COMPARATIVE STUDY ON MUSIC SOURCE SEPARATION METHODS

MÜZİK KAYNAĞI AYIRMA YÖNTEMLERİ ÜZERİNE KARŞILAŞTIRMALI ÇALIŞMA

BURAK BAYSAL

PROF. DR. MEHMET ÖNDER EFE
Supervisor

Submitted to
Graduate School of Science and Engineering of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of the Degree of Master of Science
in Computer Engineering

December 2022

ABSTRACT

COMPARATIVE STUDY ON MUSIC SOURCE SEPARATION METHODS

Burak Baysal
Master of Science, Computer Engineering
Supervisor: Prof. Dr. Mehmet Önder EFE
December 2022, 102 pages

Blind source separation is the concept that separates the source signals from the mixture signal. ”Blind” means no prior knowledge of the source or the mixing environment. The blind source separation problem is a problem domain that has been studied in the literature for a long time. The most familiar problem example of the domain is the ”Cocktail Party Problem.” Imagining the party environment and the sound of the environment is to be recorded. The recorded audio signal comprises audio signals such as speech, laughter, music, or even the footstep from the street. Is it possible to extract the source signals, i.e., the audio signal of the music, from this mixture signal? Blind source separation methods aim to obtain the original signals with the least possible loss.

In the beginning, statistics and computational approaches were dominant in the literature. Independent component analysis methods were widely used in blind source separation studies in early studies. Following these approaches, which are based on matrix factorization, methods such as the Degenerate Unmixing Estimation Technique, which contains more complex calculations, have emerged. Recently, machine learning-based approaches have become dominant in the literature, and deep learning methods have begun to be utilized broadly in separating signals.

This thesis aims to comprehensively compare the methods related to the problem domain of blind source separation. In addition to the techniques in the literature for a long time, deep learning-based models employed effectively by today's technologies are also included in the comparative study. Seven different methods of source separation are studied in the thesis. While the classical methods FastICA, NMF, and DUET are included within the scope of the thesis, the machine learning-based models Open Unmix, Spleeter, Wave-U-Net, and Hybrid Demucs have been examined. After providing detailed information about the source separation methods, the experimental study was carried out. The MusDB18-HQ dataset was used during the experiment. Accordingly, an experiment was performed to analyze the audio signals and separate them into four components: vocal, drum, bass, and other. The performance of which method was evaluated with the SDR metric. The evaluation was also made according to music genres and added to the results of the thesis experiment.

Keywords: blind source separation, music source separation, music information retrieval

ÖZET

MÜZİK KAYNAĞI AYIRMA YÖNTEMLERİ ÜZERİNE KARŞILAŞTIRMALI ÇALIŞMA

Burak Baysal
Yüksek Lisans, Bilgisayar Mühendisliği
Danışman: Prof. Dr. Mehmet Önder EFE
Aralık 2022, 102 sayfa

Kör kaynak ayrıştırma problemi uzun zamandır literatürde üzerine çalışmalar yapılan bir problem alanıdır. Problem alanına dair bilinen en yaygın örnek ise ”Kokteyl Parti Problemi”’dir. Problemin tanımında bir parti ortamından bahseder ve ortamın sesi kaydedilecek olunursa, kaydedilen bu ses sinyali konuşma, kahkaha ve müzik v.b ses sinyallerinin bir karışımı olacaktır. Peki bu karışım sinyalinden kaynak sinyalleri yani örneğin müziğe ait ses sinyalini çıkartmak mümkün müdür? Kör kaynak ayırma metotları, karışım sinyalinden orijinal sinyallerin mümkün olan en az kayıpla elde edilmesini amaçlar.

Başlarda literatürde istatistik ve hesaplama temelli yaklaşımlar hakimdi. Bağımsız bileşen analizi metotlar ilk zamanlarda kör kaynak ayrıştırma çalışmalarında çokça kullanılmaktaydı. Matris faktorizasyonunu temel alan bu yaklaşımların ardından daha karmaşık hesaplamaları içeren Dejenere Ayrıştırma Tahmin Tekniği gibi yöntemler ortaya çıkmıştır. Son zamanlarda ise literatürde makine öğrenmesi temelli yaklaşımlar baskın hale gelmiş ve derin öğrenme metotları sinyalleri ayrıştırmada yoğun halde kullanılır olmaya başlamıştır.

Bu tez çalışmasıyla kör kaynak ayırma problem alanına dair metotların kapsamlı bir karşılaştırması amaçlanmıştır. Literatürde uzun zamandır yer alan metotların yanı sıra günümüz teknolojilerinin etkin kullandığı derin öğrenme temelli modeller de karşılaştırmalı çalışmaya dahil edilmiştir. Kaynak ayrıştırmaya dair yedi farklı metot tez kapsamında çalışmaya dahil edilmiştir. Klasik metotlardan FastICA, NMF ve DUET tez kapsamında çalışırken, makine öğrenmesi temelli metotlardan da Open Unmix, Spleeter, Wave-U-Net ve Hybrid Demucs ile modelleri incelenmiştir. Kaynak ayrıştırma metotlarına dair detaylı bilgi sağladıktan sonra deneysel çalışma gerçekleştirilmiştir. Bu doğrultuda ses sinyallerinin analiz edilerek vokal, davul, bas ve diğer olmak üzere dört farklı bileşene ayrıştırılması deneyinde hangi metodun nasıl performans gösterdiği SDR metriği ile değerlendirilmiştir. Aynı zamanda mizük türlerine göre de değerlendirme yapılarak tez deney sonuçlarına eklenmiştir.

Keywords: kör kaynak ayrıştırma, müzik kaynağı ayrıştırma, müzik bilgisi alma

ACKNOWLEDGEMENTS

I dedicated this study to my esteemed advisor Prof. Dr. Mehmet Önder EFE, who assisted me in bringing this study with all his patience, supported and guided me with his wisdom and approach,

And to my dear friends Hakan SARKIN and Barış YEŞİLYURT, who comforted me when I was overwhelmed during my thesis study and supported me in moving on,

And to my mother-in-law and father-in-law, Şengül and Mehmet ŞEN, who always supported me during this study.

And to my dear parents, Fatma and Rasim BAYSAL, who have made outstanding efforts to get me to where I am today.

And finally, to my lovely wife, Aydan ŞEN BAYSAL, and my dear little son Teoman BAYSAL, who have supported me with extraordinary devotion, been by my side, and helped me in my long study.

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

BSS	:	**B**lind **S**ource **S**eparation
ICA	:	**I**ndependent **C**omponent **A**nalysis
MIR	:	**M**usic **I**nformation **R**etrieval
MIMO	:	**M**ultiple **I**nput **M**ultiple **O**utput
CLI	:	**C**ommand **L**ine **I**nterface
ANN	:	**A**rtificial **N**eural **N**etwork
CNN	:	**C**onvolutional **N**eural **N**etwork
RNN	:	**R**ecurrent **N**eural **N**etwork
LSTM	:	**L**ong **S**hort-**T**erm **M**emory
ICA	:	**I**ndependent **C**omponent **Analysis**
PCA	:	**P**rincipal **C**omponent **Analysis**
NMF	:	**N**on-negative **M**atrix **F**actorization
TF	:	**T**ime-**F**requency
DUET	:	**D**egenerate **U**nmixing **E**stimation **T**echnique

1. INTRODUCTION

Blind Source Separation (BSS) is a significant field of study in digital signal processing. In the early times of BSS studies, approaches were mainly based on matrix factorization and higher-order statistics; after the emergence of artificial neural networks, researchers tried to use network-based approaches in BSS studies. Some of these studies gave promising results. BSS methods are used in various domains like feature extraction, financial time series analysis, speech recognition, biomedical signal analysis (e.g., EEG, fMRI), telecommunications, data mining, and signal denoising [3]. With its opportune outputs, BSS is a popular topic in signal processing, artificial network, and other disciplines [4].

What makes the separation of sources ”blind”? The ”blind” term refers to having little or no knowledge about source signals’ or the mixing environments’ frequency characteristics. A deficiency of knowledge makes BSS problems more challenging. That is why some assumptions about source signals are used while applying these methods, such as non-Gausuanity, and statistical independence [4]. Also, it is tough to make a clean measurement of signal mixtures since the environment in which the signals are mixed will often be less sensitive than the experimental environment. Two main limitations cause this situation; a) a source signal always has noise, and b) environmental voices such as footsteps, e.g., may corrupt the source signal [5]. When thinking of the audio example, with that limited information, BSS aims to construct source signals by using only observed signals received by a sensor, mostly a microphone, located in the room or concert hall e.g.

One of the well-known typical BSS problems is the ”cocktail party problem.” Talking, laughing, music, and noise can be heard simultaneously when considering a party atmosphere. If enough microphones are located to surround the atmosphere, all of these microphones record the different linear mixtures of these sounds. BSS refers to the operation of constructing original music or voice signals from these recorded linear mixture signals by microphones [4]. The overall ”cocktail party” problem is depicted in Figure 1.1.

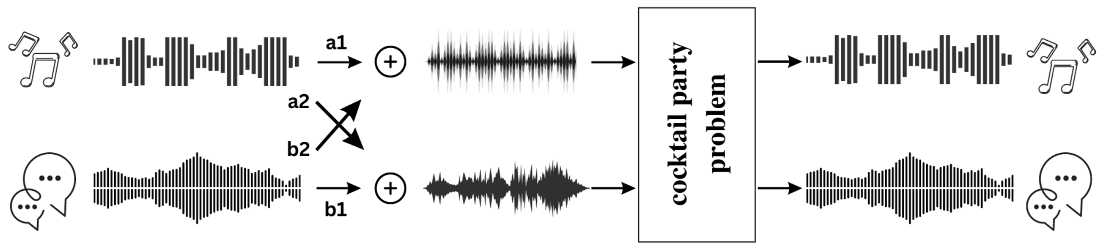


Figure 1.1 Sample depiction of the cocktail party problem.
$s1$,$s2$ are denotes the sounds which are produced by music, and voice are superposed and recorded by the microphones. Both microphones receive the unique weighted sum of the two sources. The weight of each microphone ($a1$,$b1$ and $a2$,$b2$) indicates the proximity of the microphone to the sound source [6].

1.1. Motivation

Many algorithms offer solutions to BSS problems. Most BSS methods are based on prior knowledge and a theoretically constructed cost function; therefore, they are considered unsupervised learning methods. Pre and post-processing stages of data are essential for extracting physically expressive, reliable, coherent components [4]. BSS methods may be divided into two groups; classical approaches and machine learning-based approaches. Some classical algorithms such as Non-negative matrix factorization (NMF), FastICA [7] [8] are based on matrix factorization but with the emerging machine learning methods in recent years, models are getting more sophisticated and versatile. In addition to being complex, machine learning-based models also produce promising results in Music Information Retrieval (MIR) field. In this thesis, we aims to provide an extensive treatment of well-known classical and contemporary machine learning-based methods in the field of MIR. We give a guiding comparison of experimental results in terms of time, genre, and SDR scores. The methods used and their classification are shown in Table 1.1.

1.2. Contribution of this Thesis

In this thesis, we aim to provide a comprehensive comparative study of BSS methods in the field of MIR, implement the methods used, and then observe and report their experimental

Classical Approaches	**Machine Learning-Based Approaches**
FastICA	Spleeter
Non-Negative Matrix Factorization	OpenUnmix
Degenerate Unmixing Estimation Technique	Hybrid Demucs
	WaveUNet

Table 1.1 Used Blind Source Separation Methods

performances in similar hardware and software environments. The key contributions of the dissertation can be listed as follows;

- Giving mathematical principals of well-known classical algorithms and model implementation details of popular machine learning algorithms used in music source separation
- Generally, well-known classical algorithms have not been used in music source separation problems; machine learning algorithms mostly run with CLI. Sample implementations of these algorithms in the thesis may be a reference for future studies.
- Comparison of classical and machine learning algorithms with the same dataset and environmental conditions.
- In addition to comparing SDR results, extensive experiments have been done on genre and time domain. These results may help researchers with their genre-specific studies.

1.3. Organization

The organization of the thesis is as follows:

- Chapter 1 presents an introduction to the main subject of the thesis, our motivation, and our contributions.
- Chapter 2 provides background knowledge about blind source separation, and machine learning terms and definitions.

- Chapter 3 shows the results of literature research about comparative studies on blind source separation methods.
- Chapter 4 introduces classical methods and machine-learning based methods which are used in this study
- Chapter 5 states the methodology of the experimental study.
- Chapter 6 demonstrates the experiments' time, genre, and SDR results.
- Chapter 7 states the conclusion of the thesis, which includes a summary and comments about future directions.

2. BACKGROUND OVERVIEW

The following part of this thesis moves on to describe in detail the BSS problem with the mixing and unmixing model and the mathematical foundations of the problem. Then, some mathematical principles used in the algorithms are introduced. At the end part of the section, machine learning basics and some concepts used in machine learning models are explained.

2.1. Blind Source Separation

In real life, the easiest way to solve big problems is to break them into smaller pieces. On this subject, Henry Ford's aphorism is generic in terms of describing the issue; ”There are no big problems; there are just a lot of little problems.” Blind source separation also sees mixing signals as the big problem and aims to reconstruct the source signals as separated signals to make this big problem smaller. The ”blind” concept here indicates that we do not have enough priori knowledge about the source signals or the coefficients of the mixing environment that produces the mixture.

BSS has been applied in various fields. For example, in [9], a popular BSS method Independent Component Analysis, shortly ICA, was used in image fusion tasks for remote sensing at the geological spatial information processing domain. The most famous BSS example in the audio signal processing domain is the ”cocktail party” problem. If a party atmosphere is to be imagined, music is played in the background, and some people talk or laugh loudly; also, there may be coughing or clapping sounds. In case enough input sensors, microphones for this example, are placed around the room, each microphone can measure these audio signals with different weights [4]. The main goal of BSS is to reconstruct original signals from observed mixed signals and generate isolated components of mixed audio, such as talking, music, and clapping.

Human-made signals are transmitted in a medium such that convolutive transmission may change the spatial and spectral characteristics of the signals. Consequently, the initially transmitted source signals and the output may differ completely [1]. Mixing the source

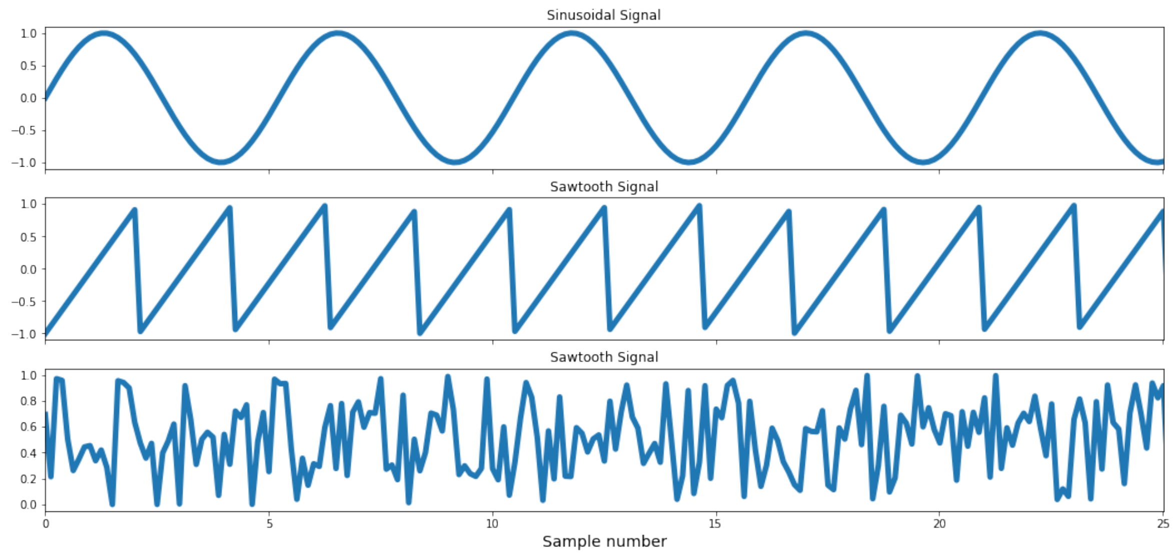


Figure 2.1 The original signals

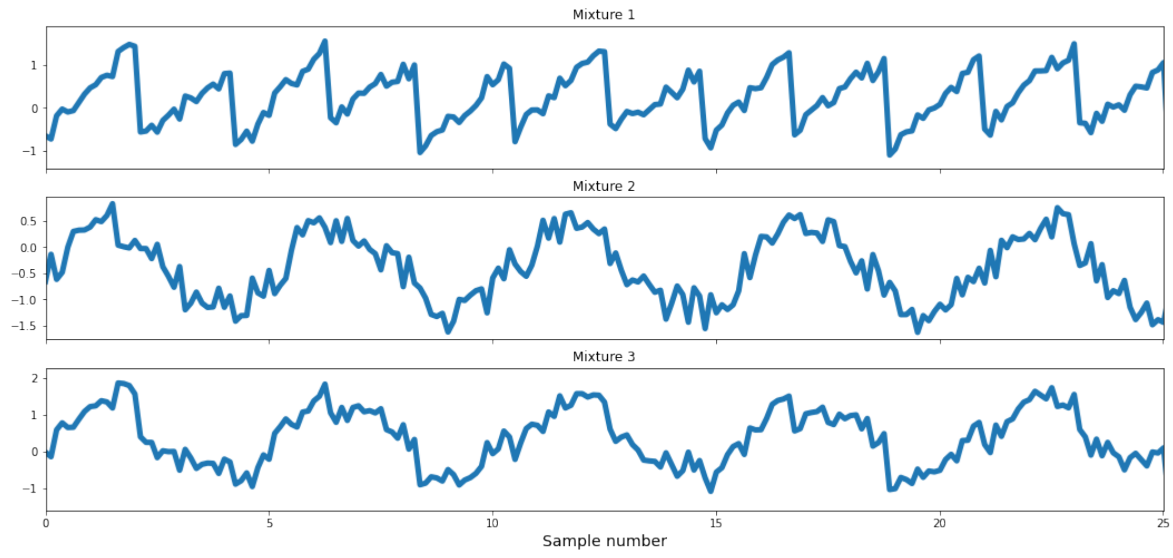


Figure 2.2 Mixing signals formed after source signals are mixed

signals produces some effects. These effects then provide a basis for extracting the source signals from the mixing signals. For example, suppose that three independent signals are artificially generated, such as a sine wave, a sawtooth signal, and a random signal, as seen in Figure 2.1. After a random mixing matrix mixes these signals, mixture signals comprise as in Figure 2.2. The three effects mentioned are as follows [10];

- Independence: Assuming statistically independence of source signals plays a crucial role in BSS problems. Although this assumption does not fit well with real-world scenarios, it is necessary to solve the problem in a practical sense. However, mixture signals are dependent, even if the source signals are independent. The source signals are shared between each mixture signal, so this common usage of signals ensures the dependencies between mixture signals. Figure 2.3 illustrates this point clearly.

- Normality: Source signals are tended to be non-Gaussian distribution. For example, if a speech signal is plotted as a histogram, it shows a ”peaky” distribution. However, based on the central limit theorem, when these source signals are mixed, the mixture signals distribution are tended to be more Gaussian, meanly plotted as a ”bell-shaped” histogram. The mixture of non-Gaussian signals will contain more information about the mixing. If looking at the joint distribution of non-Gaussian signals, it will scatter on a square. After getting the mixture by mixing these two independent, non-gaussian signals by an orthogonal mixing matrix, if the mixtures of these two signals are plotted, a meaningful parallelogram structure will be formed and show uniform distribution. This prior knowledge will help us find the other from the signal. The distribution of Gaussian signals and the mixture of these two signals will be almost symmetrical. It will not be possible to obtain meaningful data for the mixing matrix to solve the mixture. Figure 2.4 is a good illustration of the gaussian distribution effect on mixing.
- Complexity: The complexity of the mixture signal is greater than or equal to the complexity of the least complex signal among the source signals that constitute the mixture signal. This can be briefly in Figure 2.5

These effects help extract source signals using several mixture signals by extracting source signals with as many independent, non-gaussian and complex characteristics in that mixture signals [10]. The overall steps of BSS are depicted in Figure 2.6 and include pre-processing, separation, and post-processing. For inferring informative components after the separation process, importance should be given to pre-processing and post-processing steps [4].

