Speech Enhancement Using a Reduced Complexity MFCC-based Deep Neural Network

Ryan Razani

Department of Electrical & Computer Engineering
McGill University
Montreal, Canada

November 2017

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master.

2017/11/30

Abstract

In contrast to classical noise reduction methods introduced over the past decades, this work focuses on a regression-based single-channel speech enhancement framework using DNN, as recently introduced by Liu *et al.*. While the latter framework can lead to improved speech quality compared to classical approaches, it is afflicted by high computational complexity in the training stage. The main contribution of this work is to reduce the DNN complexity by introducing a spectral feature mapping from noisy mel frequency cepstral coefficients (MFCC) to enhanced short time Fourier transform (STFT) spectrum. Leveraging MFCC not only has the advantage of mimicking the logarithmic perception of human auditory system, but this approach requires much fewer input features and consequently lead to reduced DNN complexity. Exploiting the frequency domain speech features obtained from the results of such a mapping also avoids the information loss in reconstructing the time-domain speech signal from its MFCC. While the proposed method aims to predict clean speech spectra from corrupted speech inputs, its performance is further improved by incorporating information about the noise environment into the training phase. We implemented the proposed DNN method with different numbers of MFCC and used it to enhance several different types of noisy speech files. Experimental results of perceptual evaluation of speech quality (PESQ) show that the proposed approach can outperform the benchmark algorithms including a recently proposed non-negative matrix factorization (NMF) approach, and this for various speakers and noise types, and different SNR levels. More importantly, the proposed approach with MFCC leads to a significant reduction in complexity, where the runtime is reduced by a factor of approximately five.

Sommaire

Contrairement aux méthodes classiques de réduction du bruit introduites au cours des dernières décennies, ce travail se concentre sur un cadre d'application de réhaussement de la parole monocanal basé sur la régression au moyen d'un réseau de neurones profond (DNN), proposé par Liu *et al.*. Alors que ce nouveau cadre d'application peut conduire à une meilleure qualité de la parole par rapport aux approches classiques, il est caractérisé par une complexité de calcul élevée dans la tâche d'apprentissage. La principale contribution de ce travail est de réduire la complexité du DNN par modélisation de la transformation entre les coefficients cepstraux à fréquence mel (MFCC) du signal bruité et la transformée de Fourier à court terme du signal parole rehaussé. Le fait de tirer parti des MFCC a non seulement l'avantage d'imiter la perception logarithmique du système auditif humain, mais cette approche nécessite beaucoup moins de variables d'entrée, ce qui en retour réduit la complexité du DNN. Exploiter les caractéristiques spectrales de la parole obtenues au moyen de cette transformation évite également la perte d'information inhérente lors de la reconstruction du signal parole dans le domaine temporel à partir des MFCC. Alors que la méthode proposée vise à prédire des spectres de parole propres à partir d'entrées corrompues, ses performances sont encore améliorées en intégrant des informations sur l'environnement de bruit dans la phase d'entraînement. Nous avons mis en oeuvre la méthode DNN proposée avec différents nombres de MFCC et l'avons appliquée au rehaussement de signaux de parole contaminés par différents types de bruit. Les résultats expérimentaux de l'évaluation perceptuelle de la qualité de la parole (PESQ) montrent que l'approche proposée surpasse les algorithmes de référence incluant un algorithme récent de factorisation matricielle non-negative (NMF), et ceci pour différents locuteurs, types de bruit, et niveaux de rapport signal-sur-bruit De manière plus importante, la nouvelle approche conduit à une réduction significative de la complexité de calcul et du temps d'éxécution, par un facteur d'environ cinq.

Acknowledgments

First and foremost, I wish to express my deepest appreciation to my supervisor, Prof. Benoit Champagne for the patient guidance, encouragement and advice he has provided throughout my time as his student. I would also like to thank Mr. Hanwook Chung (Ph.D. student) and Dr. Yazid Attabi (post-doctoral fellow) for their help and constructive comments over the course of my thesis work.

