey, "Alternative objective functions for deep clustering," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 686–690, 2018.

[16] Z. Chen, Y. Luo, and N. Mesgarani, "Deep attractor network for single-microphone speaker separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 246–250, 2017.

[17] J. Hidalgo, "Low latency audio source separation for speech enhancement in cochlear implants," Master's thesis, Universitat Pompeu Fabra, 2012.

[18] M. A. Stone, B. C. Moore, K. Meisenbacher, and R. P. Derleth, "Tolerable hearing aid delays. V. estimation of limits for open canal fittings," *Ear and Hearing*, vol. 29, no. 4, pp. 601–617, 2008.

[19] J. Agnew and J. M. Thornton, "Just noticeable and objectionable group delays in digital hearing aids," *Journal of the American Academy of Audiology*, vol. 11, no. 6, pp. 330–336, 2000.

[20] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Upper Saddle River, NJ, USA: Prentice Hall Press, 3rd ed., 2009.

[21] D. Wang, "On ideal binary mask as the computational goal of auditory scene analysis," in *Speech separation by humans and machines*, pp. 181–197, Springer, 2005.

[22] C. Hummersone, T. Stokes, and T. Brookes, "On the ideal ratio mask as the goal of computational auditory scene analysis," in *Blind source separation*, pp. 349–368, Springer, 2014.

[23] P. Comon and C. Jutten, *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press, 2010.

[24] H. E. Driver and A. L. Kroeber, *Quantitative expression of cultural relationships*, vol. 31. University of California Press, 1932.

[25] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[26] D. E. Rumelhart, G. E. Hinton, R. J. Williams, *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[27] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645–6649, IEEE, 2013.

[28] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, "A system for robotic heart surgery that learns to tie knots using recurrent neural networks," *Advanced Robotics*, vol. 22, no. 13-14, pp. 1521–1537, 2008.

[29] J. F. Kolen and S. C. Kremer, *Gradient Flow in Recurrent Nets: The Difficulty of Learning LongTerm Dependencies*. IEEE, 2001.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[31] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with lstm recurrent networks," *Journal of machine learning research*, vol. 3, no. Aug, pp. 115–143, 2002.

[32] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[33] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," in *International Workshop on Artificial Neural Networks*, pp. 195–201, Springer, 1995.

[34] J. Spanier and K. B. Oldham, *An atlas of functions*. Hemisphere publishing corporation New York, 1987.

[35] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

[36] D. M. Hawkins, "The problem of overfitting," *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.

[37] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, pp. 1137–1145, Montreal, Canada, 1995.

[38] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, vol. 13, p. 402, 2001.

[39] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[41] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[42] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," in *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, p. 4, ACM, 2014.

[43] E. M. Grais and M. D. Plumbley, "Single channel audio source separation using convolutional denoising autoencoders," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1265–1269, IEEE, 2017.

[44] S. Wang, G. Naithani, and T. Virtanen, "Low-latency deep clustering for speech separation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 76–80, IEEE, 2019.

[45] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, (Stroudsburg, PA, USA), pp. 357–362, Association for Computational Linguistics, 1992.

[46] 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.

[47] B. McFee *et al.*, "librosa 0.5.0." `https://doi.org/10.5281/zenodo.293021`, Feb. 2017. Online; accessed November 2019.

[48] F. Chollet *et al.*, "Keras." `https://github.com/fchollet/keras`, 2015. Online; accessed November 2019.

[49] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, 2016.

[50] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

# APPENDIX

In chapter 3.3, for offline deep clustering separation system, the loss is defined as the Equation 3.7. However, in the real implementation, Equation 3.8 is employed. The mathematical prove is given as follow.

$$
\begin{aligned}
&L = \|VV^T - YY^T\|^2 \\
&= tr[(VV^T - YY^T)^T(VV^T - YY^T)] \\
&= tr[(VV^T - YY^T)(VV^T - YY^T)] \\
&= tr[VV^TVV^T - VV^TYY^T - YY^TVV^T + YY^TYY^T] \\
&= tr[VV^TVV^T] - tr[VV^TYY^T] - tr[YY^TVV^T] + tr[YY^TYY^T] \\
&= tr[(VV^T)^T(V^TV)] - tr[(V^TY)^T(V^TY)] - tr[(V^TY)^T(V^TY)] + tr[(Y^TY)^T(Y^TY)] \\
&= \|VV^T\|^2 - 2\|V^TY\|^2 + \|Y^TY\|^2
\end{aligned}
$$