The broad use of BSS can be expressed mathematically as below;

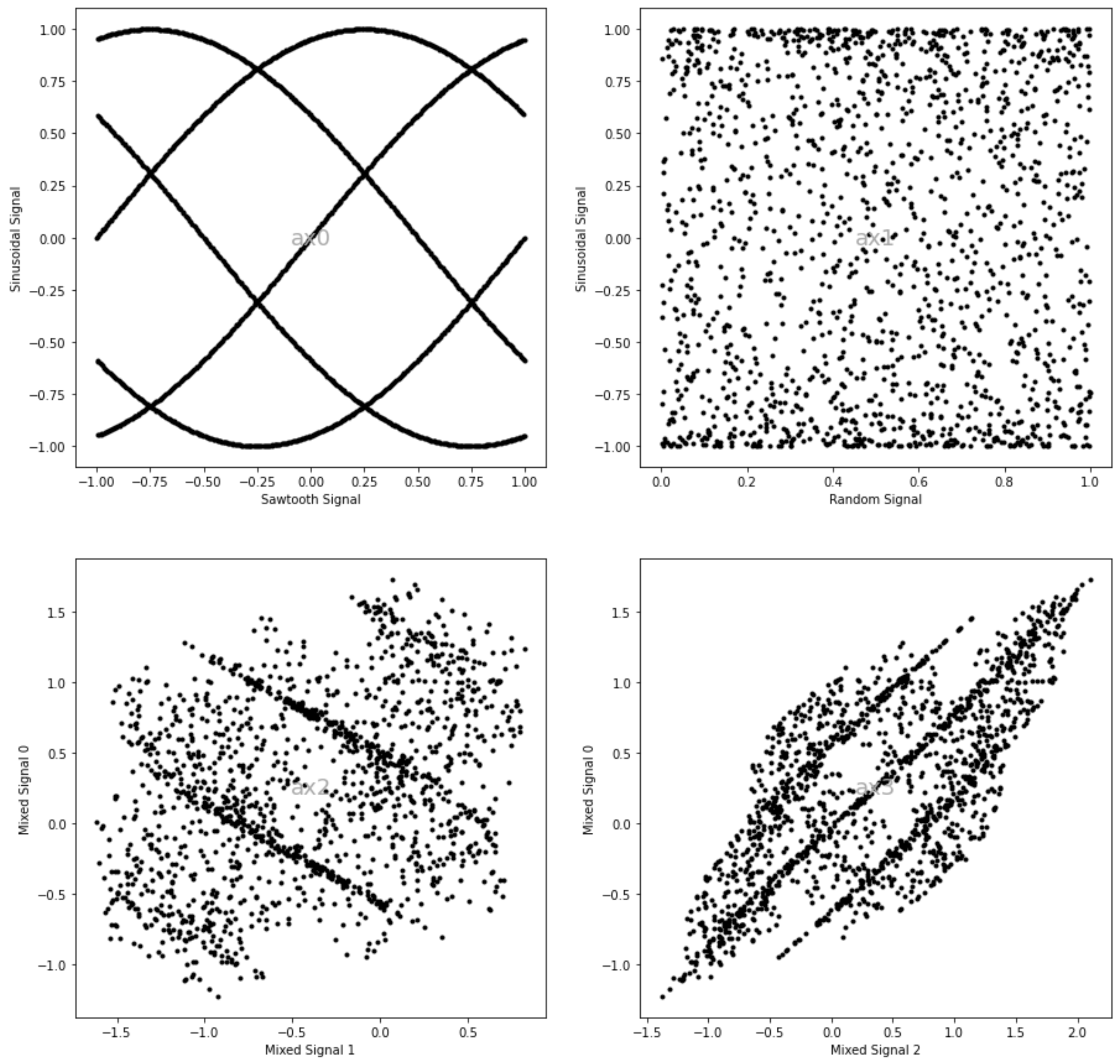


Figure 2.3 Independence of source and mixing signals

Even if the independence of the source signals, the mixture signals that comprised by the source signals are correlated with each other, because they include the proportion of source signals within them. If the scatter of the signals defined in Figure 2 and their mixtures are plotted, the top row of the graph shows the scatters of the sinusoidal-sawtooth signal on the left side and the sinusoidal-random signal on the right side. The scatter structure in these two subgraphs indicates the independence between signals. The scatters of the mixtures of signals are shown at the bottom row of the graph. On the left side, the scatter of the mixture signal 0-1, and on the right side, the scatter of the mixture signal 0-2 is plotted. In contrast to the graph above, the structure's direction shows the dependence between mixture signals. They behave the same or in opposite directions, indicating the correlation between them.

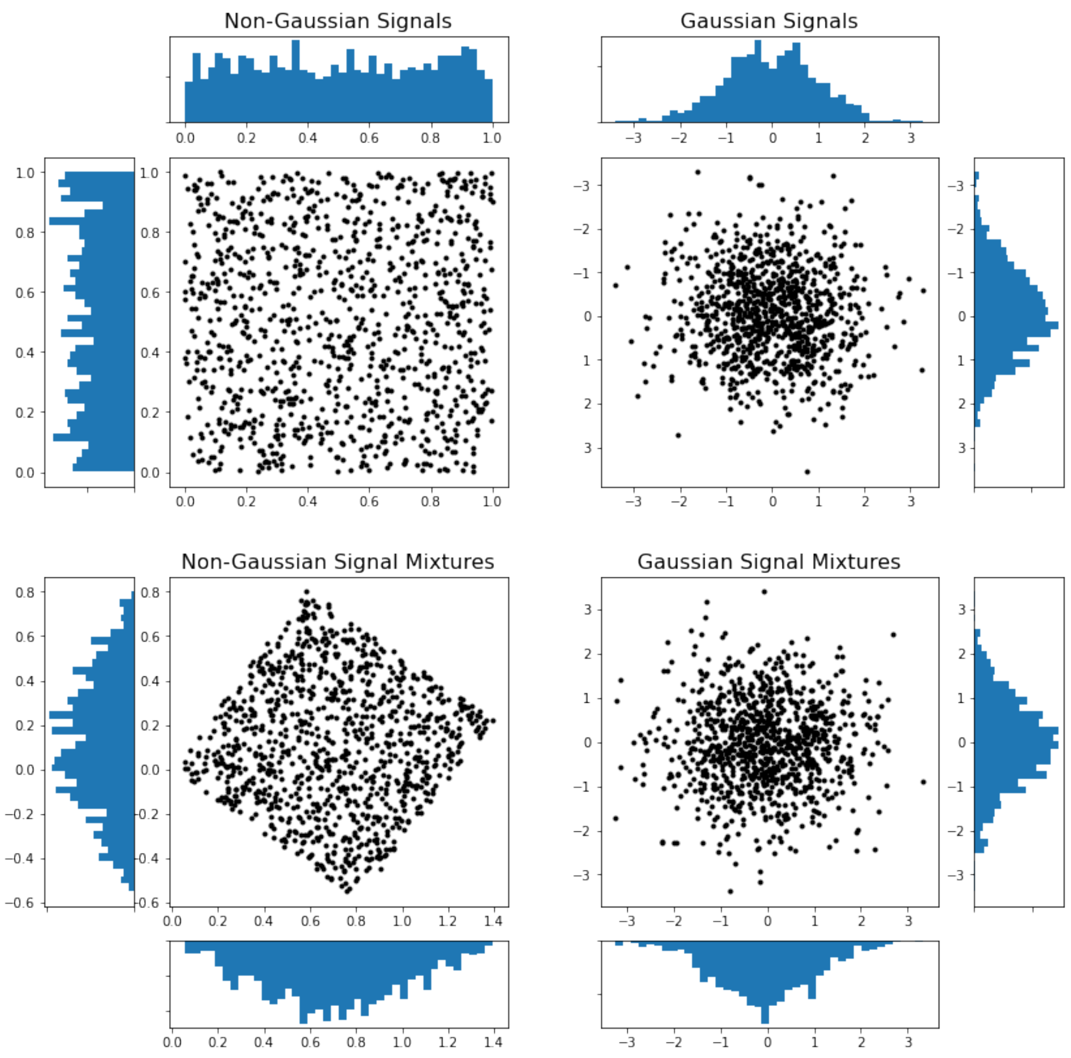


Figure 2.4 Gaussian and Non-Gaussian effect on signals and mixtures
The signals plotted on both sides were randomly generated, and the scatter plot describes the joint density distributions of the signals and their mixtures, and the histogram describes the distribution of the signals added to the graph.
Left: The originals and mixtures of the signals show a unifor distribution. In addition, although the original signals have a non-gaussian distribution, the mixture of their signals are tended to be gaussian distribution and creates a more bell-shaped histogram output.
Right: Both the original distribution and the mixing distribution of the two gaussian signals are almost symmetrical, and it is difficult to obtain data on the mixing matrix.

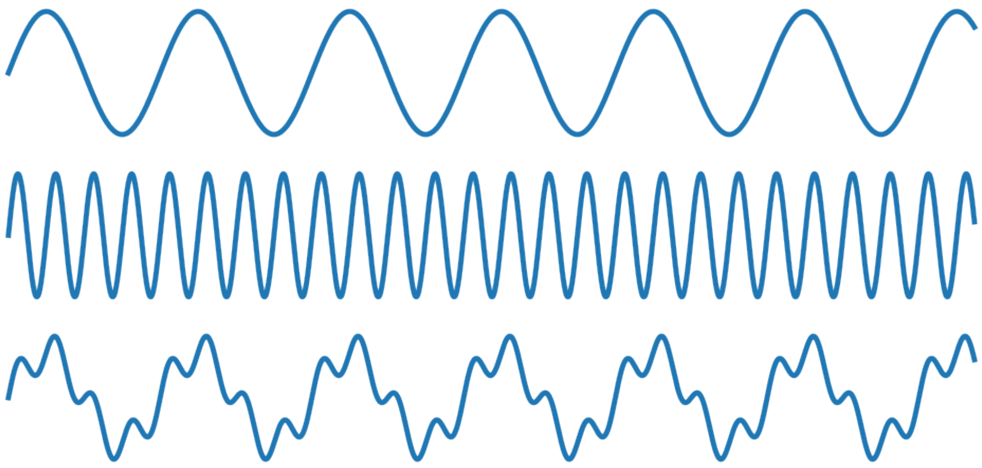


Figure 2.5 Complexity of mixing two signals
Two pure sine waves are plotted at the top of the figure. When these two sinusoidal waves are mixed, they create a more complex mixture signal, plotted below the figure. The complexity conjecture can be taken as a basis, and the prior knowledge that the signal to be extracted from a complex signal will be less complex will contribute to solving the BSS problems.

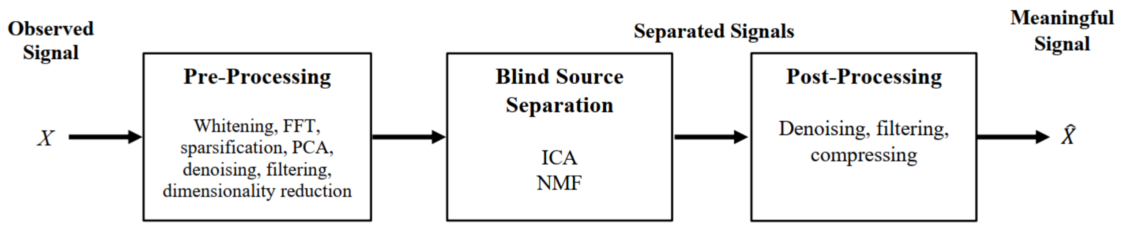


Figure 2.6 Processing Steps of Blind Source Separation

$\mathbf{x}(k) = [x_1(k), x_2(k), ..., x_n(k)]$ is denoted as an output sensor value of MIMO nonlinear system. BSS methods aim to discover a reverse procedure and rebuild the source signals, represented as $\mathbf{s}(k) = [s_1(k), s_2(k), ..., s_m(k)]$. The mathematical formulation is expressed below, and the block diagram of the overall architecture is depicted in Figure 2.7 [4];

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

$$\mathbf{y}(k) = \mathbf{W}\mathbf{x}(k) = \mathbf{W}\mathbf{A}\mathbf{s}(k) \tag{2}$$

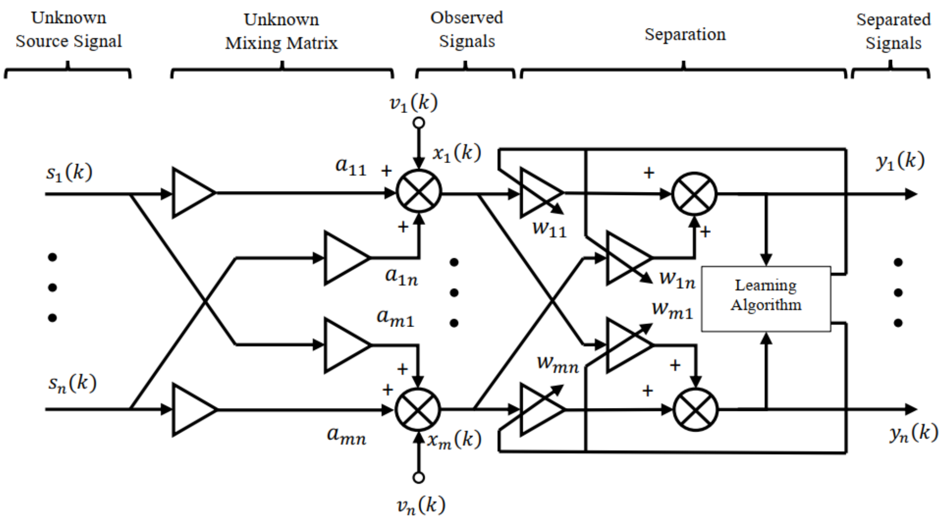


Figure 2.7 Blind Source Separation Block Diagram [1]

where **A** is an unknown, invertible, and square matrix, namely a mixing matrix, and **W** is unmixing matrix which is also the inverse of mixing matrix **A** [6]. The ”blind” characteristic of BSS reveals at Formula 1. Both mixing matrix, meanly how the sources are mixed, and source signals are unknown, so there is no prior knowledge while signals are observed.

The number of signals is also crucial to BSS. A consequence of many sources than observed mixing signal can lead to a challenging task in a BSS domain. If this constraint may be defined formally, there should be at least as many observed mixing signals as the source signals; if the system cannot ensure that $(m < n)$, this concept is called an overcomplete BSS problem [4] [10].

Sources are mixed in different physical environments. Also, how easily separate sources blindly depend enormously on this environment. Instantaneous mixing is the most straightforward mixing strategy, and the beginning BSS studies were designed accordingly. Nevertheless, in real-world cases, instantaneous mixing is commonly impossible [1]. BSS problems can be categorized into three titles to address different mixture styles such as[4];

(i) instantaneous linear mixing

(ii) linear convolution mixing

(iii) nonlinear mixing

2.1.1. Linear Mixing

2.1.1.1. Instantaneous Linear Mixture In an instantaneous mixing system, M source signals are linearly combined and received by the N input sensor. Thus, Equation 3 represents a time-invariant instantaneous linear mixing system, and in Figure 2.8 depicts the block diagram [1].

$$x_j(k) = \sum_{i=1}^{m} a_{ji} s_i(k) + v_j(k) \tag{3}$$

where the observed signals are indicated as $x_j(k)(j \in 1, 2, ..., N)$, the source signals are indicated as $s_i(k)(i \in 1, 2, ..., M)$, additive noise related with the kth sensor is indicated as $v_j(k)$, and $a_{ji}(k)(j \in 1, 2, ..., N, i \in 1, 2, ..., M)$ represents nxm mixing matrix $\mathbf{A}$ which includes linear coefficients of source signals while mixing.

Equation 4 is obtained in case using the vector form of the components and ignored noise factor in the Equation 3, resulting in a more easy-to-use equation; where $\mathbf{s(k)} = [s_1(k), s_2(k), ..., s_N(k)^T]$, $\mathbf{x(k)} = [x_1(k), s_2(k), ..., s_N(k)^T]$, and $\mathbf{A}$ is the mixing matrix [4].

$$\mathbf{x(k)} = \mathbf{As(k)} \tag{4}$$

BSS aims to find an mxn demixing matrix as denoted by $\mathbf{W}$, which is formulated at Equation 5 [1].

$$y_i(k) = \sum_{j=1}^{w} w_{ij}(k) x_j(k) \tag{5}$$

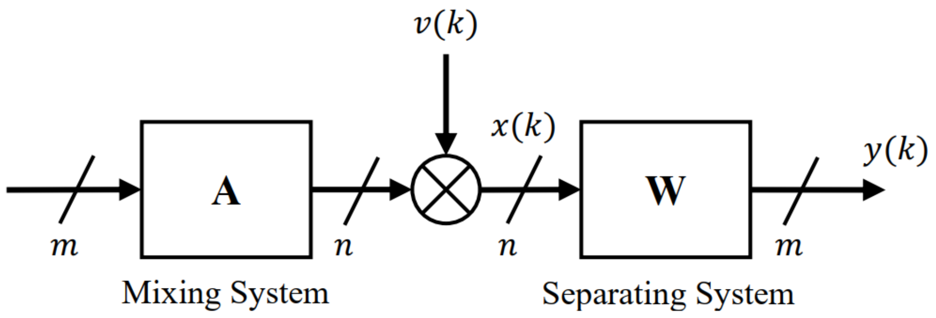


Figure 2.8 Instantaneous Linear Mixture Block Diagram [1]

2.1.1.2. Linear Convolution Mixture Convolutive mixtures of signals are closer to real-world cases. For instance, sound waves can be reflected, and the delays that occur due to the reflections caused by the waves hitting various obstacles during the propagation of the sound waves yield an acoustic sound. Hundreds or thousands of signals' reflections in such reverberant environments lead to time dependencies in the mixture. In addition to the time dependencies of signals, mixtures likewise have time dependencies [1].

Assuming the M sensor measures N statistically independent source signals following the convolution mixing process [11]. Observed signals are indicated as $x_j(t), j = 1, 2, ..., M$, source signals are indicated as $s_i(k), i = 1, 2, ..., N$, and the following expression, Eq. 6 indicates the linear convolution model [4].

$$x_j(t) = \sum_{i=1}^{N} a_{ji}(t) * s_i(t) = \sum_{i=1}^{N} \sum_{\tau=0}^{L-1} a_{ji}(\boldsymbol{\tau}) s_i(t-\tau) \tag{6}$$

where convolution operation is denoted by $*$, and a_{ij} is the jth sensor's impulse response about ith source signal.

All channels can be expressed as Lth-order finite impulse response filter; thus convolution mixing model can be shown with the FIR matrix representation proposed by Lambert as follows [4];

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{7}$$

where A is the finite impulse response filter of the form;

$$A = \begin{pmatrix} a_{11}^T & \dots & a_{1m}^T \\ \vdots & \ddots & \vdots \\ a_{n1}^T & \dots & a_{nm}^T \end{pmatrix} \tag{8}$$

where a_{ji} is a column that denotes the Lth-order FIR filter and has the L dimension.

To express it as a vector form, Equation 7 is represented as;

$$x(t) = \sum_{\tau=0}^{L-1} A(\tau)s(t-\tau) \tag{9}$$

2.1.2. Nonlinear Mixture

Nonlinear mixtures of signals are more convenient for real-world facts. It is a challenging problem to solve nonlinearly mixed signals in the BSS domain. Besides requiring additional information or well-defined boundaries. Nonlinear mixing is an extended version of linear mixing [12] [13] [14];

$$x(t) = f(s(t)) + n(t) \tag{10}$$

where $x(t)$ are denoted as an observed signal vectors with M-dimensional, $s(t)$ are denoted as a source signal vectors with N-dimensional, $n(t)$ are additive noise vectors with M-dimensional, and $f : \mathbf{R}^N \rightarrow \mathbf{R}^M$ is a reversible nonlinear separating function [4]. BSS aims to find a mapping function g to decompound source signals from measured signals by sensors. The expression of the mathematical model is as follows;

$$y(t) = g(x(t)) \tag{11}$$

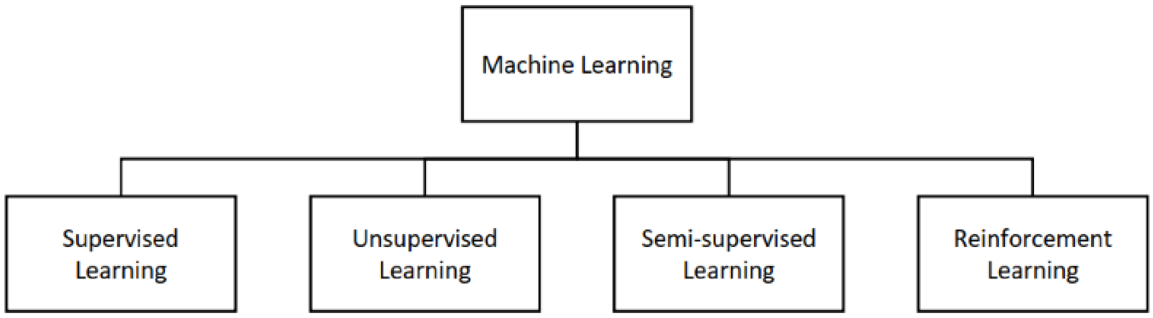


Figure 2.9 Classifications of Machine Learning Approaches

2.2. Machine Learning Essentials

2.2.1. Machine Learning Concept

Human beings base their past mistakes, behaviors, or previous developments and build new learnings on them. So, can computers do this skill too? This insight constitutes the basis for machine learning. Machine learning is the science of making better predictions or foresight by concluding old experiences, which are problem-specific historical data provided as input to the algorithm. The ”learning” in the phenomenon means that the algorithm can improve itself based on the data collected in various ways. While the algorithm explicitly designed for the specific problem domain can only solve that existing problem, the researchers search for an approach in which the machine, meanly the algorithm, will discover its solution based on the sample or training dataset initially provided [15]. Machine learning is one of the sub-branches of computer science and initially emerged due to pattern recognition and artificial intelligence studies. The machine learning concept has succeeded in time with computational statistics studies and has gained proficiency in prediction. Today, studies in areas such as pattern recognition, natural language processing, image recognition, and computer vision are built on machine learning [16].

Machine learning approaches may be divided into four main categories, as shown in Figure 2.9

- *Supervised learning:* Supervised learning aims to predict unobserved data by identifying the linkage between the data and the related label in the training set with various mathematical and statistical operations. The training set contains input data such as an image or audio and label data corresponding with the input as an output. If the problem is formulated, an unknown mapping function f maps input variable X to the dependent output variable Y, $Y = f(X)$. Supervised learning aims to find an approximate $\hat{f}$ function with provided $\{x_i, y_i\}_{i=1,\ldots,n}$ as a sample, which maps $\hat{f}(\mathcal{X}) \rightarrow \mathcal{Y}$, where $X \in \mathcal{X} \subset \mathbb{R}^d$, $Y \in \mathcal{Y} \subset \mathbb{R}$ are two random variables, and minimizes the expected loss. If the hypothesis space for f is restricted as $\mathcal{F}$, supervised learning formulation can be expressed as follows [17];

$$\hat{f} = \underset{f \in \mathcal{F}}{\operatorname{argmin}} E\left[L(Y, f(X))\right] \tag{12}$$

- *Unsupervised learning:* The goal of the unsupervised learning approach is to find the latent structure lying under the training dataset. In unsupervised learning, data in the training set is unlabelled. Therefore algorithm has an input X but not any output. Unsupervised learning is suitable for clustering o segmentation tasks.

- *Semi-supervised learning:* Observed data in the real world might only have a label sometimes. In semi-supervised learning, labeled data is rare in the dataset, and numerous unlabeled data exist. Semi-supervised learning focuses on classifying unlabelled data with the inferred information from labeled data.

- *Reinforcement learning:* It represents learning based on trial and error. The agent interacts with the environment e with state s after input i comes to the agent and takes action a. The system produces a scalar reinforcement signal according to the environment and state transition. This signal is a reward to the agent for the subsequent state. The software agent is not addressed on what to do, and it is ensured that the agent gains intelligence with the environment, situation, action, transition, and reward knowledge [18].