I am also grateful for the financial support provided by Prof. Champagne via his research grants from the Natural Sciences and Engineering Research Council (NSERC) of Canada, and Microsemi Canada Ltd, without which the realization of this thesis would not have been possible.

I must express my gratitude to my family for their continued support and encouragement throughout the years. I will be forever indebted to them, without whose unconditional support, love, encouragement, I would have never made it this far.

Special appreciation goes out to my friends and fellows in the Telecommunications and Signal Processing (TSP) laboratory for their moral support and inspiring discussions.

Contents

List of Figures

List of Tables

List of Acronyms

AFB	Analysis Filter Bank
AI	Artificial Intelligence
ANN	Artificial Neural Network
BM	Boltzmann Machine
BNN	Biological Neural Network
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
FFNN	Feed-forward Neural Network
FFT	Fast Fourier Transform
HMM	Hidden Markov Model
KLD	Kullback-Leibler Divergence
LMS	Least Mean Squares
MLP	Multilayer Perceptron
MMSE	Minimum Mean-square Error
MU	Multiplicative Update
NMF	Non-negative Matrix Factorization
NN	Neural Network
PDF	Probability Density Function
PESQ	Perceptual Evaluation of Speech Quality
PSD	Power Spectral Density
RBM	Restricted Boltzmann Machine

RL	Reinforcement Learning
RNN	Recurrent Neural Network
SDR	Source-to-Distortion Ratio
SegSNR	Segmental SNR
SFB	Synthesis Filter Bank
SNR	Signal-to-Noise Ratio
STFT	Short-time Fourier Transform
STSA	Short-time Spectral Amplitude
VAD	Voice Activity Detection
WF	Wiener Filter
WWF	Weighted Wiener Filter

Notations

Bold lower case letter	Vector
Bold upper case letter	Matrice
Lower case letter	Scalar
Upper case letter	Sscalar (in frequency domain)
Superscript ^{T}	Transposition
Subscript m or k	Frequency index
Subscript n or l	Time instance
$\Re$	Real Part of a Complex Value
$\Im$	Imaginary Part of a Complex Value
$E(.)$	Expectation Operator
z^*	Complex Conjugate of z
$\mathbb{R}$	Real Numbers
$\mathbb{R}_+$	Non-negative Real Numbers
$\ln(.)$	Natural Logarithm
$\log(.)$	Logarithm
$\otimes$	Element-wise Multiplication

Chapter 1

Introduction

This chapter provides a general introduction to the thesis. An incisive overview of the single channel speech enhancement problem to be studied is given. Then existing literature aimed at solving the denoising problem is surveyed. Next, we summarize the technical contributions made by this thesis. Finally, the thesis organization is explained and key notations are defined for reference.

1.1 Overview of Speech Enhancement

Humans possess different ways to communicate to one another and retrieve information from the outside world. However, speech as opposed to images and written text, stands out as the most efficient and convenient source of information to communicate. Speech conveys linguistic contents as well as other useful information about the speaker. Speech communication is the dominant mode of human social bonding and information exchange. In real-world application of speech processing, when a desired speech signal propagates through an acoustic channel and is received at a microphone, it is distorted by unwanted noise and other sources of interference that results in degradation of its quality and intelligibility. Thus, in order for a speech processing system to operate satisfactorily, sophisticated noise reduction techniques are needed to extract the desired speech signal content from its corrupted received version.