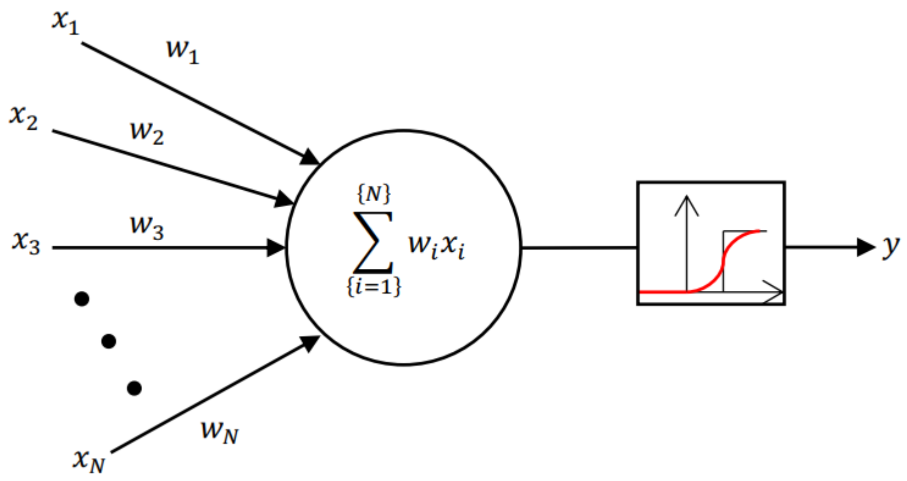


Figure 2.10 A Neuron in Artificial Neural Network

2.2.2. Artificial Neural Network

Scientific studies gain inspiration from life and the earth. A precedent of this behavior is artificial neural networks, which have a structure that illustrates the brain's logic. There are neurons in the human brain, and they communicate through synapses. A similar structure exists in artificial neural networks; each neuron builds on what the previous neuron learned.

In artificial neural networks, each neuron is a computational unit. This unit receives N inputs numbered 1 to N from other neurons or external data sources, such as a training set. One input has its data values denoted as x_i, and corresponding weight value w_i. The neuron's input is the sum of the product of associated weight and data values, $\sum_{i=1}^{N} w_i x_i = w_1 x_1 + w_2 x_2 + ... + w_N x_N$. After the summation, if the resulting value is greater than a threshold value t, the neuron's output would be 1, else 0. This equation can be expressed as $g(\sum_{i=1}^{N} w_i x_i - t)$, where g is an activation function. If the parameter for g is negative, then the output is 0; else, for the nonnegative parameter, the output is 1. However, also it depends on the activation function [19]. There are various activation functions used in artificial neural networks for different purposes. Commonly used functions are shown in Table 2. The unit structure is depicted in Figure 2.10.

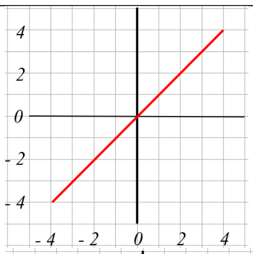
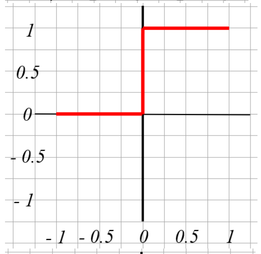
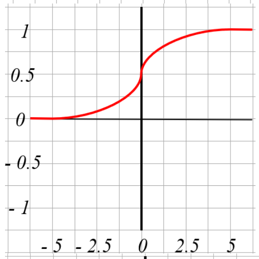
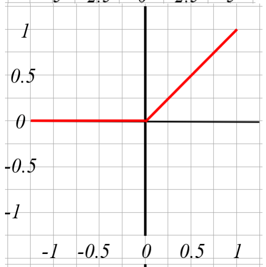
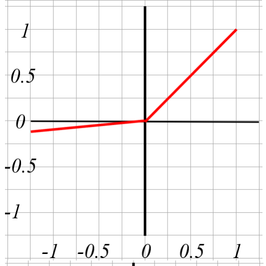
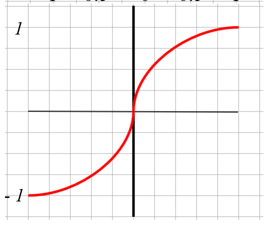
Name	**Function**	**Plot**
Linear Activation	$f(x) = x$	
Binary Step Activation	$f(x) = \begin{cases} 0, & \text{for } x < 0 \\ 1, & \text{for } x \geq 0 \end{cases}$	
Sigmoid Activation	$f(x) = \frac{1}{1+e^{-x}}$	
ReLU Activation	$f(x) = max(0, x)$	
Leaky ReLU Activation	$f(x) = max(0.1x, x)$	
TanH Activation	$f(x) = \frac{(e^x - e^{-x})}{(e^x + e^{-x})}$	

Table 2.1 Activaion Functions

ANN phenomenon has been a hotspot research topic in literature in recent years. It produces promising results in various domains, such as pattern/image/speech recognition, classification, stock market predictions, and other major branches of engineering and science as well [20]. Over the course of time, types of artificial neural networks followed an evolutionary development. The deficiency of some network types to find acceptable solutions to some problem domains led to the emergence of new network types. Today, diverse network types can be employed in distinct ANN problem domains, such as convolutional neural networks in image processing problems or recurrent neural networks in problem domains, including time dependencies. The following is a brief description of commonly used ANN types.

2.2.2.1. Feed-Forward Neural Networks

One of the simplest forms of ANN. Neurons are basic units of ANN, and if neurons are grouped, they compose a layer of networks. There are three types of layers; *input layer*, *hidden layer*, and *output layer*. The training data enters the network with the input layer. Neurons, i.e., nodes, are connected to nodes at the next layer. Nodes do some computations described above and transfers new value to the next node. If all nodes at the preceding layer are connected with all nodes at the subsequent layers, this network can be called *fully connected*; otherwise, called a *partially connected* network. The output layer's result is the network's prediction based on training data. The basic structure of the feed-forward network can be described as the input data flow by multiplying with weights.

A network can include one layer or consist of more than one layer. It is called a *single-layer feed-forward network* when it has one layer, otherwise, called a *multi-layer feed-forward network*. Figure 2.11 depicts the multi-layer feed-forward neural network.

The critical distinction between traditional computational algorithms and neural networks is the ”learning” ability of neural networks. Neural networks can learn from past experimentations and tries to give better performance at new attempts than earlier. **”Backpropagation algorithm”** is mainly used as a ”learning” way while training neural networks. Like other networks, the backpropagation algorithm is used to gain the ”learning”

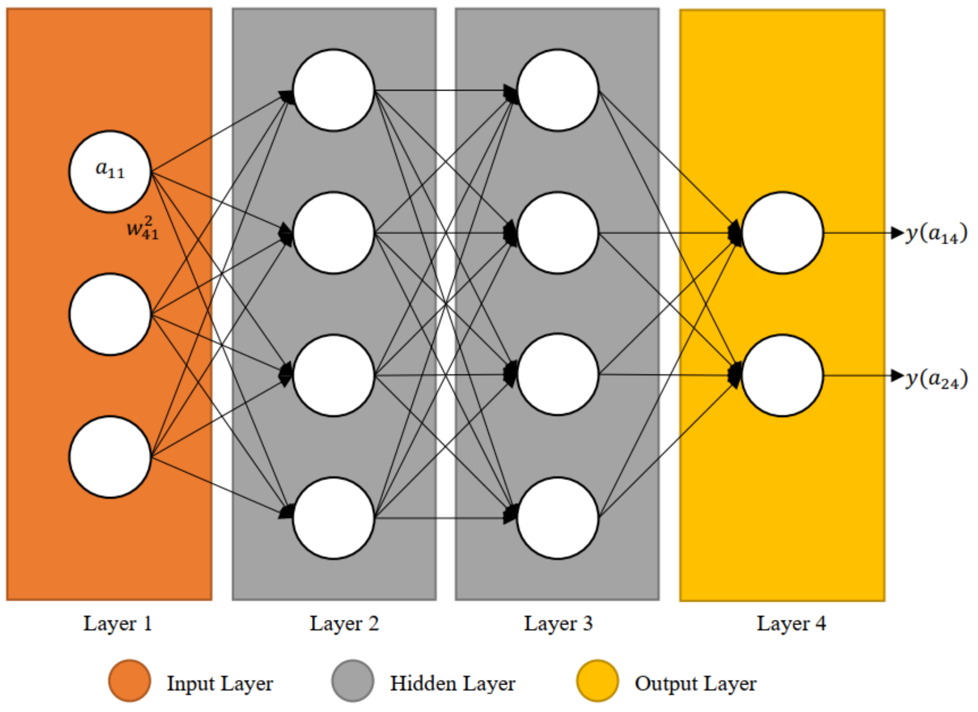


Figure 2.11 A multi-layer feed-forward neural network

ability to feed-forward neural networks. Essentially, backpropagation involves the process of updating weights in the network. The error signal is the difference between the output produced by the network and the desired output. The nodes' weights are updated by the backward propagation of the gradient vector, which is obtained after taking the derivative of the error signal according to the weight parameter [20].

Considering that "iteration" represents the training sample here, The error of neuron j at the output at iteration n can be expressed;

$$e_j(n) = x_j(n) - y_j(n) \tag{13}$$

where $x_j(n)$ is the appropriate output of neuron j, and $y_j(n)$ is the actual output.

The instantaneous value of the energy error on behalf of neuron j can be defined as $\frac{1}{2}e_j^2(n)$. Thus, the total error value of the network can be expressed in Equation 14.

$$\xi(n) = \frac{1}{2}\sum_{j\in C} e_j^2(n) \tag{14}$$

where C represents the neuron set that takes part at the output layer. Considering the train set consists of N samples, then average squared error energy is obtained, such as in Equation 15.

$$\xi_{av} = \frac{1}{N}\sum_{n\in 1}^{N} \xi(n) \tag{15}$$

Weights and biases are free parameters in the network, and both instantaneous energy error and average energy error function of all these free parameters. ξ_{av} represents the cost function for the training set used as input to the network and indicates the learning progress. The backpropagation aims to adjust the parameters according to the ξ_{av} function and minimize the cost function.

Two approaches are utilized while updating free parameters; *batch update* and *sequential update*. In the sequential update, free parameters are updated after learning from every sample in the training set. However, in the batch update, they are updated after all training samples. One complete processing of the training set is called an *epoch*.

The output of neuron j can be given as in Equation 16.

$$y_j(n) = f\left(\sum_{i=0}^{m} w_{ji}(n)y_i(n)\right) \tag{16}$$

where *f* is the nonlinear activation function, and m is the number of total inputs feeds neuron j. In this equation, bias is excluded.

Determining the changes in the neuron j's weights is related to the proportional partial derivatives of instantaneous error energy with related weight. The chain rule simplifies the statement. It can be expressed in Equation 17.

$$\frac{\partial \xi(n)}{\partial w_{ji}(n)} = \frac{\partial \xi(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial w_{ji}(n)} \tag{17}$$

End of the backpropagation, free parameters are updated, and learning can only occur after this. The update to employed to w_{ij} is created with the delta rule and is expressed as follows;

$$\Delta w_{ij} = -\eta \frac{\partial \xi(n)}{\partial w_{ij}(n)} \tag{18}$$

This backpropagation mechanism is the basis for the learning process of many neural networks.

2.2.2.2. Convolutional Neural Networks

In ANN, the network's neurons optimize themselves throughout the learning process. The neurons in convolutional neural networks, i.e. CNN, do the same optimizing as well. To do some computational operations over the raw input with neurons and produce a class score at the output layer. Also, there is a cost function to assess the network's success, such as in traditional ANN. The crucial distinction between traditional ANN and CNN is that CNN has mastery in pattern recognition in the image processing specialization. This specialization ensures the robustness and suitability of the network by providing image-specific characteristics and reducing the parameters [21].

Another essential dissimilarity between ANN and CNN is the third dimension of neurons composing the CNN layer. Along with the spatial height and width dimensions, the neuron has a depth size and refers to the activation volume. Before the completion of the network, neurons are intensified and then match the possible output class number [21]. The overall architecture is depicted in Figure 2.12

In CNN, there are three layers; convolutional layer, pooling layer, and fully-connected layer;

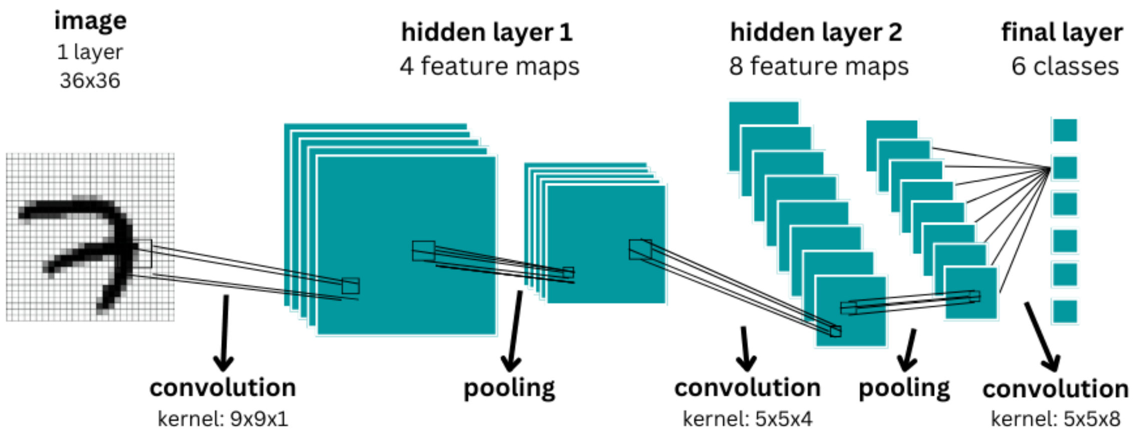


Figure 2.12 Convolutional neural network architecture

Convolutional layer It is CNN's leading player, and learnable kernels are a valuable instrument of this player. Each layer accepts feature maps generated from the earlier layers as input, convolves them with learnable kernels, and yields an output. Nonlinear functions such as ReLU are used as activation functions. The output is transformed into a feature map with the activation function and then provided as input to the following layer. This can be expressed as follows [22];

$$\mathbf{x}_j^l = f\left(\sum_{i\in M_j} \mathbf{x}_i^{l-1} * \mathbf{k}_{ij}^l + b_j^l\right) \tag{19}$$

where M_j is the set of input maps, and b is an additive bias given to each output map. In addition, the discrete convolution operation is expressed as [17],

$$(f * g)(x) = \sum_{u=\infty}^{\infty} f(u)g(x-u) \tag{20}$$

Convolution helps us see the big picture. High-level features such as borders and edges can be extracted with the help of convolution. Initially, the convolution receives low-level features such as color or gradient orientations. As this information is stacked in the following layers, the convolution layer will begin to receive high-level features. At the end of the network, the input becomes more fully meaningful.

One of the fundamental properties of CNN is that features are not required to extract manually. The kernel, i.e., filter, is being employed for this objective. Relevant filters can capture the spatial and temporal dependencies at the input. Kernels are small depending on the spatial dimensionality of the input. Nevertheless, they shift along the input and can analyze all values without being detached from the spatial dimensionality. An example of convolution is illustrated in Figure 2.13. Input data with $9x9$ size exists on the left side, and by applying a $3x3$ Sobel filter to this input, the output value is positioned in the $4x4$ size output feature map on the right.

Three hyperparameter changes the output size, thus affecting network complexity significantly; stride, padding, and depth [21].

- *Stride:* The stride specifies the pixel number to shift in the subsequent convolution. In the example above, the stride is set to 1. In such a case, too many activations will be generated for receptive fields that overlap too often, increasing complexity. Otherwise, if the value is determined more than the optimized value, it will not be able to capture the features in the input.
- *Padding:* The padding hyperparameter adjusts the boundaries of the input by padding the borders with zero values, and In the example above, the padding is set to 0. Thus, the kernel will also capture data located at the boundaries of the input and take all spatial dimensionality into account.
- *Depth:* A hyperparameter significantly affects the network's number of neurons. It corresponds to the number of filters used in the hidden layer. Changing the number of filters will also cause a significant change in the number of parameters in the network, and its complexity will change at this rate.

Let W be the input size, F be the kernel size, S be the number of steps in the stride, and P is the value to be applied for zero-padding, the spatial dimensonality of output resulting from the convolution be calculated as;

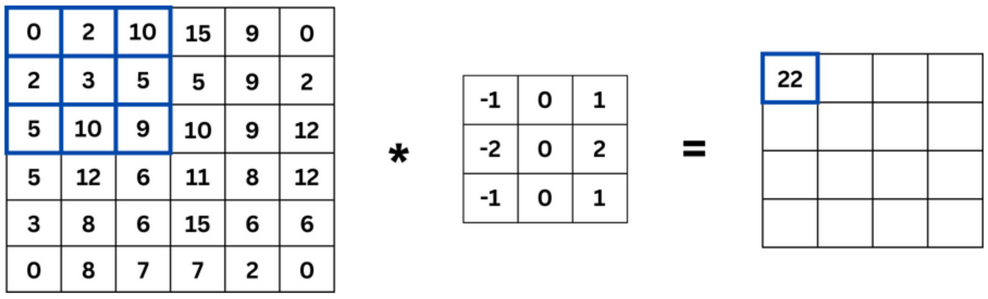


Figure 2.13 A simple convolution representation

$$\frac{(W - F) + 2P}{S + 1} \tag{21}$$

Pooling layer To make the network more manageable, reducing the parameters can be followed. The pooling layer aims to reduce the parameters by reducing the spatial dimensionality of the input, which is a representation of the input data of the network at the relevant layer. Generally, the pooling layer is employed in between the subsequent convolutional layers. Various types of computing are used in pooling layers, such as max, average, and L2 norm. MAX operation, 2x2 kernel, and 2 for stride value are commonly utilized for the pooling layers. For this setting, the pooling layer downsizes the activation map by 25

Fully connected layer It is the last layer shown in the figure. The neurons have no connection within the layer. They bind to neighboring layers and function similarly to traditional ANN [21].

2.2.2.3. Recurrent Neural Networks

In a feed-forward network, the neuron's output is input to the neuron placed in the next layer. This structure is sometimes useless, especially considering the whole learning process. For example, the stock price series depends on the former runs, and the road map of the market gives a clue for future predictions. For operating this type of problem domain, recurrent Neural Networks (RNNs) are employed.

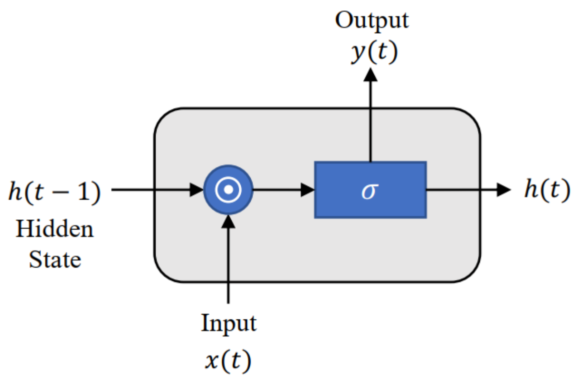


Figure 2.14 A typical cell structure of RNN

RNNs work with the principle that feeds the neuron with its produced output at the preceding step. Thus, the neuron acts like a memory cell. In this way, for example, RNN can take into account even the first word of the sentence representing a sequence of words because its weight and values are included in the computational process producing the network's output. RNNs are a kind of artificial neural network primarily used to hit on patterns in a data sequence, such as stock prices, sensors used in industry, and genome data [23]. The cell structure of RNNs is depicted in Figure 2.14. The mathematical expression of the cell is as follows [24];

$$h_t = \sigma(W_h h_{t-1} + W_x x_t + b),$$

$$y_t = h_t$$

where $x(t)$ is the input data, $y(t)$ is the output of the cell at time t, and it equals the hidden state $h(t)$, which is provided as an input to the subsequent layer. In addition, W_i and W_j denote the weights, and b denotes the bias.

The critical problem presumably to be encountered in RNN is vanishing, or exploding gradient values [25]. Especially for long sequences, small gradient values such as < 1 diminish through backpropagation computation, and finally, they have the risk of vanishing.

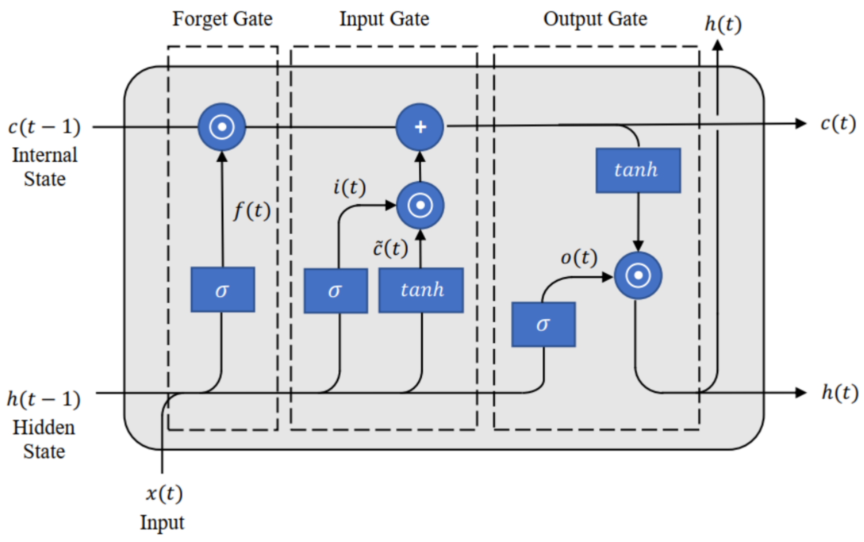


Figure 2.15 A typical cell structure of LSTM

As a consequence, the contribution of the data at the beginning of the sequence or a particular time ahead to the output will be eliminated. The same scenario is suitable for large gradient values, which will cause to exploding gradient and changes output dramatically. This weakness has been an essential origin of motivation in the emergence of LSTM networks [26].

Long Short-Term Memory Network Long Short-Term Memory Network, i.e., LSTM, is a type of RNN. LSTMs vary from to handle long–term dependencies more successfully than RNNs. Cells in the LSTM network are called memory cell because it keeps their internal state across the cell by defining their weight as 1. This kind of cell ensures that the gradient can pass throughout the network and prevents it from vanishing or exploding. Typical LSTM cell and structure are both depicted in Figure 2.15.

To overcome long-term dependency problems in LSTMs, additional information is stored in gated cell structures out of the neural network flow [26]. A gated structure enables forgetting or carrying the related data to with cell state. There are three gates in LSTM networks; input, forget, and output. Considering Figure 2.15, x_t, h_t, and y_t are denoted as input, recurrent hidden state, and output, respectively. W_x, W_h, W_i, W_o, and W_c are the weight for the related node, and b is the bias. Gates and their mathematical expressions as follows [24];

- *Input gate:* Uses tanh activation function for computing and regulates how much data needs to be used from the input node to add the cell's internal state.

$$i_t = \sigma(W_{ih}h_{t-1} + W_{ix}x_t + b_i) \tag{22}$$

$$\tilde{c}_t = tanh(W_{\tilde{c}h}h_{t-1} + W_{\tilde{c}x}x_t + b_{\tilde{c}}) \tag{23}$$

- *Forget gate:* Determines what information will be abandoned and no longer be included. This can be expressed as follows;

$$f_t = \sigma(W_{fh}h_{t-1} + W_{fx}x_t + b_f) \tag{24}$$

- *Output gate:* Determines the amount of data that will be used from that cell's output.

$$o_t = \sigma(W_{oh}h_{t-1} + W_{ox}x_t + b_o) \tag{25}$$

Finally, the internal state and hidden state of the LSTM cell at the end of the flow can be expressed as follows;

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \tag{26}$$

$$h_t = o_t - tanh(c_t) \tag{27}$$

Bi-Directional Long Short-Term Memory Network Traditional RNN architecture deals with only previous context. To overcome this point, the bi-directional RNN model, i.e. BiLSTM, introduced by [27] that work in a way that is simultaneously being trained with an isolated hidden layer in both directions [24]. The additional hidden layer is being trained from the end of the data sequence to the beginning, so the model can also reflect the data to the output without depending on the previous context [23]. The overall structure of the BiLSTM is depicted in Figure 2.16.

The LSTM cells in the architecture demonstrated in Figure 2.16 are identical to the LSTM cells demonstrated in Figure 2.15. Therefore, in the forward layers, the mathematical

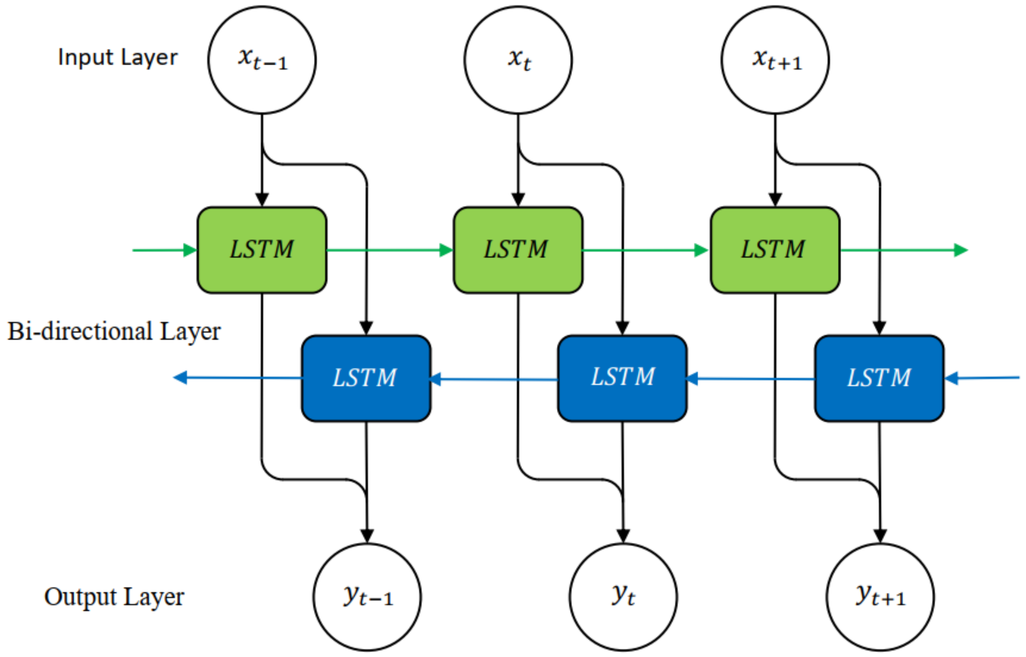


Figure 2.16 The overall structure of Bi-directonal LSTM

computations will take place as Paragraph 2.2.2.3., where the sequence inputs $(\overrightarrow{h}_t^L, \overrightarrow{c}_t^L)$ from $t = 1$ to T. In contrast, in the backward layers, data sequence flows with reverse time steps, $t = T$ to 1, and sequence inputs are as $(\overleftarrow{h}_t^L, \overleftarrow{c}_t^L)$. Considering the duplicate notions in Figure 2.15, the mathematical expressions of the cell of the backward layer, denoted as L, are as follows [24];

$$\begin{aligned}
\overleftarrow{f}_t^L &= \sigma(W^L_{\overleftarrow{f}h} h^L_{t+1} + W^L_{\overleftarrow{f}x} h^{L-1}_t + b^L_{\overleftarrow{f}}), \\
\overleftarrow{i}_t^L &= \sigma(W^L_{\overleftarrow{i}h} h^L_{t+1} + W^L_{\overleftarrow{i}x} h^{L-1}_t + b^L_{\overleftarrow{i}}), \\
\overleftarrow{\tilde{c}}_t^L &= tanh(W^L_{\overleftarrow{\tilde{c}}h} h^L_{t+1} + W^L_{\overleftarrow{\tilde{c}}x} h^{L-1}_t + b^L_{\overleftarrow{\tilde{c}}}), \\
\overleftarrow{c}_t^L &= \overleftarrow{f}_t^L \cdot \overleftarrow{c}_{t+1}^L + \overleftarrow{i}_t^L \cdot \overleftarrow{\tilde{c}}_t^L, \\
\overleftarrow{o}_t^L &= \sigma(W^L_{\overleftarrow{o}h} h^L_{t+1} + W^L_{\overleftarrow{o}x} h^{L-1}_t + b^L_{\overleftarrow{o}}), \\
\overleftarrow{h}_t^L &= \overleftarrow{o}_t^L \cdot tanh(\overleftarrow{c}_t^L)
\end{aligned} \tag{28}$$

The final output of the BiLSTM architecture is as follows;

$$y_t = W_{\overrightarrow{h}y}\overrightarrow{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y \tag{29}$$

3. RELATED WORK

Comparative studies of BSS methods have been carried out in different fields and have guided researchers from different disciplines. In particular, the blind separation of signals from health data (e.g., EEG) has been a popular topic. In [28], the authors focus on removing muscle artifacts in EEG signals. They use Empirical Mode Decomposition (EMD) based BSS, Independent Component Analysis (ICA) based BSS, and Canonical Correlation Analysis based BSS methods. In [29], the authors compare BSS methods on fMRI signals. Differently from the preceding study, the authors use Infomax (maximum likelihood) [30], FastICA, joint approximate diagonalization of eigenmatrices (JADE) [31], and eigenvalue decomposition (EVD) [32] algorithms for separation.

In [33], matrix factorization-based BSS methods were compared for audio signals. The authors have done an experimental study utilizing FastICA, Principal Component Analysis, and NMF algorithms. Their results show that FastICA performs superior in separating mixed signals with the negentropy technique for finding a maximum non-Gaussianity. In [34], on the other hand, the authors present a BSS study using the Kernel Additive Model (KAM) [35] approach apart from conventional methods. In addition, promising results have emerged with the KAM-CUST model developed by the authors.

When these studies are considered for a thorough assessment, it can be seen that classical and neural network-based approaches are not widely studied to reveal all subtleties involved. In addition, the FastICA and NMF approach is insufficient in the MIR literature. In this respect, this study aims to fill the mentioned gap.

4. USED MODELS

4.1. Classical Methods

4.1.1. FastICA

Independent component analysis, i.e., ICA, has been one of the popular topics in the literature for a long time. Signals can be mixed in various ways or mediums. For instance, in a cocktail party problem, music or talking are separated signals, and sensors perceive these separated signals as mixed. The primary goal of ICA is to extract meaningful features from mixed signals such as stock market data, sensor data, or sound waves. From another standpoint, it can also be considered dimensionality reduction or a filtering operation to determine which feature will be passed and which will still be represented in the data [5].

ICA is a version of the principal component analysis (PCA) approach [36, 37]. The main distinction between ICA and PCA is that ICA aims to optimize higher-order statistics, but PCA aims to optimize the covariance matrix [8, 37]. Thus, ICA finds independent components, whereas PCA finds uncorrelated components of the input data [8].

Two major approaches are used to solve ICA problems; statistical approaches and neural network approaches. In statistical approaches, objective functions are built on higher-order cumulants of the observed signals and then minimized [38]. Statistical ICA approaches have many popular algorithms, such as Infomax and FastICA. The primary objective of these algorithms is to extract independent elements by [8];

(i) maximizing the non-Gaussianity

(ii) minimizing the mutual information

(iii) employing the maximum likelihood (ML) estimation technique

The fast-fixed point algorithm (FastICA) is the most widespread due to its quick convergence and good separation performance [39].

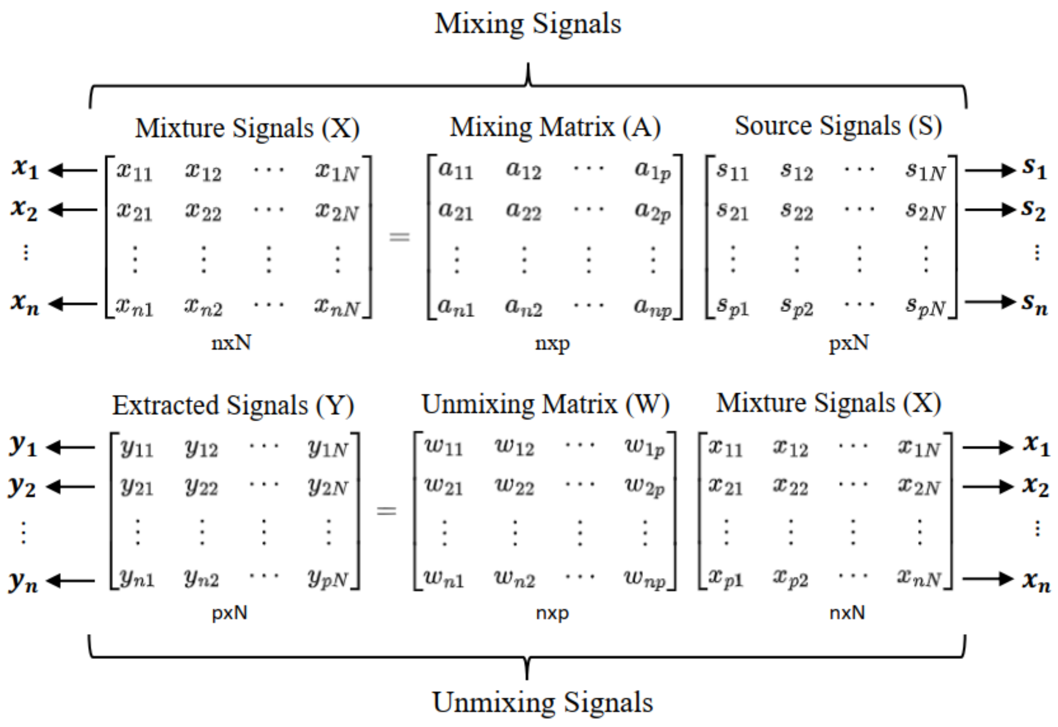


Figure 4.1 ICA mixing and unmixing block diagram

The ICA method performs a matrix multiplication to obtain the mixed output as follows:

$$x = As \tag{30}$$

where x is an observed signal vector, s is a source signal vector, and A is a constant invertible mixing matrix to be estimated. The main goal of ICA algorithms is to estimate the invertible mixing matrix A and extract the predicted source vector denoted by $\hat{s}$ with the unmixing matrix W, which is an approximation of A^{-1} as follows:

$$\hat{s} = Wx \tag{31}$$

The mixing and unmixing steps are depicted in Figure 4.1. The number of components of the mixture and signals can be different sometimes, and the number of mixtures depends on the sensors that receive and transmit the signals. As depicted in the matrix, the first component of observed signals depends on the first row of mixing matrix A, which can be expressed as $x_{11} = a_{11}s_{11} + a_{12}s_{21} + ... + a_{1p}s_{p1}$. Similar to the mixing operations; the first extracted signal can be expressed as $y_{11} = w_{11}x_{11} + w_{12}x_{21} + ... + w_{1p}x_{p1}$ [5].

ICA has two ambiguities;

- *Ambiguity of sign:* The ICA model has not been affected by changing the independent components' sign. Because, in Equation 30, both A and s are unknown, any divider at the a_j column in the mixing matrix A can easily exclude the scalar multiplier in the component s_i. Also, without affecting the generated signals, the weight in unmixing matrix W can multiply with -1 [5, 8].
- *Ambiguity of order:* Considering the Equation 30, it can be expressed as follows,

  $$x = \sum_{i=1}^{n} a_i s_i \tag{32}$$

  Similar to the ambiguity of signs, in virtue of the anonymous nature of A and s, their order can freely modify without affecting the sum of expression. Also, the weight matrix W is initialized unsystematically. While attempting to find the independent component, the weight matrix is rotated, and when it encounters the independent component, it extracts it from the observed signals. For that reason, the order of components is not taken into account [5, 8].

4.1.1.1. ICA Estimation Principles The basic concept of estimation in the ICA model is non-gaussianity as, without nongaussian, a prediction cannot be made. Keep the Central Limit Theorem in mind. According to the Central Limit Theorem, a mixture of the two independent distributions tends to be Gaussian. That is, the independent components that compose the distribution in each case must show a less gaussian distribution than the sum of distribution [8].

Assuming that the data vector x, which is a mixture of independent components, is distributed according to the ICA model and to predict one of the independent components, Equation 31 can be employed. Equation 31 represents independent components as a linear transformation of x_i, which can be expressed as; $\hat{s} = \mathbf{w}^T\mathbf{x} = \sum_{i=1} w_i x_i$ where $\mathbf{w}$ is needed to be determined. If $\mathbf{w}$ is considered the inverse of $\mathbf{A}$, this linear transformation should correspond to an independent component. $\mathbf{w}$ cannot be precisely determined because it represents the inverse of $\mathbf{A}$, and $\mathbf{A}$ is unknown. Here a good estimator can be found utilizing the central limit theorem. The underlying assumption is that $\mathbf{w}^T\mathbf{x}$ must be at least gaussian as $\mathbf{x}$, which is a sum of source signals, and the estimation can be improved by maximizing the non-gaussianity. Since x is a data vector and $\mathbf{w}$ is a weight vector, maximizing the on-gaussianity of the weight vector $\mathbf{w}$ will bring it closer to the solution [8].

More briefly, all these three ways are based on searching a rotation of unmixing matrix and projecting into it to preprocessed data [5]. Maximizing the non-gaussianity can be enhanced in 3 ways;

- By using classical statistics measures, such as kurtosis, or negentropy
- By minimizing the mutual information
- By maximum likelihood estimation

Measures of Non-Gaussianity Maximizing the non-gaussianity is vital for ICA estimation. For this reason, two measurements are utilized [40]; kurtosis and negentropy.

Kurtosis: One method of measuring the non-gaussianity is to use the fourth central moment, i.e., kurtosis. Unmixing matrix aims to maximize the kurtosis of the yielded signals to find an optimal solution to the ICA problem [7]. Although kurtosis calculation is straightforward, it is not robust for identifying non-gaussianity because of the sensitivity to outlier data [8].

The kurtosis of y is defined as;

$$K(y) = E\{y^4\} - 3(E\{y'2\})^2 \tag{33}$$

and normalized kurtosis is defined as the ratio of the fourth moment to the second moment and expressed as;

$$\hat{K}(y) = \frac{E\left[y^4\right]}{E\left[y^2\right]^2} - 3 = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^4}{(\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2)^2} - 3 \tag{34}$$

Recall y has a unit variance, and Eq. becomes to as;

$$K(y) = \hat{K}(y) = E\left[y^4\right] - 3 \tag{35}$$

In [12], the fourth moment of the Gaussian signals is defined as $(3E\left[y^3\right])^2$, thus $\hat{K}(y) = E\left[y^4\right] - 3 = E[3(E\left[y^2\right])^2] - 3 = E\left[3(1)^2)\right] - 3 = 0$, where $E\left[y^2\right] = 1$. Therefore the gaussian random variable has zero kurtosis [5]. In other words, nongaussian random variables, at least for most, have nonzero kurtosis [12]. Yielded signal's kurtosis is needed to be maximize to extract independent components from $y = \mathbf{w}^T\mathbf{x}$ [5].

Negentropy: The other measurement of the non-gaussianity is negentropy.

Initially, if the concept of entropy has to be defined, entropy indicates how many ”various” random variables are in a distribution. The greater the entropy, the more various random variables are present and the more unpredictable the distribution.

Let Y discrete random values and a_i are the possible values, entropy H of Y can be expressed as;

$$H(Y) = -\sum_i P(Y = a_i)logP(Y = a_i) \tag{36}$$

This expression can easily be adapted for continuous variables as well. It is then called differential entropy and is expressed as:

$$H(Y) = -\int f(y)logf(y)dy \tag{37}$$

where y is a random continuous variable with density $f(x)$ and H is the entropy of y.

Gaussian distributions have random variables that span a wider area. According to information theory, gaussian variables have the most scattered structure in a distribution that includes random variables with equal variance. Therefore, it will have the most considerable entropy. The random variables that compose the distribution will form a more ”peaky” structure if gathered in a particular region. Therefore, the distribution will be more predictable, and its entropy will be lower. As a result, entropy can be employed as a handy instrument to measure the level of non-gaussianness [8].

To measure non-gaussianity as well, negentropy, which is a slightly distinct version of entropy, is used and let $\mathbf{y}_{gauus}$ is normal distribution that has the identical mean and variance with $\mathbf{y}$ [41]; negentropy can be expressed as ,

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y}) \tag{38}$$

Two features of negentropy are critical;

- Always non-negative, but the pdf is definitely gaussian if it is zero.
- Invariant to invertible linear transformations.

However, calculating negentropy is a challenging operation because of its computational heaviness. Thus, an approximation of negentropy is employed. One of the useful approximations is stated in [8]. Approximation in the Eq. 39 can be obtained by using the nonquadratic function $\mathbf{G}$;

$$J(y)\alpha \Big[E\left\{G(y)\right\} - E\left\{G(v)\right\}\Big]^2 \tag{39}$$

where both v and y have zero mean and unit variance, but y is a random variable, v is a gaussian variable. The choice of G has importance there. There are two different nonquadratic function options for good approximation stated in [8].

$$G_1(u) = \frac{1}{a_1}\text{log cosh}a_1 u \quad \text{and} \quad G_2(u) = -\text{exp}(-u^2/2) \tag{40}$$

where some proper can be chosen, such $1 \leq a_1 \leq 2$.

Minimizing the Mutual Information One way of ICA estimation is minimizing the mutual information.

Mutual information is utilized for finding the dependencies between random variables. Mutual information can be expressed as;

$$I(y_q, y_2, ..., y_n) = \sum_{i=1}^{m} H(y_i) - H(\mathbf{y}) \tag{41}$$

where I is the information between the y_i, $i = 1, ..., m$ random values, and m is the scalar that indicates the number of random variables. The essential measurement of the dependencies between random variables is mutual information. It has two crucial features;

- Always nonnegative
- If and only if statistically independence of the random variables is ensured, mutual information is zero.

Therefore, it can confidently be used to figure out statistical states. In [42], a significant property of mutual information is stated for invertible linear transformations;

$$I(y_q, y_2, ..., y_n) = \sum_{i=1}^{m} H(y_i) - H(\mathbf{x}) - \text{log}\,|\text{det(y)}| \tag{42}$$

It can be written as;

$$H(\mathbf{y}) = H(\mathbf{Wx}) = H(\mathbf{x}) + \text{log}\,\big|\text{det}(\mathbf{W})\big| \tag{43}$$

where det$(\mathbf{W})$ is the determinant of $\mathbf{W}$.

Let recall negentropy and form the expression as;

$$I(y_q, y_2, ..., y_n) = C - \sum_{i=1} J(y_i) \tag{44}$$

where C is a constant that is unrelated to $\mathbf{W}$.

Mutual information yields two different ICA estimations; one of these in Equation 2, $\mathbf{W}$, have to be determined by minimizing the mutual information between the s_i which is transformed. The other one is in Equation 45, maximizing the sum of non-gaussianity for minimizing the mutual information.

Maximumum Likelihood Estimation One of the widely known information-theoretical-based ICA estimators is maximum likelihood.

The probability density function of the data set in a statistical model can be represented with the likelihood function. It can be used to find unknown parameters of the statistical model. In [43], the log-likelihood function of the nose-free ICA model is expressed as;

$$L = \sum_{t=1}^{T} \sum_{i=1}^{n} \text{log} p_i(\mathbf{w}_i \mathbf{x}(t)) + T\text{log}\left|\text{det}(\mathbf{W})\right| \tag{45}$$

where $\mathbf{W}$ is the matrix that is the inverse of $\mathbf{A}$, on the distribution of s_i known assumption, f is the pdf of s_i, and $x = 1, ..., T$ is the observed inputs. For the last part of the equation, log $\left|\text{det}(\mathbf{W})\right|$ for matrix $\mathbf{W}$, random vector $\mathbf{x}$, and probability density function p_i, the density of $\mathbf{y} = \mathbf{W}\mathbf{x}$ becomes to $p_x(Wx)\left|\text{det}(\mathbf{W})\right|$.

4.1.1.2. ICA Preprocessing Some preprocessing steps are applied, but the most widely known are; centering and whitening.

Centering The purpose of the centering phase is to subtract the data value from the mean, thus, center the data. The calculation of the centering phase is expressed as follows [5];

$$\mathbf{D} = \mathbf{X} - \mu = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} x_1 - \mu \\ x_2 - \mu \\ \vdots \\ x_n - \mu \end{bmatrix} \tag{46}$$

where $\mathbf{X}$ is a vector, μ is the mean of $\mathbf{X}$, and $\mathbf{D}$ is the vector after the centering step.