The purpose of speech enhancement is to improve the perceived quality or intelligibility of speech signals that have been degraded due to different types of acoustic background noise and interference. Nowadays, the increasing demand for high quality speech signals

motivates the search of improved signal processing techniques that can remove such unwanted noise from the observed speech signal. Speech enhancement finds various applications such as hearing aids, cellular phones, multiparty conferencing, robust speech or speaker recognition, voice over internet protocol (VoIP), security monitoring and intelligence, etc. On the basis of the number of microphones being employed, speech enhancement techniques can be broadly classified as single versus multiple channels [3]. In contrast to the single channel approaches, multichannel techniques [4] take advantage of the availability of multiple input signals. Theoretically, these techniques can exploit additional information about the acoustic environment and possibly lead to improved performance. However, the advantages of such systems come at cost of an increased computational complexity in processing, especially when the microphone geometry is unknown. In this work, however, the focus is on single channel techniques due to convenience in implementation as well as cost considerations.

Speech enhancement has been an arduous task to tackle for many years due to the complex nature of the noise signals whose characteristics can change abruptly in time. Accordingly, developing speech enhancement systems that perform well in different environment is challenging. In addition, performance characterization of such systems depend to some extent on the specific application being considered. Two commonly used perceptual criteria to measure the performance are intelligibility and quality. While the former is objective as it indicates how comprehensive is the speech, the latter remains subjective for human listeners as different persons will judge in different ways the level of noise reduction versus the amount of distortion introduced in the processed speech signal. It has been shown that these criteria are rarely satisfied simultaneously. For this reasons, several objective (i.e., computational) measures of speech quality have been considered in the literature [3]. In effect, the majority of speech enhancement algorithms obtain improvement in noise reduction at the expense of some speech distortion. It is therefore essential to ensure that resulting speech distortion lies within a certain threshold in an attempt to perform noise reduction.

1.2 Literature Review

Several single channel speech enhancement techniques have been proposed during the past decades, including spectral subtraction [5, 6], Wiener filtering [7, 8], minimum mean square

error short-time spectral amplitude estimation (MMSE-STSA) [9, 10], MMSE log spectral amplitude (MMSE-LSA) [11], Weighted- Euclidean STSA (WE STSA) estimator [12], Kalman filtering [13, 14], subspace methods [15–17], etc. These techniques rely on a simplified signal model where the background noise is assumed to be additive with statistical characteristics that change slowly over time [18]. While such modeling leads to tractable signal processing operations, the enhancement performance of these traditional methods suffers from limited noise reduction, musical noise, and non-linear distortion.

Recently, there has been much interest towards the application of machine learning techniques to the speech enhancement problem, including non-negative matrix factorization (NMF) [19,20] and deep neural networks (DNN) [21,22]. In particular, neural network models with non-linear activation functions are believed to be suitable for representing the complex mapping relationship between the noisy and clean speech signals. Early work on the use of shallow neural networks (SNN) as non-linear filters in speech enhancement has been presented in [23, 24]. Yet, the performance of the SNN model with limited network size and small training set is not satisfactory. With the advancement of machine learning algorithms and improvement in digital hardware performance, the DNN structure has been drawing considerable attention lately within the research community, as it can achieve significantly better performance compared to SNN, at the cost of increased computational complexity. DNN with multiple hidden layers are now preferred for many applications as they can more efficiently learn and represent statistical information [21].

An alternative approach to tackle speech enhancement is provided by NMF, a popular dictionary learning technique which has found successful applications in numerous fields such as source separation [25], speech enhancement [19], and speech recognition [26]. In this approach, a given non negative matrix of signal descriptors is decomposed into the product of a nonnegative basis matrix (also known as dictionary) and activation matrix. NMF is a dimensionality reduction tool which, as apposed to principal components analysis (PCA) and vector quantization (VQ), only allows additive (and not subtractive) combinations of the basis vectors. In the context of speech enhancement, either the short-term power or magnitude spectrum of the speech signal is expressed through NMF as a linear combination of the basis vectors. Particularly, in the case of the supervised NMF method, the basis vectors are derived for each source during the training phase and later used in the enhancement phase. However due to the mismatch between the training and test signals, the quality of the processed speech is limited. To compensate for this type of limitations,

one can incorporate regularization terms into the NMF cost function [27, 28].

In recent works on speech enhancement, DNN-based models have been presented that employ multi-condition training procedures to initialize the network parameters, such as the restricted Boltzmann machine (RBM) [29] and deep denoising autoencoder (DAE) [30]. However, the use of these pre-training approaches is computationally expensive and does not seem to notably affect the final enhancement performance of the DNN with ReLU activation function, given sufficiently large and varied training data sets [31]. In [22], Liu *et al.* presented a simpler speech enhancement approach using DNN with no pre-training, which can achieve better performance when compared to NMF techniques with comparable complexity. In [32], a DNN-based speech separation technique was proposed using time-frequency masking, as obtained from a second DNN. The SVD (singular value decomposition) reduction techniques with DNN training for noisy reverberant speech recognition was investigated in [33]. The NMF-based target speech enhancement using DNN was proposed in [34]. In [35], a signal pre-processing front-end based on DNN was presented to enhance the speech signal for robust speech recognition; however, the learning-based noise model was not considered. Besides, the deep recurrent neural network (DRNN) [36] was introduced to exploit temporal information in the source separation problem. Although DRNN is capable of modeling sequential data for speech processing tasks, its performance is weak when trained on limited noise types [37]. Subsequently, the long short-term memory (LSTM) [38, 39] model was used to tackle the gradient vanishing and exploding problem with DRNN and to learn long-term dependencies. While the use of LSTM with DRNN leads to improved performance, it requires increased complexity in implementation.

1.3 Thesis Objective and Contributions

The main goal of this work is to overcome some of the limitations pointed above by introducing a low-complexity DNN for the purpose of regression-based single channel speech enhancement, based on the framework introduced by Liu *et al.* [22]. While the latter framework can lead to improved speech quality compared to classical approaches, it is afflicted by high computational complexity in the training stage. The main contribution of this work is to reduce the DNN complexity by introducing a spectral feature mapping from noisy mel frequency cepstral coefficients (MFCC) to enhanced short time Fourier trans-

form (STFT) spectrum. The use of MFCC not only has the advantage of mimicking the logarithmic perception of human auditory system [40], but this approach requires much fewer input features and consequently lead to reduced DNN complexity. Consequently, the DNN is characterized by a simpler training and denoising procedure and indeed does not necessitate any pre-training or complex recurrent scheme. In addition, we leverage one of the best performing first-order learning algorithm for training our DNN model, namely iRprop$^-$ introduced in [41]. The latter allows faster convergence than standard Rprop without increasing algorithmic complexity. The enhanced speech is obtained by applying Wiener filtering to the DNN output followed by time domain reconstruction. The noise reduction performance is further improved by incorporating information about the noise environment into the training phase.

We implemented the proposed DNN method with different numbers of MFCC and used it to enhance several different types of noisy speech files. Experimental results of perceptual evaluation of speech quality (PESQ) show that the proposed approach can outperform the benchmark algorithms including a recently proposed non-negative matrix factorization (NMF) approach, and this for various speakers and noise types, and different SNR levels. More importantly, the proposed DNN structure using MFCC as inputs, leads to a significant reduction in complexity, where the runtime is reduced by a factor of approximately five in our experiments.

1.4 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 outlines the motivation behind deep learning and brief overview on how artificial neural network have inspired researchers to derive successful algorithms to solve many problems. The chapter then further discusses the basic principles of neural networks and provides a mathematical description of the DNN structure and associated training approaches. On the basis of the system model equations presented in Chapter 2, the application of DNN to speech enhancement using STFT coefficients is introduced in Chapter 3, as well as several derivatives of the RPROP training algorithm. In Chapter 4, we turn over our attention to the proposed method which includes the incorporation of MFCC-based spectral feature mapping into the DNN architecture and the associated training procedure. The experimental evaluation and performance results for the proposed method for various noise types are presented in Chapter 5. Finally, we

summarize the findings of our work in Chapter 6, where we also briefly discuss potential research directions in the future.