Whitening One of the preprocessing methods of ICA is whitening. After the centering step, two computations are made to whiten the data; 1) uncorrelated the observed signal by transforming the observed signal linearly, and 2) scaling each signal to have unit variance.

Lets $[\hat{x}]$ be the whiten data vector; the covariance matrix of the whiten vector $[\hat{x}]$ and the identity matrix is equal;

$$E\left\{\hat{\mathbf{x}}\hat{\mathbf{x}}^T\right\} = \mathbf{I} \tag{47}$$

One way of whitening transformation is to exploit the eigenvalue decomposition (EVD) of the covariance matrix, $E\{\hat{x}\hat{x}^T\} = VDV^T$, where V is the eigenvectors of the covariance matrix, and D is the diagonal matrix of eigenvalues. The following transformation can whiten the observed vector [8];

$$\hat{x} = VD^{-1/2}V^T x \tag{48}$$

where D can be obtained by the component-wise prcess as $D^{-1/2} = diag(d_1^{-1/2}, d_2^{-1/2}, ..., d_n^{-1/2})$. The equation transforms into a new form with Eq. 30 and Eq. 48 [8];

$$\hat{x} = VD^{-1/2}V^TAs = \hat{A}s \tag{49}$$

Hence;

$$E\{\hat{x}\hat{x}^T\} = \hat{A}E\{ss^T\}\hat{A}^T = \hat{A}\hat{A}^T = I \tag{50}$$

As seen in the expression above, the newly formed mixing matrix becomes orthogonal due to whitening the data vector [8].

After decorrelation, scaling the vectors can be done by projecting onto PCA space as $\mathbf{U} = \mathbf{VD}$. Vectors in U have a unit length. For scaling the decorrelated data to have unit variance as follows; $\hat{\mathbf{x}} = \lambda^{-\frac{1}{2}}\mathbf{U} = \lambda^{-\frac{1}{2}}\mathbf{VD}$, where λ is the eigenvalue of the covvariance matrix. The scaling phase provides becoming the data rotationally symmetric at the end of the phase.

4.1.1.3. Algorithm FastICA is based on measuring non-gaussianity of $\mathbf{w}^T\mathbf{x}$ and aims to learn to maximize it. It uses a fixed-point iteration scheme for maximizing. Measuring non-gaussianity uses the convergence specified in Eq 39 because it is computational expensiveness. For example, the nonquadratic G function specified in Eq 39 and its derivative are as follows [8];

$$g_1(u) = \tanh(a_1u) \quad \text{and} \quad G_2(u) = -\exp(-u^2/2) \tag{51}$$

With this approach, it is assumed that data had been pre-processed, as mentioned above. FastICA algorithm has the steps given in Algorithm 1, where k denote the total step number of the algorithm, which is used to stop the algorithm if the algorithm does not converge below the given threshold value.

FastICA has some advantages comparing the other ICA methods [41];

Algorithm 1 FastICA Algorithm

```
 1: for 1 to number of components $c$ do
 2:     $w_p \equiv random\ initialisation$
 3:     for 0 to $k$ do
 4:         $w_p \equiv \frac{1}{n}(Xg(W^TX) - g'(W^TX)W)$
 5:         $w_p \equiv w_p - \sum_{j=1}^{p-1}(w_p^T w_j)w_j$
 6:         $w_p \equiv w_p / \left\|w_p\right\|_2$
 7:         if $w_p <\ threshold$ then
 8:             break;
 9:         end if
10:     end for
11:     $W \equiv [w_1, w_2, ..., w_c]$
12: end for
```

- *Fast convergence:* Cubic convergence compared to the ordinary ICA
- *Simplicity:* No learning parameter
- *Generalization:* Using nonlinearity can find independent components of any non-gaussian distribution.
- *Performance:* With suitable nonlinearity g

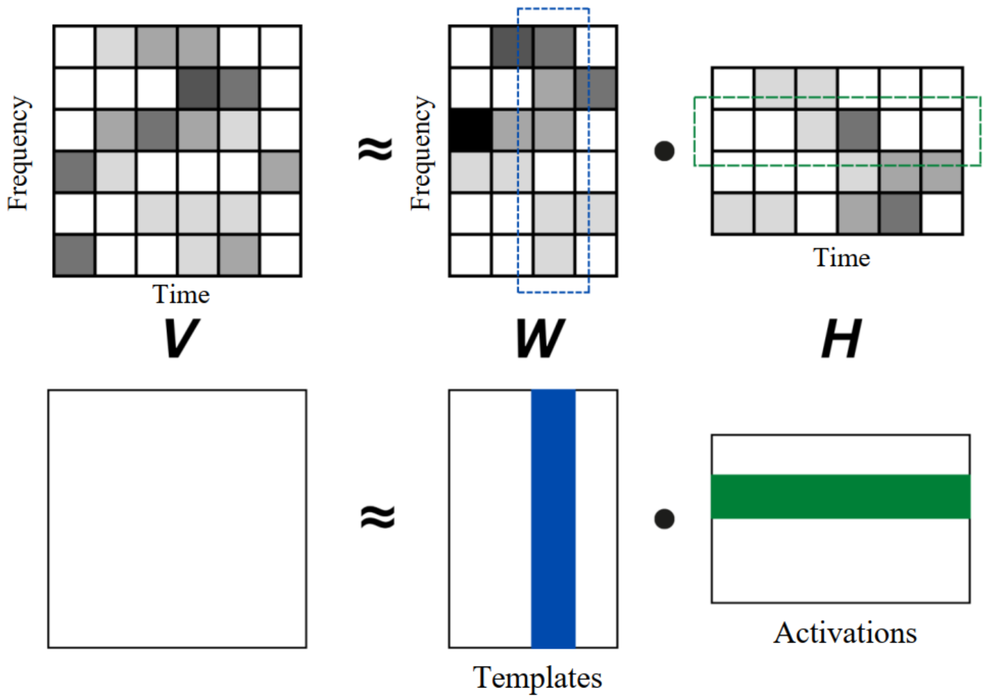


Figure 4.2 Factorization of the V matrix into non-negative matrices W and H [2]

4.1.2. Non-Negative Matrix Factorization

Unsupervised learning algorithms use matrix factorization operations. A matrix can be a representation of the significant components that compose it. Matrix factorization aims to extract these representations and then benefit them in the stages, such as analysis, from these representations. However, some constraints can be employed during factorization. Different constraints may be preferred according to different problem domains, and these extracted factors may deduce different representations of the input data [2].

Practically, there is an implication between the values of the factors and their physical meanings. If a factor is positive, it has physical significance. It has an impact area, but if it is zero or negative, it indicates that it has no physical effect on the system [2].

Non-negative matrix factorization (NMF) is a type of factorization with a non-negative constraint and factors a non-negative matrix V into two separate non-negative matrices, W and H. Figure 4.2 depicts an example for a more insightful understanding of NMF. The V matrix, an illustrative time-frequency representation of any audio source signal, consists of non-negative spectral vectors containing six time frames, $N = 6$, and six spectral coefficients, $K = 6$. The representation may appear unstructured, but factorization will result in ”prototype” four vectors, $R = 4$, that can represent spectral vectors. These vectors are weighted and superimposed accordingly. Behind the factorization, two matrices emerge a template vectors W representing spectral vectors and an activation matrix H containing the weights for this template. In this way, the source signal will become easier to process [2].

The mathematical expression of obtaining the non-negative V and the non-negative W matrix for a given V non-negative matrix can be expressed as;

$$V \approx WH \tag{52}$$

where, V is composed of N observed data vectors with K-dimensional data samples. Then, this matrix can be approximately factorized into a ($K \times R$ dimensional) W matrix called the basis vectors or template vectors and a ($R \times N$ dimensional) H matrix called the activations. The dimension parameter R here represents the *rank* of the factorization and is chosen smaller than K and N [2].

4.1.2.1. Mathematical Expression A non-negative matrix can be defined as all matrix elements equal or greater than zero. A non-negative $V \in \mathbb{R}_{\geq 0}^{KxN}$ matrix has $K \in \mathbb{N}$ rows and $N \in \mathbb{N}$ columns, and K and N are considered large. Given $R \in \mathbb{N}$ number is often less than K, and N. Matrix V can be factorized into two matrices that, $W \in \mathbb{R}_{\geq 0}^{KxR}$ and $H \in \mathbb{R}_{\geq 0}^{RxN}$, such in Eq. 52 or as follows;

$$V_{ij} \approx (WH)_{ij} = \sum_{m=1}^{R} W_{ik}H_{kj} \tag{53}$$

Considering Eq. 53, the model can be evaluated column by column, $v \approx Wh$, each data vector in the V matrix consists of the linear combination of the columns of W containing the basis vectors using the activation vector h. W represents a large number of data vectors in low dimensions. In this respect, it is crucial to correctly estimate W to find the observed data's latent structure and make a good source separation approximation [44].

The cost function is used to measure how successful the approximation is. Cost functions are based on different metrics, and the same for NMF. The two most widely used cost functions use Euclidean distance or Kullback–Leibler divergence. Both have simple implementation and convergence guarantees and are frequently used in NMF reference algorithms [4].

For $A, B \in \mathbb{R}_{KxN}$ is the two matrices that have entries A_{kn} and B_{kn} for $k \in [1 : K]$ and $n \in [1 : K]$, the square of the Euclidian distance of between these two matrices is as follows;

$$\|A - B\|^2 = \sum_{k=1}^{K} \sum_{n=1}^{N} (A_{kn} - B_{kn})^2 \tag{54}$$

Considering Eq. 54, the Euclidean distance for the NMF problem is as follows;

$$\|V - WH\|^2 \tag{55}$$

where R is the rank parameter, $V \in \mathbb{R}_{\geq 0}^{KxN}$ is a non-negative source matrix, and $W \in \mathbb{R}_{\geq 0}^{KxR}$ and $H \in \mathbb{R}_{\geq 0}^{RxN}$ are the resulting matrices after factorization.

According to the cost function, W and H should be optimized simultaneously. This case reveals a further problem that needs to be solved, the joint optimization problem, as the cost function holds two matrices that need to be optimized. Joint optimization is a challenging task due to the difficulty of computation. Firstly, the non-negativity constraint makes the problem harder to solve. Secondly, if W or H matrices are attempted to optimize separately, the function will be convex, but optimizing both matrices will lose the convexity of the function. Losing convexity will make it unattainable to reach a global minima using convexity instruments, but numerical optimization is feasible to reach at least one local minima [2].

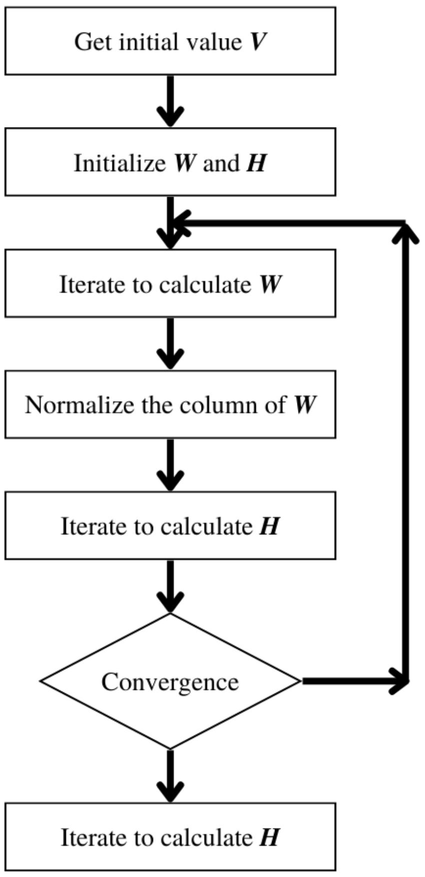


Figure 4.3 Flowchart of NMF optimization

4.1.2.2. Learning the Factorization Gradient descent can be widely employed in optimization problems and can be applied to cost function $\|V - WH\|^2$ by utilizing W and H functions. Because of the joint optimization problem's computational challenge, one method is to optimize sequentially. First, factor H is fixed, and the W matrix is learned concerning H; then, the roles change so that the logic remains the same, and factor H is learned concerning W by keeping the learned W matrix fixed [2]. The flow of the optimization is depicted as 4.3 [4].

Let express the gradient function $\phi : \mathbb{R}^D \to \mathbb{R}$ as;

$$\phi^W(H) := \|V - WH\|^2 \tag{56}$$

with setting $D := RN$ for $H \in \mathbb{R}^{RxN}$ where H has D dimensional vector space. In that gradient computation, $W_{\geq 0}^{KxR}$ is fixed. The parameter of the gradient function ϕ^W, H_{pv} has parameters $p \in [1 : R]$ and $v \in [1 : N]$. The reason for utilizing the distinct parameters from r and n for H is to differentiate the parameters used in the partial derivative expression below [2].

The partial derivative of gradient function ϕ^W to the parameter H_{pv} can be expressed as;

$$\frac{\partial \phi^W}{\partial H_{pv}} = \frac{\partial (\sum_{i=1}^{K} \sum_{n=1}^{N} (V_{kn} W_{kr} H_{rn})^2)}{\partial H_{pv}} \tag{57}$$

end of the partial derivate computation, the expression is obtained as;

$$\frac{\partial \phi^W}{\partial H_{pv}} = 2((W^T W H)_{pv} - (W^T V)_{pv}) \tag{58}$$

Considering $H^{(0)} \in \mathbb{R}^{RxN}$ as a random starting point, reducing the Euclidean distance with the additive update rule can be obtained with the help of Eq. 58;

$$H_{rn}^{(l+1)} = H_{rn}^{(l)} - \gamma_{rn}^{(l+1)} \cdot ((W^T W H^{(l)})_{rn} - (W^T V)_{rn}) \tag{59}$$

,for $l = 0, 1, 2, ...$ and $\gamma_{rn}^{(l)} \geq 0$ is some appropriate variable called as step size [2].

After finding the learned H factor, the same computations are done for the W factor, and similarly for a random starting point $W^{(0)} \in \mathbb{R}^{KxR}$, the following reduce the Euclidean distance;

$$W_{kr}^{(l+1)} = W_{kr}^{(l)} - \gamma_{kr}^{(l+1)} \cdot ((W^{(l)} H H^T)_{kr} - (V H^T)_{kr}) \tag{60}$$

for $l = 0, 1, 2, ...$ and $\gamma_{rn}^{(l)} \geq 0$ is some appropriate value for step size [2].

Although the gradient descent method is widely used, there are still some issues for NMF. These are;

- Ambiguity about choosing step size that guarantees convergence.
- Unclear behavior at joint local minima.
- Non-negativity constraint have yet to be handled.

In [44], the authors propose a therom for solving these issues called *multiplicative update rule*. Multiplicative update rules are obtained by setting the step size parameter in the additive update rules [2]. More exactly, the step size is set;

$$\gamma_{rn}^{(l)} := \frac{H_{rn}^{(l)}}{(W^T W H^{(l)})_{rn}} \tag{61}$$

Thus, the update rule in Eq. 59 becomes;

$$\begin{aligned} H_{rn}^{(l+1)} &= H_{rn}^{(l)} - \frac{H_{rn}^{(l)}}{(W^T W H^{(l)})_{rn}} \cdot \Big((W^T W H^{(l)})_{rn} - (W^T V)_{rn} \Big) \\ &= H_{rn}^{(l)} \, \frac{(W^T V)_{rn}}{(W^T W H^{(l)})_{rn}} \end{aligned} \tag{62}$$

As well step size for W;

$$\gamma_{kr}^{(l)} := \frac{W_{kr}^{(l)}}{(W^{(l)} H H^T)_{kr}} \tag{63}$$

then Eq. 60 becomes;

$$W_{kr}^{(l+1)} = W_{kr}^{(l)} \, \frac{(V H^T)_{kr}}{(W^{(l)} H H^T)_{kr}} \tag{64}$$

NMF algorithm has the steps given in Algorithm 2. Similar to Algorithm 1, the hyperparameter k indicates the total step number of the algorithm.

Algorithm 2 NMF Algorithm

```
1: $W^{(0)}, H^{(0)} \equiv random\ initialisation$
2: $l = 0$
3: for l to $k$ do
4:     $H^{(l+1)} \leftarrow H^{(l)} \frac{(W^{(l)})^T V}{(W^{(l)})^T W^{(l)} H^{(l)}}$
5:     $W^{(l+1)} \leftarrow W^{(l)} \frac{V (H^{(l+1)})^T}{W^{(l)} H^{(l)} (H^{(l+1)})^T}$
6:     if $\left\| H^{(l)} - H^{(l-1)} \right\|_2 \leq \varepsilon$ and $\left\| W^{(l)} - W^{(l-1)} \right\|_2 \leq \varepsilon$ then
7:         $H \leftarrow H^{(l)}$
8:         $W \leftarrow W^{(l)}$
9:         break;
10:    end if
11:    $l = l + 1$
12: end for
```

4.1.3. Degenerate Unmixing Estimation Techniques

There is a fundamental assumption in the classical methods mentioned earlier. Signal receivers were more than signal sources. This blind source separation medium is called a degenerate. Degenerate refers to more sources than mixtures in the environment, and blind source separation became challenging when the mixing matrix could not be inverted. Therefore, the methods which separate sources by inverting the mixing matrix can not solve under-determined problems. Degenerate Unmixing Estimation Technique, shortly DUET, uses time-frequency representations of the mixture to obtain attenuation and delay values for estimating independent components [45].

The primary intuition on which Degenerate Unmixing Estimation Techniques, DUET, are based can be expressed as follows. Source signals that do not overlap much in the time-frequency representation can be extracted from the given two anechoic mixture signals, and it doesn't matter how many source signals there are. The medium, as mentioned above, already applies to speech signals [45].

The disjointness assumption is essential to DUET. The time-frequency domain contains the source signals' separated interpretation if the signals are disjoint. After correctly partitioning the time-frequency plane, DUET can extract the source signal from the anechoic mixture signals. Although disjointness cannot be achieved in simultaneous speeches, the signal energy that dominates the time-frequency plane belongs to one of the source signals. Thus, the speech signals that make up the source signals can be separated [45].

4.1.3.1. Assumptions

Anechoic Mixing Considering $s_n(t) \in 1,..,N$ are the original signals, mixtures are received from two anechoic signal sensors, e.g., microphone, pair. If the path is only direct, the first mixture's attenuation and delay parameters can be absorbed into the source characterization. Let the number of the sources be denoted as N, γ_n is denoted as the coming delay between the microphones, and d_n is the comparable attenuation of the routes between sensors and sources [45].

$$x_1(k) = \sum_{n=1}^{N} s_n(k) \tag{65}$$

$$x_2(k) = \sum_{n=1}^{N} d_n s_n(k - \gamma_n) \tag{66}$$

W-Disjoint Orthogonality One of the essential assumptions for DUET is the W-disjoint orthogonality assumption. For a given functions $s_n(k)$ and $s_m(k)$, and windowed Fourier transform function, $W(k)$. Then two supports $\hat{s}_n(k)$ and $\hat{s}_m(k)$, are obtained from $s_n(k)$ and $s_m(k)$, after being transformed by $W(k)$. If $\hat{s}_n(k)$ and $\hat{s}_m(k)$, are orthogonal, $s_n(k)$ and $s_m(k)$ are called w-disjoint orthogonal. Firstly, in the following expression, the windowed Fourier transform function of $s_n(k)$ is;

$$\hat{s}_n(\upsilon, \rho) := F^W[s_n](\upsilon, \rho) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} W(k - \upsilon) s_n(k) e^{-i\rho k} dk \tag{67}$$

Then the W-disjoint orthogonality may be more precisely expressed as;

$$\hat{s}_n(\upsilon, \rho) \hat{s}_m(\upsilon, \rho) = 0, \qquad \forall \upsilon, \rho, \qquad \forall n \neq m \tag{68}$$

W-disjoint orthogonality also indicates the useful feature of the signals which is the disjointness of the supports of them in the time-frequency (TF) representation. Thus, it constitutes the mathematical expression of the assumption that the dominant signal energy belongs to one source in forming the TF representation of the mixture signal.

For appropriate separation of TF representation of the mixture signal, the mask function is specified, and the fractionating is conducted according to this mask function. Orthogonality is also employed to specify the mask function. The mask function can be represented as;

$$M_n(\upsilon,\rho) := \begin{cases} 1, & \hat{s}_n(\upsilon,\rho) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{69}$$

Then the source $s_n(k)$ is partitioned from the mixture signal as follows.

$$\hat{s}_n(\upsilon,\rho) = M_n(\upsilon,\rho)\hat{x}_1(\upsilon,\rho), \qquad \forall \upsilon,\rho \tag{70}$$

In this regard, it is crucial to correctly specify the masking function for successfully dividing the TF representation [45].

Local Stationarity Fourier transform can be stated for the pair as a widely known as;

$$s_n(k-\gamma) \leftrightarrow e^{-i\rho\gamma}\hat{s}(\rho) \tag{71}$$

Regarding the notation in Eq. 67

$$F^W\left[s_n(\cdot-\gamma)\right](\upsilon,\rho) = e^{-i\rho\gamma}F^W\left[s_n(\cdot)\right](\upsilon,\rho) \tag{72}$$

where $W(k) \equiv 1$. $W(k)$, a windowing function, Eq.72 is not needed to be true. Considering the Hamming windows function, for example, to think signal windows separated by a few seconds are associated with the shift of phase, there is no reason. From the DUET perspective, this is called local stationary. Let Δ denoted as the maximum probable time change in the mixing model Eq. 72 holds $\forall\gamma$, $|\gamma| \leq \Delta$ [45]. Local stationary can be expressed more formally as;

$$F^W\left[s_n(\cdot-\gamma)\right](\upsilon,\rho) = e^{-i\rho\gamma}F^W\left[s_n(\cdot)\right](\upsilon,\rho), \quad \forall\gamma, |\gamma| \leq \Delta \tag{73}$$

Microphones Close Together Estimating delay and attenuation parameters accurately from the TF point is a critical step of DUET. The local stationary assumption is utilized for this estimation and forms the delay in time to the multiplicative parameter. To avoid phase ambiguity because of phase-wrap, the multiplicative parameter $e^{-i\rho\gamma}$ uniquely identifies γ, if $|\rho\gamma| < \pi$ so that this constraint can be defined as;

$$|\rho\gamma_n| < \pi, \quad \forall\rho, \forall n \tag{74}$$

If microphones are close together to meet the distance requirement of less than $\pi l/\rho_{max}$, where the speed of sound is denoted as l and the sources' maximum frequency is denoted as ρ_{max}, this assumption is satisfied [45].