Chapter 2

Background on DNN

In this chapter, we start with a brief introduction providing motivation behind deep learning. Next, an overview of artificial neural network (ANN), a class of machine learning techniques which is being used for solving regression and classification problems, is presented. ANN models are motivated by research on the human brain and have inspired researchers for decades to develop learning models to solve complex problems in many fields of research. Subsequently, the structure of the most fundamental neural network, namely the feed forward network along with its mathematical representation are exposed.

2.1 Deep Learning

Recently, machine learning has been the driver of a big wave of technological innovations. According to T. Mitchel, "The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience and seeking to find out the fundamental laws that govern all learning processes" [42]. Machine learning algorithms are powerful artificial intelligence tools that are being used in many aspect of modern society, including but not limited to: healthcare, trading, fraud detection, loan or insurance underwriting, and human-machine interfaces.

Conventional machine learning algorithms have limited ability to process natural data and abstract suitable features. Furthermore, a careful engineering and domain expertise is required for constructing a machine learning system to discover the suitable representation patterns from raw data. However, with the advancement of machine learning algorithms and improvement in digital hardware performance, and especially parallel computing which

allows the processing of massive amount of data, so called *deep learning* techniques have achieved much success and drawn researcher's attention recently. Deep learning strives to learn a mathematical representation of data with multiple levels of abstraction using complex models which are composed of an intricate structure. More specifically, this is achieved through the use of a high-dimensional neural network architecture whose parameters are adjusted with the help of sophisticated training algorithms. The meaning of deep learning varies among researchers. In the literature, the qualifier"deep" in deep learning refers to a neural network with more than two layers. Some researchers however have their own interpretation of the word "deep" as being associated to models which make use of unlabeled data. Deep learning models, such as the deep recurrent neural networks (RNN) and convolutional neural networks (CNN), have brought amelioration in the processing of speech, audio, image, and video data as applied to different fields of research, such as recognition, enhancement and detection problems. It is believed that deep learning will continue to accumulate successes in the near future, due to further increase in processing power and its advantage of built-in automated engineering ability. In brief, deep learning offers the key advantage of efficient learning through processing of large amount of data.

In general, machine learning approaches are classified into three categories, namely: unsupervised, supervised, and reinforcement learning. In the unsupervised learning task, the given training data does not include any corresponding target values. This corresponds to a process whereby a system tends to infer or discover a function automatically in order to represent and characterize patterns or regularities from its input data. This task is distinguishable from the other learning types due to the fact that the given data is unlabeled and therefore there is no evaluation of the accuracy of the model. Several approaches have been developed for unsupervised learning such as clustering, anomaly detection, and neural networks. In particular, among the unsupervised neural network algorithms, the adaptive resonance theory (ART) [43] and self-organizing map (SOM) are commonly used. The SOM learner is a topographic organization which produces a low dimensional representation, a so-called map, of input features which share similar properties. The ART network has its motivation in how the brain processes data. The ART model takes its input data and maps them into one of several permissible classes depending upon which pattern yields the highest resonance. Besides, ART is also concerned about how such model can learn new information while retaining the one that was previously learned.

Supervised learning is the most common form of machine learning and it is the focus

of this thesis. The training data set in this class of problem comprises instances of input vectors with their corresponding desired values. A supervised learning algorithm analyses the labeled training data and designs a function that can process new input samples, e.g. producing a desired output among a continuum of possible values or assigning a class label from a finite number of possible categories. A well defined training algorithm can be designed to optimize the correct prediction of the desired output or class label from unseen examples.

Reinforcement learning (RL) on the other hand, tends to be motivated by behaviorist psychology. This technique is concerned with how the software agent can find suitable actions to take in a given environment, in order to maximize its performance by some notion of rewards. In effect, RL is a data-driven approach towards learning behavior; however, in contrast to supervised learning algorithm, RL is not given desired outputs. Therefore, RL model is forced to identify the optimal output by a process of trial and error. This problem is studied in many disciplines, such as information theory, genetic algorithm, game theory, etc.

2.2 Overview of Artificial Neural Network

To fully grasp the motivation behind ANN, one should study how the brain actually works. Under a microscope one can appreciate the intricate structure of a brain. In fact, the human brain consists of approximately 10^{11} interconnected brain cells, also called neurons. As illustrated in Fig. 2.1, a neuron consists of three parts, namely: the axon, the cell body and the dendrites. Electrical or chemical signals travel along the axon from dendrites to the other end of the axon called the axon terminal. Eventually, the axon connects to thousands of other neurons via their dendrites, forming a biological neural network (BNN). The juncture between the axon terminal and the dendrites is referred to as a synapse which serves like a gate, regulating the flow of information within the brain.

The findings from the research on human brain indicates the capabilities of BNN to learn and execute complex processes using a conceptually simple structure, yet comprising billions of interconnected neurons. Therefore, ANN aim to mimic BNN representation and problem solving abilities so that, in a similar way, they can be trained to solve engineering problems.

As shown in Fig. 2.2, the artificial neuron consists of several components, namely: the

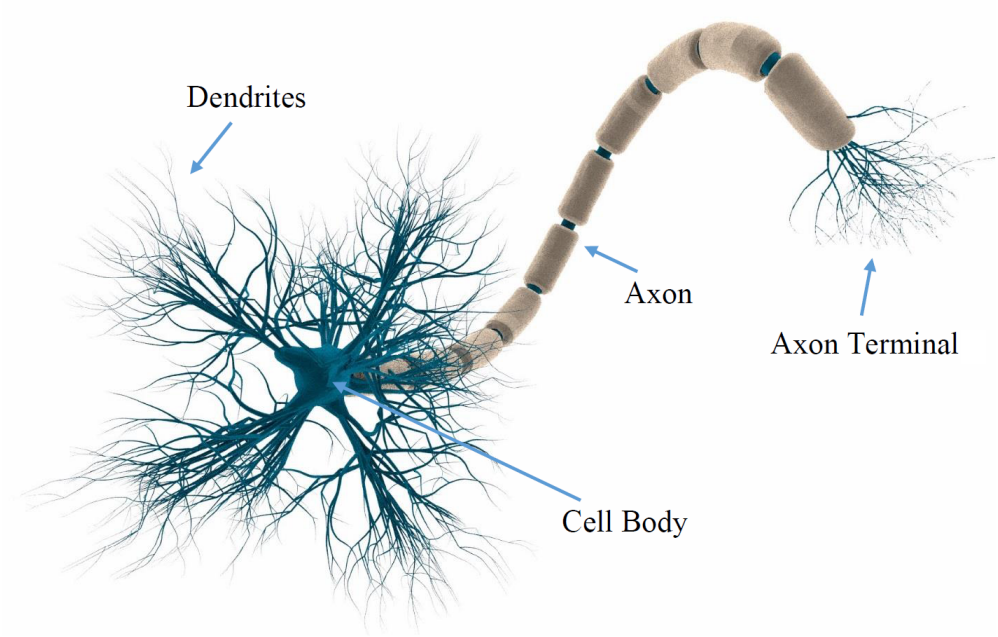

Fig. 2.1 Structure of a biological neuron (taken from [1])

inputs, weights, accumulator, activation function, and output [44]. The input feature vector is defined by $\mathbf{x} = [x_1, x_2, \ldots, x_I]$, where I denotes the size. The individual features x_i can take various forms: in the case of speech, they can be STFT or MFCC coefficients while for images, they can be individual pixel values.