Different Spatial Signatures DUET relies on the relativity of two parameters which delay and attenuation. If the spatial characteristics of source signals are identical, then their mixing parameters are also the same, and they do not change the mixture spatial characteristics uniquely. In such cases, DUET may not be able to parse correctly and fail as these mixing parameters are significantly critical to DUET. This constraint can be expressed in mathematical form as;

$$(d_n \neq d_m) \text{ or } (\gamma_n \neq \gamma_m), \quad \forall n \neq m \tag{75}$$

4.1.3.2. Algorithm In this paragraph, the assumptions will be employed separately. Considering the assumptions of local stationary and anechoic mixing, the expressions in Eq. 65 and Eq.66 can be arranged as follows;

$$\begin{bmatrix} \hat{x}_1(\upsilon,\rho) \\ \hat{x}_2(\upsilon,\rho) \end{bmatrix} = \begin{bmatrix} 1 & \cdots & 1 \\ d_1 e^{-i\rho\gamma_1} & \cdots & d_N e^{-i\rho\gamma_N} \end{bmatrix} \begin{bmatrix} \hat{s}_1(\upsilon,\rho) \\ \vdots \\ \hat{s}_n(\upsilon,\rho) \end{bmatrix} \tag{76}$$

Afterward, continuing with the W-disjoint orthogonality assumption, the idea that the energy of one source signal is dominant at each (ρ, υ) the mixing procedure can be represented as follows;

$$\text{for each } (\upsilon, \rho), \quad \begin{bmatrix} \hat{x}_1(\upsilon, \rho) \\ \hat{x}_2(\upsilon, \rho) \end{bmatrix} = \begin{bmatrix} 1 \\ d_n e^{-i\rho\gamma_n} \end{bmatrix} \hat{s}_n(\upsilon, \rho) \quad \text{for some } n \tag{77}$$

,where n indicates the active sources at (ρ, υ). DUET uses the observation that the ratio in the TF representation relies on the mixing parameter of the dominant source signal at (ρ, υ) rather than the source signals.

$$\forall(\rho, \upsilon) \in \psi_n, \quad \frac{\hat{x}_2(\upsilon, \rho)}{\hat{x}_1(\upsilon, \rho)} = d_n e^{-i\rho\gamma_n} \tag{78}$$

where

$$\psi_n := \{(\upsilon, \rho) : \hat{s}_n(\upsilon, \rho) \neq 0\} \tag{79}$$

The mixing parameters of the TF points can be computed as represented;

$$\tilde{d}(\upsilon, \rho) := \left|\hat{x}_2(\upsilon, \rho)/\hat{x}_1(\upsilon, \rho)\right| \tag{80}$$

$$\tilde{\gamma}(\upsilon, \rho) := (-1/\rho)\angle(\hat{x}_2(\upsilon, \rho)/\hat{x}_1(\upsilon, \rho)) \tag{81}$$

Since the mixing parameters are found, the indicator functions that will produce the binary masks can be described as follows;

$$M_n(\upsilon, \rho) := \begin{cases} 1, & (\tilde{d}(\upsilon, \rho), \tilde{\gamma}(\upsilon, \rho)) = (d_n, \gamma_n) \\ 0, & \text{otherwise} \end{cases} \tag{82}$$

and the mixture can be demixed by this mask. However, assumptions may not always be met. In this case, the local parameters of the mixing cannot be predicted correctly. In such a circumstance, the definition of ψ in Eq.79 would revise to;

$$\psi_n := \left\{(\upsilon,\rho) : \left|\hat{s}_n(\upsilon,\rho)\right| \gg \left|\hat{s}_m(\upsilon,\rho)\right|, \quad \forall m \neq n\right\} \tag{83}$$

and then,

$$\forall(\rho,\upsilon) \in \psi_n, \quad \frac{\hat{x}_2(\upsilon,\rho)}{\hat{x}_1(\upsilon,\rho)} \approx d_n e^{-i\rho\gamma_n} \tag{84}$$

Thus, a cluster around the local mixing parameter will be obtained, and this cluster will be separable as long as it satisfies the assumption of not being spatially identical [45].

Two-Dimensional Smoothed Weighted Histogram Some cases where the assumptions are not fully satisfied, the mechanism changes to the clustering structure. The relevant attenuation and delay parameters are clustered. For d_n and γ_n, the maximum likelihood estimators are employed. Considering the following model;

$$\begin{bmatrix}\hat{x}_1(\upsilon,\rho)\\ \hat{x}_2(\upsilon,\rho)\end{bmatrix} = \begin{bmatrix}1\\ d_n e^{-i\rho\gamma_n}\end{bmatrix}\hat{s}_n(\upsilon,\rho) + \begin{bmatrix}\hat{\upsilon}_1(\upsilon,\rho)\\ \hat{\upsilon}_2(\upsilon,\rho)\end{bmatrix}, \quad \forall(\upsilon,\rho) \in \psi_n \tag{85}$$

,where $\hat{\upsilon}_1$ and $\hat{\upsilon}_2$ denotes the noise that represent not satisfied assumptions. The other term, *symmetric attenuation*, is utilized instead of estimating d_n as before. Now, estimating as;

$$\phi_n := d_n - \frac{1}{d_n} \tag{86}$$

If the sensor, i.e., microphone, signals are swapped, the reflection of the attenuation will be around a center ($\phi = 0$) and symmetrical; for that reason, it is called symmetric attenuation. Thus, estimating the local symmetric attenuation parameter can be defined as;

$$\tilde{\phi}(\upsilon,\rho) := \left|\frac{\hat{x}_2(\upsilon,\rho)}{\hat{x}_1(\upsilon,\rho)}\right| - \left|\frac{\hat{x}_1(\upsilon,\rho)}{\hat{x}_2(\upsilon,\rho)}\right| \tag{87}$$

Following the maximumum likelihood estimator in [46], the estimators can be defined as;

$$\tilde{\phi}_n = \frac{\iint_{(\upsilon,\rho)\in\psi_n} \left|\hat{x}_1(\upsilon,\rho)\hat{x}_2(\upsilon,\rho)\right|^p \rho^q \tilde{\phi}(\upsilon,\rho)\mathrm{d}\upsilon\mathrm{d}\rho}{\iint_{(\upsilon,\rho)\in\psi_n} \left|\hat{x}_1(\upsilon,\rho)\hat{x}_2(\upsilon,\rho)\right|^p \rho^q \mathrm{d}\upsilon\mathrm{d}\rho} \tag{88}$$

and

$$\tilde{\gamma}_n = \frac{\iint_{(\upsilon,\rho)\in\psi_n} \left|\hat{x}_1(\upsilon,\rho)\hat{x}_2(\upsilon,\rho)\right|^p \rho^q \tilde{\gamma}(\upsilon,\rho)\mathrm{d}\upsilon\mathrm{d}\rho}{\iint_{(\upsilon,\rho)\in\psi_n} \left|\hat{x}_1(\upsilon,\rho)\hat{x}_2(\upsilon,\rho)\right|^p \rho^q \mathrm{d}\upsilon\mathrm{d}\rho} \tag{89}$$

which p and q are parameters of the expressions; in this way, the expressions are parameterized. The p and q parameters can be chosen in various ways, but rather than picking the parameters, estimators need the support of the TF representation of the source signal. This requirement makes the solution difficult, but estimates form clusters around the delay mixing parameters and the proper symmetric attenuation, as stated in the previous paragraph. The two estimators in Eq.88 and Eq.89 suggest a two-dimensional weighted histogram construction to specify the clusters, then estimate the parameters, (d_n, γ_n). The histogram plays a significant role in localization and separation. The center of the resulting cluster points on the histogram constructed using indices, $(\tilde{\phi}(\upsilon,\rho), \tilde{\gamma}(\upsilon,\rho))$, and weights, $\left|(\hat{x}_1(\upsilon,\rho), \hat{x}_2(\upsilon,\rho)\right|^p \rho^q$, will represent the exact mixing parameters. Thus, for a sufficiently separated (d_n, γ_n) mixing parameter from each other, if a window is chosen big sufficiently to perceive the additions of one source signal, N different peaks will emerge, and this emergence provides the anechoic mixing parameters[45].

Firstly, for the formal definition of two dimensional smoothed histogram, the points that contribute to the histogram is needed to be defined. Let Δ_ϕ, Δ_γ are the smoothing resultant widths;

$$I(\phi,\gamma) = \left\{(\upsilon,\rho) : \left|\tilde{\phi}(\upsilon,\rho) - \phi\right| < \Delta_\phi, \left|\tilde{\gamma}(\upsilon,\rho) - \gamma\right| < \Delta_\gamma\right\} \tag{90}$$

When all assumptions are fulfilled, the histogram can be defined as:

$$H(\phi,\gamma) = \begin{cases} \iint \left|\hat{s}_n(\upsilon,\rho))\right|^{2p} \rho^q \mathrm{d}\upsilon \mathrm{d}\rho, & |\phi_n - \phi| < \Delta_\phi, |\gamma_n - \gamma| < \Delta_\gamma \\ 0, & \text{otherwise} \end{cases} \tag{91}$$

Separating the Sources By locating the peaks In the histogram by different methods, such as k-means and peak tracking [47], mixing parameters can be found. After the mixing parameters are determined, creating the mask to separate the sources from the TF representation is performed. Each TF point is assigned to the peak that the nearest parameter estimates deducted from the TF point. Symmetric attenuation is transformed to attenuation as;

$$\tilde{d}_n = \frac{\tilde{\phi} + \sqrt{\tilde{\phi}_n^2 + 4}}{2} \tag{92}$$

where (ϕ_n, γ_n), $n = [1 : N]$ is peak centers of histogram. Assigning peak to TF points by;

$$J(\upsilon,\rho) := \underset{k}{\text{argmin}} \frac{\left|\tilde{d}_k e^{-i\tilde{\gamma}_k\rho}\hat{x}_1(\upsilon,\rho) - \hat{x}_2(\upsilon,\rho)\right|^2}{1 + \tilde{d}_k^2} \tag{93}$$

Then, each TF point is assigned to an estimation of mixing parameter by;

$$\tilde{M}_n(\upsilon,\rho) := \begin{cases} 1, & J(\upsilon,\rho) = n \\ 0, & \text{otherwise} \end{cases} \tag{94}$$

And the last step is separating each TF representation of source signals by;

$$\tilde{\hat{s}}_n = \tilde{M}_n(\upsilon, \rho) \left(\frac{\hat{x}_1(\upsilon, \rho) + \tilde{d}_n e^{-i\tilde{\gamma}_k \rho} \hat{x}_2(\upsilon, \rho)}{1 + \tilde{d}_n^2} \right) \tag{95}$$

After obtaining TF representation of the source signals, these representations are converted back to the time domain [45].

The overall steps for the DUET algorithm are as follows [45];

1. Build time-frequency representations from mixtures; $\hat{x}_1(\upsilon, \rho) \leftarrow x_1(k)$, $\hat{x}_2(\upsilon, \rho) \leftarrow x_2(k)$
2. $\phi \leftarrow \left|\frac{\hat{x}_2(\upsilon,\rho)}{\hat{x}_1(\upsilon,\rho)}\right| - \left|\frac{\hat{x}_1(\upsilon,\rho)}{\hat{x}_2(\upsilon,\rho)}\right|$, $\gamma \leftarrow \frac{-1}{\rho} \angle \left|\frac{\hat{x}_2(\upsilon,\rho)}{\hat{x}_1(\upsilon,\rho)}\right|$
3. Build two dimensional smoothed weighted histogram, $H(\phi, \gamma)$
4. Find peaks and peak centers and determine the actual mixing parameters
5. For each peak in the histogram, build a time-frequency binary mask
6. Apply masks to mixtures and get each source's time-frequency representation
7. Convert the time-frequency representations of the sources to the time domain.

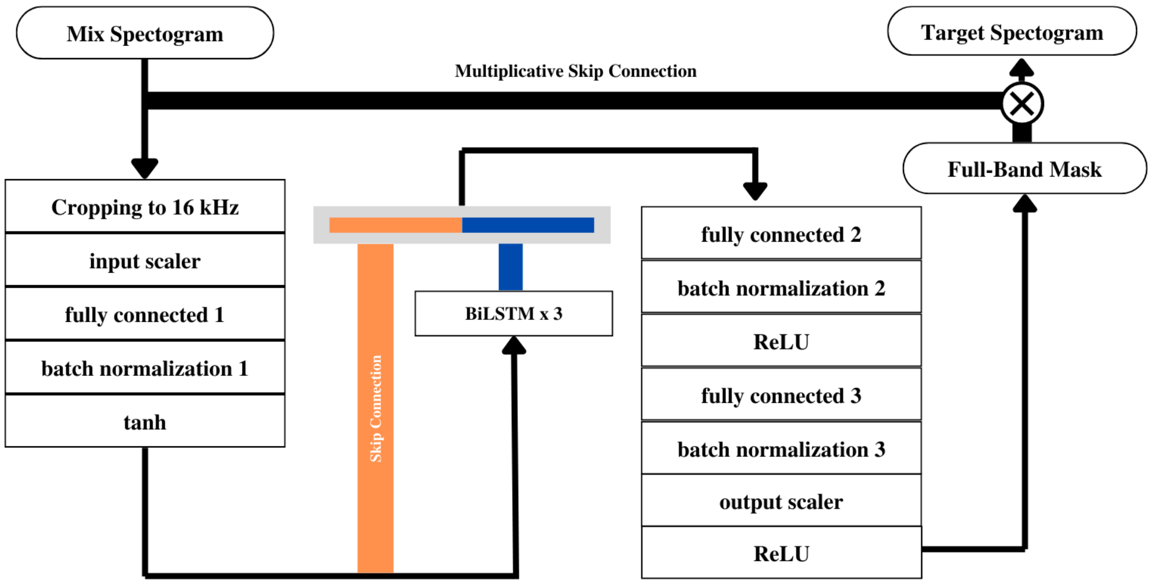


Figure 4.4 Open-Unmix Model

4.2. Machine Learning Methods

4.2.1. Open-Unmix

In addition to obtaining state-of-art results in music source separation studies, the authors also aim for Open-Unmix to be a basis for these studies. With the the structure, which is similar to MNIST, of the project, the dataset can be downloaded instantly, and training can be started immediately [48]. To separate sources with the Open-Unmix model, it is necessary to train the model separately for each source. Considering the fact that there is a separate model output for each source, this creates flexibility on the one hand and a disadvantage on the other. Figure 4.4 depicts the Open-Unmix model, and explanations of the model are as follows: Open-Unmix starts with a fully connected layer, batch normalization, and hyperbolic tangent activation function, followed by three Bi-LSTM layers and continues with two fully connected layers with batch normalizations and ReLU activation functions. At the end of the model, the output of ReLU activation multiplies by spectrogram values, and the predicted source is obtained. Some important notes about the model are as follows; [49];

- Fully connected layers before Bi-LSTM layers are helpful for dimensionality reduction to provide more distilled input data to Bi-LSTM layers while the time dimension stays the same.
- Skip connections aid convergence and allow the network to learn without the help of layers. At the same time, multiplying the input spectrogram by the output of the last ReLU activation shows that this output is the weight of the predicted source on the original spectrogram, not the spectrogram data of the predicted source [50].

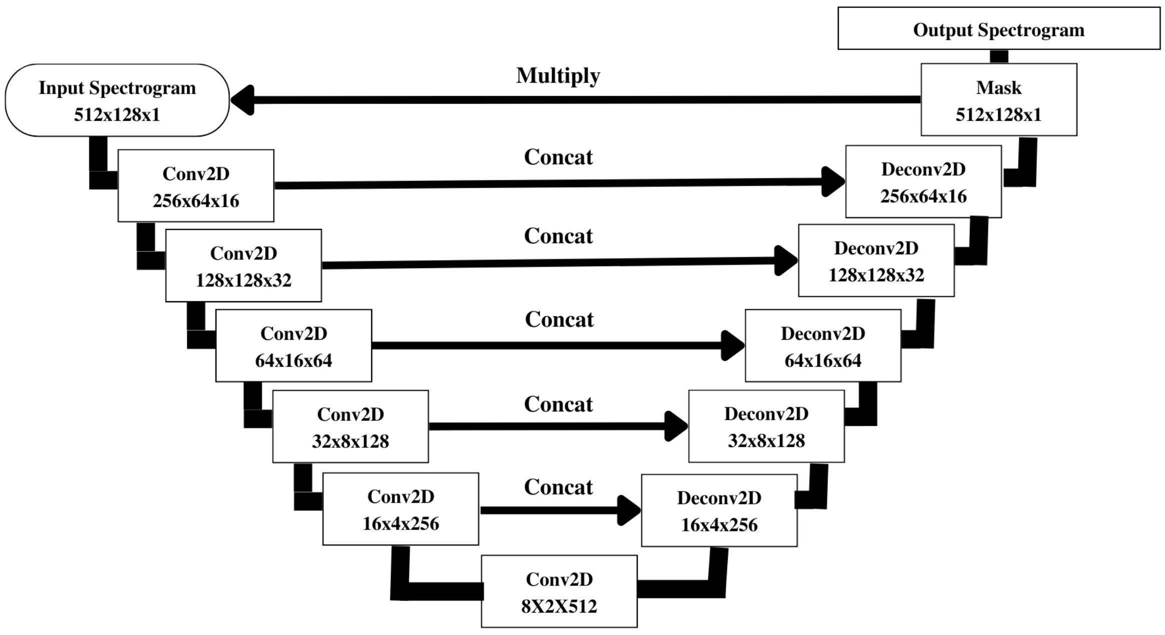


Figure 4.5 U-Net Architecture

4.2.2. Spleeter

Spleeter is an implementation of the U-Net [51] architecture. U-Net was found for biomedical image segmentation [52], later used in audio source separation studies, and after successful results, it has become one of the popular architectures in this field.

The input of the network is the spectrogram of audio. The network first processes this spectrogram with two-dimensional convolutions, and in each convolution, a smaller encoded output of the previous input emerges, and finally, latent data is obtained. This latent data is then reconstructed by decoding with two-dimensional deconvolution layers. Each deconvolution layer has the same shape as its corresponding convolution layer, and its output is concatenated with the data encoded in the corresponding convolution layer. Finally, the latent data is rescaled, and the mask applied to the spectrogram is created to predict the source [49] The U-Net model is depicted in Figure 4.5.

U-Net uses, firstly, six convolution layers for encoding and five deconvolutional layers for decoding. These layers are strides of two and have a 5×5 kernel size. Batch normalization and ReLU activation functions are used between all encoder and decoder layers. However, the output layer has softmax activation as it builds a mask for the spectrogram.

The variation between Spleeter and the original U-Net is that the audio is processed in stereo, not mono. It works with two-channel audio as input and output (channel, time steps, frequency bins), and the architecture remains identical [53].

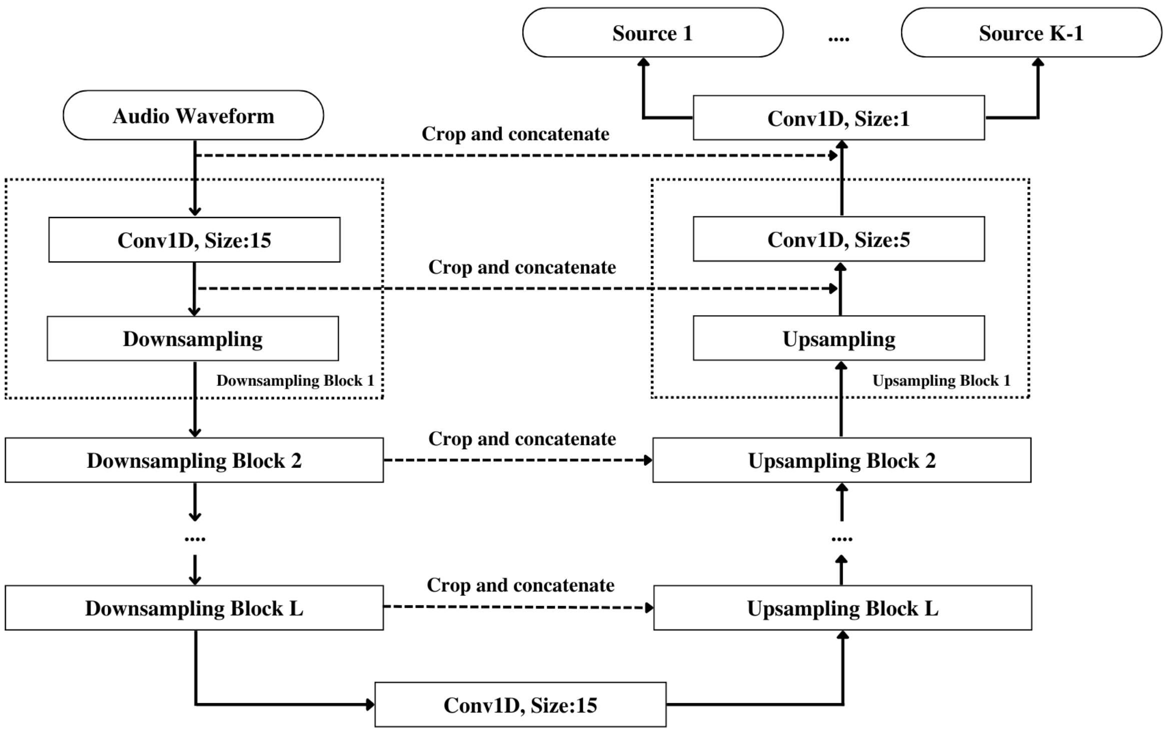


Figure 4.6 Wave-U-Net Architecture

4.2.3. Wave-U-Net

Wave-U-Net, a modification of U-Net, separates audio sources directly using waveforms instead of the spectrogram [49]. Figure 4.6 depicts Wave-U-Net architecture. Wave-U-Net uses a similar convolution/deconvolution sequence; however, it processes on waveform instead of the spectrogram. It performs source separation by applying one-dimensional convolution/deconvolution operations on audio signals. Again, similar to U-Net, it concatenates the corresponding encoded and decoded data.

Another dissimilarity in the U-Net model is the output layer. The model generates output for each source by applying convolutional filters, and hyperbolic tangent activations to the last feature map [54].

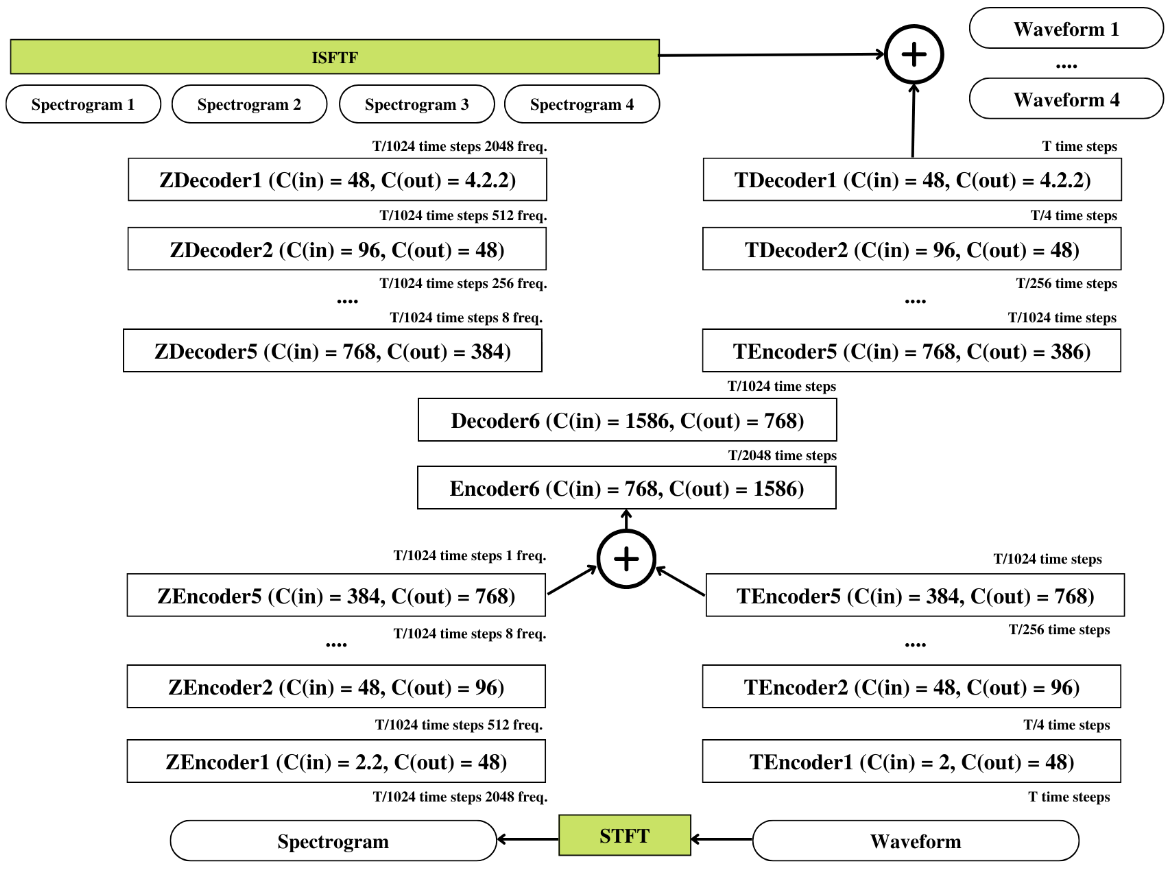

ISFTF
Waveform 1
....
Waveform 4
Spectrogram 1
Spectrogram 2
Spectrogram 3
Spectrogram 4
T/1024 time steps 2048 freq.
ZDecoder1 (C(in) = 48, C(out) = 4.2.2)
T time steps
TDecoder1 (C(in) = 48, C(out) = 4.2.2)
T/1024 time steps 512 freq.
ZDecoder2 (C(in) = 96, C(out) = 48)
T/4 time steps
TDecoder2 (C(in) = 96, C(out) = 48)
T/1024 time steps 256 freq.
T/256 time steps
....
T/1024 time steps 8 freq.
ZDecoder5 (C(in) = 768, C(out) = 384)
T/1024 time steps
TEncoder5 (C(in) = 768, C(out) = 386)
T/1024 time steps
Decoder6 (C(in) = 1586, C(out) = 768)
T/2048 time steps
Encoder6 (C(in) = 768, C(out) = 1586)
T/1024 time steps 1 freq.
ZEncoder5 (C(in) = 384, C(out) = 768)
T/1024 time steps
TEncoder5 (C(in) = 384, C(out) = 768)
T/1024 time steps 8 freq.
T/256 time steps
ZEncoder2 (C(in) = 48, C(out) = 96)
TEncoder2 (C(in) = 48, C(out) = 96)
T/1024 time steps 512 freq.
T/4 time steps
ZEncoder1 (C(in) = 2.2, C(out) = 48)
TEncoder1 (C(in) = 2, C(out) = 48)
T/1024 time steps 2048 freq.
T time steeps
Spectrogram
STFT
Waveform

Figure 4.7 Hybrid Demucs Architecture

4.2.4. Hybrid Demucs

In audio source separation studies, the methods work over waveform or spectrogram. Similarly, the original Demucs [55] only performed audio source separation with waveform input. Hybrid Demucs, on the other hand, adds the spectral domain to the original architecture and provides analysis in multi-domain [56]. The overall architecture is depicted in Figure 4.7. The original Demucs has the U-Net structure and, unlike U-Net, it consists of two Bi-LSTM layers in the middle of sequential encoders and decoders to better handle long-term context. It comprises six encoder and decoder layers with a skip connection between corresponding layers. The encoder layer includes a convolution layer with a kernel size of 8 and a stride size of 4, followed by ReLU activation. Then, the encoder end with the 1×1 convolution layer with Gated Linear Unit activation. The decoder layer sums the corresponding skip connection with the input from the previous decoder and applies GLU with 1×1 convolution. The decoder layer ends with a convolution with a kernel size of 8 and a stride of 4, followed by a ReLU activation [56].

With Hybrid Demucs, the author extends Demucs with spectral domain and provides multi-domain analysis. Besides the spectral domain, Hybrid Demucs have compressed residual branches, local attention, and singular value regularization improvements. Another distinction is that ReLU activations in the model change with Gaussian Error Linear Units. Each encoder layer has compressed residual branches that are composed of dialeted convolutions. Inside, convolutions are processed separately for both time dimension and frequency bins. However, in the fifth and sixth layers, there are additional two-layer Bi-LSTM and local attention. Thus, Bi-LSTM, between encoders and decoders in the original Demucs, is moved into the encoder at Hybrid Demucs. After decoding, the output spectrogram is inversed with the ISTFT and summed with the temporal output. This way, the model creates the final prediction output [56].

5. METHODOLOGY

5.1. Data Preparation

MUSDB18-HQ [57] dataset was used while experimental studies. It is a popular dataset in audio source separation studies. MUSDB18-HQ is an uncompressed version of the MUSDB18 dataset, and it consists of 150 tracks separated as training and test subsets. Signals of songs are stereophonic and encoded at 44.1 kHz.

Combining the train and test subsets of the dataset, 150 songs were used for the audio source separation experiment. Drums, bass, vocals, and other components were mixed artificially instead of the mixture in the dataset and the song itself. After that, the separation experiment was carried out. The mixing algorithm is as follows;

Algorithm 3 Source Mixing Algorithm

```
1: for 1 to $number\ of\ components$ do
2:     $min \leftarrow minimum\ of\ component$
3:     $max \leftarrow maximum\ of\ component$
4:     if $max < 1$ or $min > 1$ then
5:         $component \leftarrow \frac{component}{\frac{max}{2}} - 0.5$
6:     end if
7: end for
```

The matrix factorization-based approaches, FastICA and NMF, cannot solve underdetermined BSS problems as the mixing matrix is not invertible. Hence $R \times N$ matrix was given as an input to these algorithms, in which R is the number of components, and N is the number of samples of the song. $R = 4$ as components are vocal, bass, drums, and others.

Other methods, on the other hand, separate sources using two channels. The input data of these methods was prepared by applying the mixing algorithm for two channels. The sources were mixed with each channel's mixing algorithm, and then the mean of these four components was taken. The mean values represented one channel. Thus, the $C \times N$ matrix created with two channels became the input of the algorithms.

5.2. Method Implementation Details

Pre-trained models trained by the authors of methods were used in experiments for machine learning methods. Application specific subtleties are given as follows:

- FastICA; there are two important parameters for the FastICA algorithm. The threshold value can also be defined as the error of the separation and iteration. Values used at implementation are $1e-8$ and 5000, respectively.
- NMF; [58] was used while implementing the NMF algorithm. Similarly, threshold and iteration parameters have a key role in NMF, and they were determined as $1e-5$ and 1000, respectively. The reason for choosing lower values compared to FastICA is that the algorithm run time is quite long.
- DUET; The implementation has been used in experimental studies as in the library, which includes several separation methods mentioned in [59].
- Spleeter; Pre-trained model called “spleeter:4stems” was used in experimental studies, and there are no additional parameters.
- Wave-U-Net; Pre-trained model at [60] was used in experimental studies.
- Hybrid Demucs; Pre-trained model called “mdx” was used in experimental studies, and there are no additional parameters.
- Open Unmix; Pre-trained model called “umxl” was used in experimental studies, and there are no additional parameters.

5.3. Metric

The performance evaluation results are obtained by comparing the source signal with the estimated signal. Initially, the estimated source is divided into four components;

$$\hat{s}_j = s_{target} + e_{interf} + e_{noise} + e_{artif} \tag{96}$$

where $e_{interf}, e_{noise}, e_{artif}$ is the error term related to interferences, noise, and artifacts, respectively.

For calculating the related errors, in [61], the authors propose orthogonal projections. Let $\prod \{y_1, ..., y_n\}$ is denoted as the orthogonal projector onto the subspace spanned by $y_1, ...y_k$ vectors. Considering T as the length o vectors, the projector have TxT dimension. Three orthogonal projectors can be described as;

$$\begin{aligned} P_{s_j} :=& \prod \{s_j\} \\ P_s :=& \prod \left\{(s_{j'})_{1 \leq j' \leq n}\right\} \\ P_{s,n} :=& \prod \left\{(s_{j'})_{1 \leq j' \leq n}, (n_i)_{1 \leq i \leq m},\right\} \end{aligned} \tag{97}$$

,ehere $(s_{j'})_{j' \neq j}$ is denoted as perceived signal from unwanted sources and $(n_i)_{1 \leq i \leq m}$ from sensor noises.

Thus, the error term components can be defined as;

$$\begin{aligned} s_{target} :=& P_{s_J} \hat{s}_j \\ e_{interf} :=& P_s \hat{s}_j - P_{s_J} \hat{s}_j \\ e_{noise} :=& P_{s,n} \hat{s}_j - P_{s_J} \hat{s}_j \\ e_a rtif :=& \hat{s}_j - P_{s,n} \hat{s}_j \end{aligned} \tag{98}$$

Source-to-Distortion Ratio (SDR), a frequently used metric in the field of blind source separation, was used to evaluate the separated sources. SDR can be considered an overall score of source separation quality [49]. SDR can be expressed as;

$$SDR := 10\log_{10} \frac{\left\|s_{target}\right\|^2}{\left\|e_{interf} + e_{noise} + e_{artif}\right\|^2} \tag{99}$$

Museval [62] library was used to measure SDR metric values.

5.4. Experimental Environment

Experiments were conducted on the computer's operating system and hardware specifications: Ubuntu 20.04, Intel Core i7-1065G7, 24 GB RAM, and 2 GB NVidia MX330 GPU. All experiments were run with the CPU due to the lack of GPU memory and a primary processor on which all methods can run.

FastICA	NMF	DUET	Hybrid Demucs	Wave-U-Net	Open Unmix	Spleeter
9,21	296,20	29,21	574,17	357,20	148,29	23,21

Table 6.1 Average Time Spent Per Track in Seconds.

FastICA	NMF	DUET	Hybrid Demucs	Wave-U-Net	Open Unmix	Spleeter
23,22	673	73,02	1435,43	893	370,71	58,04

Table 6.2 Total Separation Time in Minutes

6. EXPERIMENTAL RESULTS

The results will be collected under three headings; time evaluation, genre evaluation, and score evaluation.

6.1. Time Evaluation

The time spent during the source separation process is an essential indicator of how method complex. The average time spent per piece for each method is shown in Table 6.1 and depicted in Figure 6.1. The total time taken in minutes for separation is given in Table 6.2. Hybrid Demucs architecture works with the spectral and time domains, and two Bi-LSTM layers are located in the center of the architecture. Hence the complexity has increased, and the separation time has increased considerably. Since the experiments are done with the CPU, it is expected that the machine learning-based methods will take longer, but it can be concluded that the GPU memory requirement is higher for Hybrid Demucs or Wave-U-Net.

6.2. Genre Evaluation

When thinking of rock music, drums come to mind, along with the guitar. Different genres of music give weight to different instruments. In this regard, it is significant to explain how the audio source separation methods perform. The average separation time of the methods

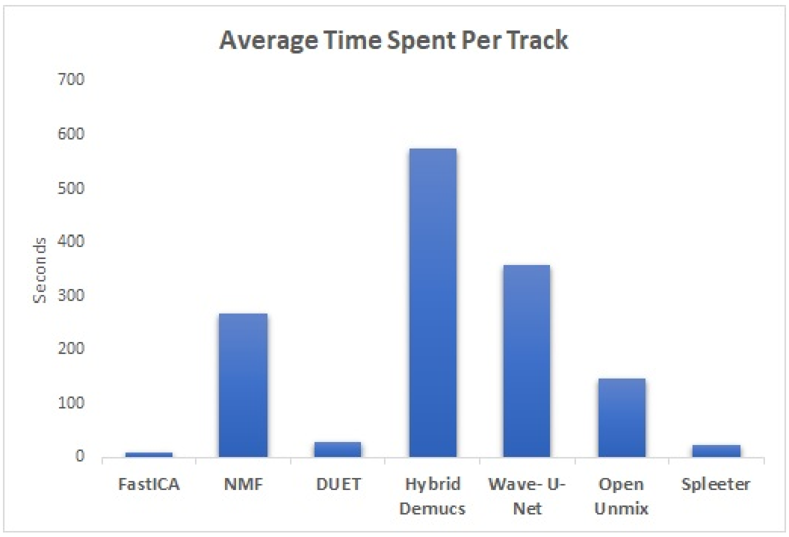


Figure 6.1 Avreage Time Spent Per Track

Singer/Songwriter	Pop/Rock	Country	Rock	Rap	Reggae	Electronic	Heavy Metal	Pop	Jazz
14	72	3	17	8	2	8	12	11	3

Table 6.3 Numbers of Tracks In The Dataset By Genre.

for each genre, the average track length of the relevant genre in the data set, and the ratio of these values are shown in Table 6.6. In addition, the number of tracks according to the genres in the dataset is shown in Table 6.3.

SDR scores of each method's performance in separating different components are tabulated. Vocal scores are in Table 6.4, bass scores are in Table 6.5, drums scores are in Table 6.6, and other scores are in Table 6.7. The mixture file created as a result of taking the mean value over the first axis of these components after stacking is also compared with the ground truth mixture file, and SDR scores are obtained and are shown in Table 6.8.

6.3. Score Evaluation

The median value of the SDR scores of the blind source separation methods after the artificial mixing and separation of 150 tracks in the MUSDB18-HQ dataset is shown in Table 6.7. Therefore, it can be seen that Spleeter, Open Unmix, and Wave-U-Net methods give better

Genre	Methods						
	FastICA	NMF	DUET	Hybrid Demucs	Open Unmix	Spleeter	Wave-U-Net
Singer/Songwriter	-25,512	-22,935	-5,678	-14,759	-16,341	0,641	-1,922
Pop/Rock	-22,445	-19,357	-4,678	-11,176	-0,031	3,886	-0,642
Country	**-13,824**	**-16,960**	**-1,071**	**-6,811**	-1,159	-1,168	**1,607**
Rock	-20,635	-20,598	-3,641	-10,107	**1,860**	**4,955**	0,234
Rap	-22,304	-20,905	-7,076	-11,642	-6,518	2,612	-1,479
Reggae	-21,645	-19,599	-6,388	-10,472	-12,639	4,800	0,271
Electronic	-30,544	-28,465	-18,262	-13,682	-4,602	-0,203	-1,547
Heavy Metal	-22,724	-20,380	-5,826	-13,030	0,627	2,077	-0,280
Pop	-21,194	-20,394	-5,539	-10,467	-2,927	2,958	-1,970
Jazz	-23,729	-23,064	-10,550	-12,651	-18,917	3,182	-1,506

Table 6.4 Vocals Component SDR Scores By Genre.

Genre	Methods						
	FastICA	NMF	DUET	Hybrid Demucs	Open Unmix	Spleeter	Wave-U-Net
Singer/Songwriter	-20,654	-19,179	-16,351	-17,084	1,313	0,251	-7,680
Pop/Rock	-22,496	-20,486	-19,221	-19,516	-0,716	0,063	-9,945
Country	-21,873	-24,113	-17,742	-18,260	**4,813**	0,293	-8,754
Rock	-20,748	-21,160	-16,009	-17,435	0,183	1,153	**-7,011**
Rap	-24,149	-22,359	-18,020	-20,260	0,673	0,742	-9,582
Reggae	**-18,213**	-17,724	-15,161	**-16,042**	2,380	0,623	-6,549
Electronic	-19,342	**-17,026**	**-13,941**	-17,365	0,308	**1,324**	-7,012
Heavy Metal	-21,995	-19,920	-17,617	-18,920	-0,543	0,018	-9,930
Pop	-21,842	-22,497	-17,373	-18,336	-1,665	-0,001	-9,282
Jazz	-57,417	-59,130	-42,362	-48,966	0,971	-5,992	-36,074

Table 6.5 Bass Component SDR Scores By Genre.

results than other methods. For each method, better results are also observed in different components.

6.4. Waveform Evaluation

For a waveform examination, the waveforms of each method and each component for that method are depicted in Figure 6.2 and Figure 6.3 for a sample track called ”Actions - Devil’s Words”. NMF and FastICA methods’ waveforms generally show a uniform distribution, while other components show a reasonable form.

	Methods						
Genre	FastICA	NMF	DUET	Hybrid Demucs	Open Unmix	Spleeter	Wave-U-Net
Singer/Songwriter	-26,323	-22,618	-10,473	-18,898	0,191	0,101	0,946
Pop/Rock	-20,957	-19,336	-5,145	-13,273	1,771	1,383	2,169
Country	-18,487	-22,147	-4,202	-12,962	**6,210**	1,155	3,220
Rock	-19,675	-20,269	-4,793	-11,521	2,616	1,165	2,068
Rap	-22,402	-20,615	-6,005	-14,098	1,939	0,628	2,248
Reggae	-20,938	-18,961	-5,383	-13,306	0,059	-0,282	0,242
Electronic	**-17,892**	**-15,269**	-4,205	**-10,259**	3,234	**2,266**	**3,259**
Heavy Metal	-20,561	-18,625	**-3,719**	-13,096	0,483	0,555	1,505
Pop	-23,190	-21,218	-7,902	-15,209	0,966	1,158	2,519
Jazz	-58,338	-59,447	-25,711	-45,379	0,203	-8,450	-17,671

Table 6.6 Drums Component SDR Scores By Genre.

	Methods						
Genre	FastICA	NMF	DUET	Hybrid Demucs	Open Unmix	Spleeter	Wave-U-Net
Singer/Songwriter	-21,368	-20,025	-3,425	-5,632	-12,462	-17,221	-1,334
Pop/Rock	-22,740	-20,221	-6,078	-7,119	-15,948	-18,552	-2,246
Country	-23,794	-26,680	**-0,738**	-8,511	-11,130	-20,174	-3,639
Rock	-22,388	-22,543	-3,392	-6,664	-18,281	-17,973	-2,205
Rap	-26,756	-24,496	-15,725	-11,672	-21,459	-22,929	-5,003
Reggae	-22,038	-20,295	-6,516	-8,930	-11,365	-19,437	-4,861
Electronic	-22,464	-18,708	-4,970	-6,539	-17,083	-17,383	-1,692
Heavy Metal	**-19,997**	**-18,704**	-3,449	**-5,051**	-15,950	**-16,113**	**-1,260**
Pop	-22,344	-20,705	-4,942	-5,616	-15,524	-16,865	-1,966
Jazz	-25,633	-25,371	-6,413	-9,122	**-6,270**	-20,925	-3,626

Table 6.7 Other Component SDR Scores By Genre.

	Methods						
Genre	FastICA	NMF	DUET	Hybrid Demucs	Open Unmix	Spleeter	Wave-U-Net
Singer/Songwriter	-15,131	-13,712	-3,008	-8,389	-2,529	-2,525	0,972
Pop/Rock	-14,430	-13,329	-2,232	-7,544	-1,885	-1,877	0,813
Country	-13,824	-16,960	**-1,071**	-6,811	-1,159	-1,168	**1,607**
Rock	**-13,133**	-15,032	-1,142	**-6,610**	-1,244	-1,236	0,982
Rap	-14,774	-14,377	-3,831	-7,967	-2,177	-2,223	0,903
Reggae	-13,203	-11,779	-1,265	-6,746	-1,311	-1,348	1,042
Electronic	-13,601	**-11,407**	-1,424	-6,708	**-0,825**	**-0,820**	1,375
Heavy Metal	-13,823	-13,364	-1,591	-6,985	-1,350	-1,346	0,498
Pop	-14,720	-14,803	-2,478	-8,043	-2,193	-2,190	1,092
Jazz	-17,955	-15,141	-4,474	-10,373	-4,117	-4,148	0,243

Table 6.8 Mixture Component SDR Scores By Genre.

	DUET		NMF		FastICA		Spleeter		Wave-U-Net		Hyrid Demucs		Open Unmix		Track Length Mean
Genre	STM[a]	Ratio[b]	STM[a]	Ratio[b]	STM[a]	Ratio[b]	STM[a]	Ratio[b]	STM[a]	Ratio[b]	STM[a]	Ratio[b]	STM[a]	Ratio[b]	
Singer/Songwriter	30.02	0.12	288.08	1.19	11.35	0.05	24.18	0.10	351.56	1.45	584.26	2.41	152.18	0.63	242.61
Pop/Rock	31.27	0.12	282.56	1.12	8.45	0.03	23.42	0.09	391.67	1.55	613.44	2.42	159.46	0.63	253.29
Country	10.57	0.12	85.59	**0.98**	1.86	0.02	9.95	0.11	123.85	1.41	195.99	**2.23**	54.60	0.62	87.76
Rock	19.97	0.12	201.70	1.25	5.33	0.03	18.38	0.11	243.93	1.51	386.41	2.39	102.14	0.63	161.74
Rap	35.79	0.13	321.37	1.15	9.03	0.03	26.39	0.09	395.05	1.41	730.56	2.61	175.67	0.63	279.73
Reggae	33.00	0.12	275.39	1.02	8.09	0.03	24.82	0.09	440.37	1.63	631.55	2.34	171.49	0.64	269.84
Electronic	32.96	0.13	312.87	1.19	6.19	0.02	29.62	0.11	388.65	1.48	640.67	2.43	165.62	0.63	263.28
Heavy Metal	27.66	0.12	255.33	1.14	9.92	0.04	21.91	0.10	323.77	1.45	559.50	2.50	140.53	0.63	223.58
Pop	28.24	0.13	268.63	1.19	23.30	0.10	27.21	0.12	338.38	1.50	556.17	2.46	141.53	0.63	225.74
Jazz	26.57	0.12	224.55	1.01	5.53	0.03	18.52	**0.08**	393.91	1.78	518.86	2.35	138.09	0.62	221.25

[a] STM: Separation Time in Minutes.[b] Ratio: STM / Track Length Mean.

Table 6.6 Genre-Based SDR Scores

Considering the data in the table, the ratio values for reggae and country genres seem low, and the ratio for rap genre seems high.

	Methods						
Components	FastICA	NMF	DUET	Hybrid Demucs	Open Unmix	Spleeter	Wave-U-Net
Vocals	**-13,630**	-13,776	**-2,050**	-7,904	-1,962	-1,954	0,798
Bass	-20,936	-19,355	-2,112	-10,992	0,462	1,874	-0,645
Drums	-20,264	-18,142	-3,178	-13,103	**4,176**	**2,702**	**3,077**
Other	-26,762	-25,503	-24,105	-23,897	-4,669	-1,145	-14,671
Mixture	-14,345	**-13,692**	-2,140	**-7,439**	-1,754	-1,768	0,941

Table 6.7 SDR Scores of Methods By Components.

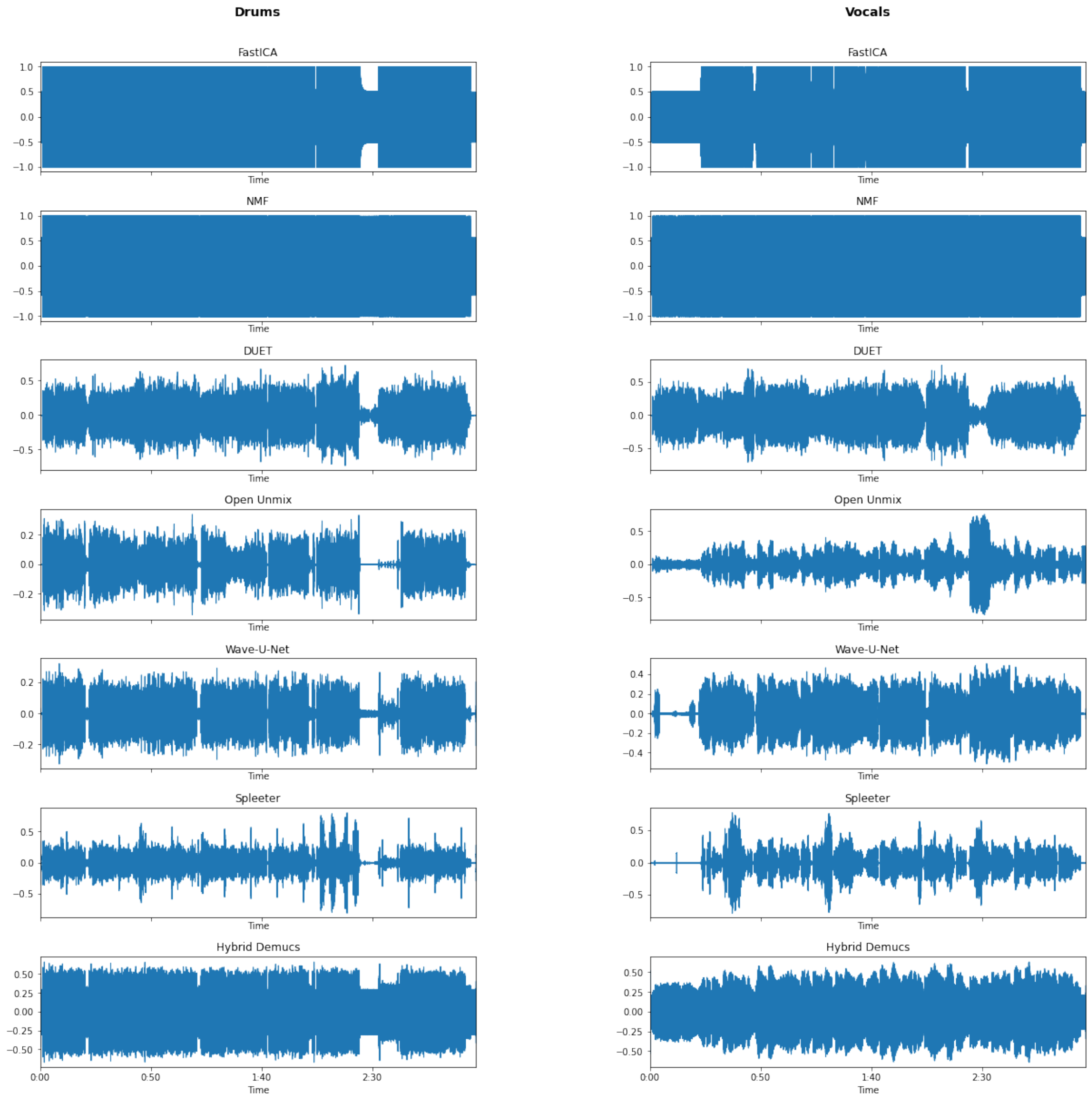


Figure 6.2 Drum and Vocal Component Waveform of Methods

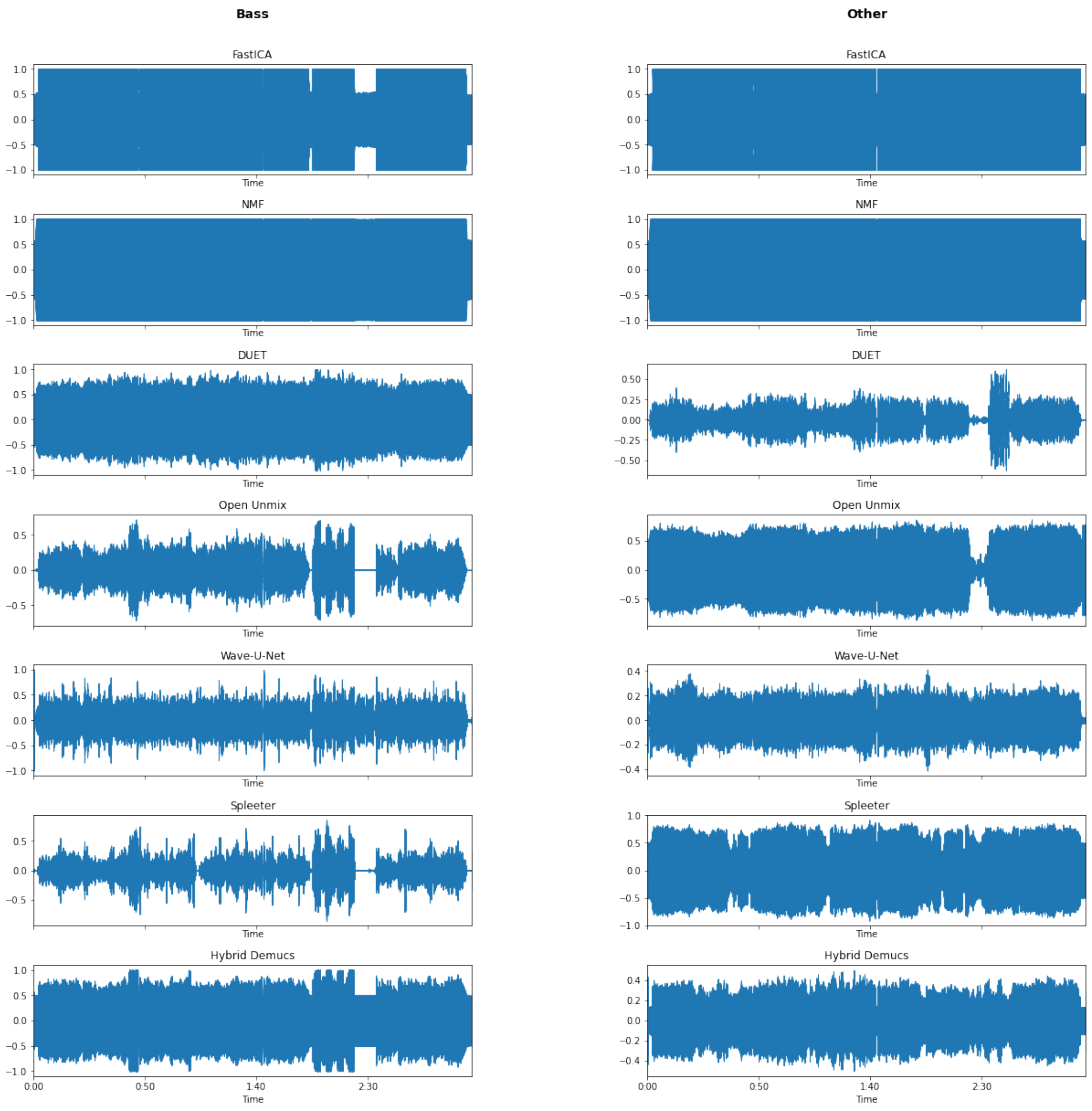


Figure 6.3 Bass and Other Component Waveform of Methods

7. CONCLUSION

In this study, various algorithms used in audio source separation studies were implemented, and results were obtained via a series of experiments. Several important features noted during the experiments are as follows;

- RAM requirements were high for NMF, Hybrid Demucs, Wave-U-Net, which have high runtimes, and DUET, which have high computational complexity.
- Although Hybrid Demucs won first place in Sony Music Demixing Challenge 2021 with SDR scores on the MusDB-HQ dataset, and it could not show a similar performance in artificially mixed components.
- Classical methods have been implemented in many studies. However, the star ratings of the official GitHub repositories of machine learning-based models are an essential indicator of their popularity in the community. Accordingly, as of October 2022, Spleeter is far ahead of other methods with 20.7k stars. Then comes Hybrid Demucs with 4.4k stars, Open Unmix with 869 stars, and Wave-U-Net with 632 stars. Our findings are in good compliance with these numbers.
- Considering the results, the Spleeter method displayed the best performance according to the working time and score values.
- The fact that matrix factorization-based NMF and FastICA algorithms cannot solve under-determined problems and this constitutes a drawback.

This study will help the researchers studying on BSS in terms of selection of the most appropriate algorithm when a particular metric is essential.

REFERENCES

[1] Scott C. Douglas and Malay Gupta. *Convolutive Blind Source Separation for Audio Signals*, page 3–42. Springer, **2007**.

[2] M. Müller. *Fundamentals of Music Processing: Using Python and Jupyter Notebooks*. Springer International Publishing, **2021**. ISBN 9783030698072.

[3] M. Kumar and V. E. Jayanthi. Blind source separation using kurtosis, negentropy and maximum likelihood functions. *International Journal of Speech Technology*, 23(1):13–21, **2020**. ISSN 1572-8110. doi:10.1007/s10772-019-09664-z.

[4] Dan Hu Xianchuan Yu and Jindong Xu. *Blind Source Separation - Theory and Applications*. Wiley, **2014**.

[5] Alaa Tharwat. Independent component analysis: An introduction. *Applied Computing and Informatics*, 17(2):222–249, **2021**.

[6] Jonathon Shlens. A Tutorial on Independent Component Analysis. *arXiv e-prints*, arXiv:1404.2986, **2014**.

[7] A. Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, **1999**. doi:10.1109/72.761722.

[8] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430, **2000**. ISSN 0893-6080. doi:10.1016/S0893-6080(00)00026-5.

[9] Zhongni Wang, Xianchuan Yu, and Libao Zhang. A remote sensing image fusion algorithm based on ordinal fast independent component analysis. ACM, **2010**. doi:10.4108/wkdd.2008.2690.

[10] James V. Stone. *Independent Component Analysis: A Tutorial Introduction*. The MIT Press, **2004**. ISBN 9780262257046. doi:10.7551/mitpress/3717.001.0001.

[11] Shun-ichi Amari, Scott C. Douglas, Andrzej Cichocki, and H.-H. Yang. Multichannel blind deconvolution and equalization using the natural gradient. *First IEEE Signal Processing Workshop on Signal Processing Advances in Wireless Communications*, pages 101–104, **1997**.

[12] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. Independent component analysis. *John Wiley & Sons, Inc*, **2001**.

[13] Ying Tan and Jun Wang. Nonlinear blind source separation using higher order statistics and a genetic algorithm. *IEEE Trans. Evol. Comput.*, 5:600–612, **2001**.

[14] Wai Lok Woo and L.C. Khor. Blind restoration of nonlinearly mixed signals using multilayer polynomial neural network. *Vision, Image and Signal Processing, IEE Proceedings -*, 151:51–61, **2004**. doi:10.1049/ip-vis:20040302.

[15] Rabi Behera and Kajaree Das. A survey on machine learning: Concept, algorithms and applications. *International Journal of Innovative Research in Computer and Communication Engineering*, 2, **2017**.

[16] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994–12000, **2009**. ISSN 0957-4174. doi:10.1016/j.eswa.2009.05.029.

[17] Jayanth Koushik. Understanding Convolutional Neural Networks, **2016**.

[18] Thomas Rincy N and Roopam Gupta. A survey on machine learning approaches and its techniques:, **2020**. doi:10.1109/SCEECS48394.2020.190.

[19] Anders Krogh. What are artificial neural networks? *Nature biotechnology*, 26:195–7, **2008**. doi:10.1038/nbt1386.

[20] Murat H. Sazlı. A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, 50(01):0–0, **2006**. doi:10.1501/commua1-2_0000000026.

[21] Keiron O'Shea and Ryan Nash. An Introduction to Convolutional Neural Networks, **2015**.

[22] Jake V. Bouvrie. Notes on convolutional neural networks. **2006**.

[23] Robin M. Schmidt. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview, **2019**.

[24] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures, **2019**. doi:10.1162/neco_a_01199.

[25] Chris Nicholson. A beginner's guide to lstms and recurrent neural networks, **2019**.

[26] Gang Chen. A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation, **2016**.

[27] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks, **1997**. doi:10.1109/78.650093.

[28] Amar Kachenoura, Doha Safieddine, Laurent Albera, Gwénaël Birot, Fabrice Wendling, Lotfi Senhadji, and Isabelle Merlet. Blind Source Separation Methods Applied to Muscle Artefacts Removing from Epileptic Eeg Recording: A Comparative Study. In *RITS 2011 (Colloque National Recherche en Imagerie et Technologies pour la Santé)*, pages 1708.1–1708.3. **2011**.

[29] Nicolle Correa, Tülay Adali, and Vince D Calhoun. Performance of blind source separation algorithms for fMRI analysis using a group ICA method. *Magn Reson Imaging*, 25(5):684–694, **2006**.

[30] A J Bell and T J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Comput*, 7(6):1129–1159, **1995**.

[31] J.F. Cardoso and Antoine Souloumiac. Blind beamforming for non gaussian signals. *Radar and Signal Processing, IEE Proceedings F*, 140:362–370, **1994**. doi:10.1049/ip-f-2.1993.0054.

[32] P. Georgiev and A. Cichocki. Blind source separation via symmetric eigenvalue decomposition. In *Proceedings of the Sixth International Symposium on Signal Processing and its Applications (Cat.No.01EX467)*, volume 1, pages 17–20 vol.1. **2001**. doi:10.1109/ISSPA.2001.949764.

[33] Norsalina Hassan and Dzati Athiar Ramli. A comparative study of blind source separation for bioacoustics sounds based on fastica, pca and nmf. *Procedia Computer Science*, 126:363–372, **2018**. ISSN 1877-0509. doi:10.1016/j.procs. 2018.07.270.

[34] Dominique Fourer and Geoffroy Peeters. Single-Channel Blind Source Separation for Singing Voice Detection: A Comparative Study. *arXiv e-prints*, arXiv:1805.01201, **2018**.

[35] Antoine Liutkus, Derry Fitzgerald, Zafar Rafii, Bryan Pardo, and Laurent Daudet. Kernel additive models for source separation. *IEEE Transactions on Signal Processing*, 62(16):4298–4310, **2014**. doi:10.1109/TSP.2014.2332434.

[36] Pierre Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3):287–314, **1994**. ISSN 0165-1684. doi:10.1016/0165-1684(94) 90029-9. Higher Order Statistics.

[37] Alaa Tharwat. Principal component analysis - a tutorial. *International Journal of Applied Pattern Recognition*, 3(3):197–240, **2016**. doi:10.1504/IJAPR.2016. 079733. PMID: 79733.

[38] David Overbye and Roland Priemer. Blind multiuser detection for ds-cdma using independent component analysis neural network. *International Journal of Smart Engineering System Design*, 5(4):555–564, **2003**. doi:10.1080/ 10255810390445517.

[39] Zhang Min, Zhu Mu, and Ma Wenjie. Implementation of fastica on dsp for blind source separation. *Procedia Engineering*, 29:4228–4233, **2012**. ISSN 1877-7058. doi:10.1016/j.proeng.2012.01.648.

[40] Te-Won Lee (auth.). *Independent Component Analysis: Theory and Applications*. Springer, 1 edition, **1998**. ISBN 9781441950567; 1441950567; 9781475728514; 1475728514.

[41] Anke Meyer-Baese and Volker Schmid. Chapter 8 - transformation and signal-separation neural networks. In Anke Meyer-Baese and Volker Schmid, editors, *Pattern Recognition and Signal Analysis in Medical Imaging (Second Edition)*, pages 245–289. Academic Press, Oxford, second edition edition, **2014**. ISBN 978-0-12-409545-8. doi:10.1016/B978-0-12-409545-8.00008-X.

[42] Athanasios Papoulis and S Unnikrishna Pillai. *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, **1986**.

[43] Dinh-Tuan Pham and Philippe Garat. Blind separation of mixture of independent sources through a quasi-maximum likelihood approach. *IEEE Transactions on Signal Processing*, 45(7):1712 – 1725, **1997**. doi:10.1109/78.599941.

[44] Daniel Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13. MIT Press, **2000**.

[45] Rickard Scott. The DUET blind source separation algorithm. In S Makino and K T W Sawada, editors, *Blind Source Separation, Signals and Communication Technology*, pages 217–241. Springer, Dordrecht, **2007**.

[46] O. Yilmaz and S. Rickard. Blind separation of speech mixtures via time-frequency masking, **2004**. doi:10.1109/TSP.2004.828896.

[47] Scott Rickard, Radu Balan, and Justinian Rosca. Real-time time-frequency based blind source separation. **2002**.

[48] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Open-unmix - a reference implementation for music source separation. *Journal of Open Source Software*, 4(41):1667, **2019**. doi:10.21105/joss.01667.

[49] Ethan Manilow, Prem Seetharman, and Justin Salamon. *Open Source Tools & Data for Music Source Separation*. https://source-separation.github.io/tutorial, **2020**.

[50] Alexander A S Gunawan, Albert Stevelino, Heri Ngarianto, Widodo Budiharto, and Rini Wongso. Implementation of blind speech separation for intelligent humanoid robot using duet method. *Procedia Computer Science*, 116:87–98, **2017**. ISSN 1877-0509. doi:10.1016/j.procs.2017.10.014.

[51] Andreas Jansson, Eric J. Humphrey, Nicola Montecchio, Rachel M. Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep u-net convolutional networks. In *ISMIR*. **2017**.

[52] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241. Springer International Publishing, Cham, **2015**. ISBN 978-3-319-24574-4.

[53] Laure Prétet, Romain Hennequin, Jimena Royo-Letelier, and Andrea Vaglio. Singing voice separation: a study on training data. *arXiv e-prints*, arXiv:1906.02618, **2019**.

[54] Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation. *arXiv e-prints*, arXiv:1806.03185, **2018**.

[55] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. Music Source Separation in the Waveform Domain. *arXiv e-prints*, arXiv:1911.13254, **2019**.

[56] Alexandre Défossez. Hybrid Spectrogram and Waveform Source Separation. *arXiv e-prints*, arXiv:2111.03600, **2021**.

[57] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. MUSDB18-HQ - an uncompressed version of musdb18, **2019**. doi:10.5281/zenodo.3338373.

[58] Patricio López-Serrano, Christian Dittmar, Yigitcan Özer, and Meinard Müller. Nmf toolbox: Music processing applications of nonnegative matrix factorization. **2019**.

[59] Ethan Manilow, Prem Seetharaman, and Bryan Pardo. The northwestern university source separation library. In *ISMIR*, pages 297–305. **2018**.

[60] Daniel Stoller. Implementation of the Wave-U-Net for audio source separation, **2022**.

[61] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14:1462 – 1469, **2006**. doi:10.1109/TSA.2005.858005.

[62] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. The 2018 signal separation evaluation campaign. In *Latent Variable Analysis and Signal Separation: 14th International Conference, LVA/ICA 2018, Surrey, UK*, pages 293–305. **2018**.