The weight vector is defined as $\mathbf{w} = [w_1, w_2, \ldots, w_I]$ where w_i are the individual weights. These are often referred to as synaptic weights since they mimic the sensitivity of synaptic connections in a BNN. In the ANN they are used to scale the contributions from the inputs. The accumulator then calculates the weighted sum of the inputs. A bias value w_0 is added to attain an affine transformation prior to the application of the activation function $f(.)$. The latter will decide on weather or not the neuron should "fire". A mathematical representation of the artificial neuron is expressed as,

$$z = \sum_{i=1}^{I} w_i x_i + w_0 \tag{2.1}$$

$$y = f(z) \tag{2.2}$$

where z is the accumulator output and y is the output of the neuron after applying the

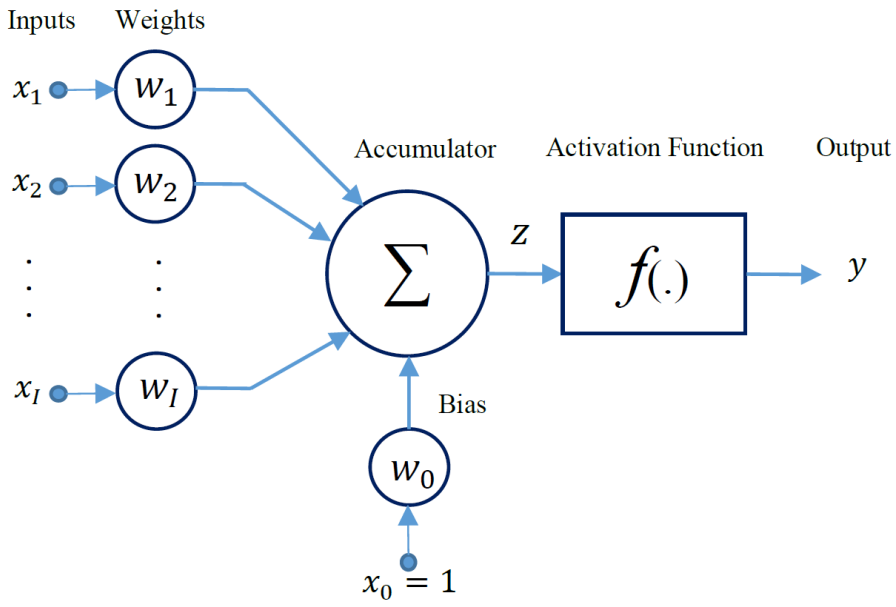

Fig. 2.2 Block diagram of an artificial neuron

activation function.

A neural network is obtained through the interconnection of several neurons. There are three main network structures, that is: feed-forward neural network (FFNN), recurrent neural network (RNN) and conventional neural network (CNN) [44]. The simplest and most commonly used neural network model is the FFNN, also known as the multilayer perceptron (MLP). Fig. 2.3 shows a fully connected FFNN, where all the nodes in each layer are connected to all the nodes in the successive layer, although there are no connections between the nodes in the same layer.

In this structure, each hidden layer is labeled with index $l \in \{1, 2, \ldots, L-1\}$, where L is the total number of layers. The input and output layers are represented by $\mathbf{h}^{(0)} = \mathbf{x} \in \mathbb{R}^{I_0}$, and $\mathbf{h}^{(L)} = \mathbf{y}^{(L)} \in \mathbb{R}^{I_L}$, where $I_0 = I$ is the number of input features and $I_L = K$ is the number of output values, respectively. Each hidden layer is represented $\mathbf{h}^{(l)} = \mathbf{y}^{(l)} \in \mathbb{R}^{I_l}$, where index $l \in \{1, 2, \ldots, L-1\}$, $\mathbf{y}^{(l)} = [y_1^{(l)}, y_2^{(l)}, \ldots, y_{I_l}^{(l)}]$ and I_l is the number of neurons in the l-th layer.

Depending upon the objective of the prediction task, supervised learning can fall into two categories of problems, i.e., classification or regression. In the classification, the objective is to take an input vector $\mathbf{x}$ and to assign it to one of C discrete classes t_C, where

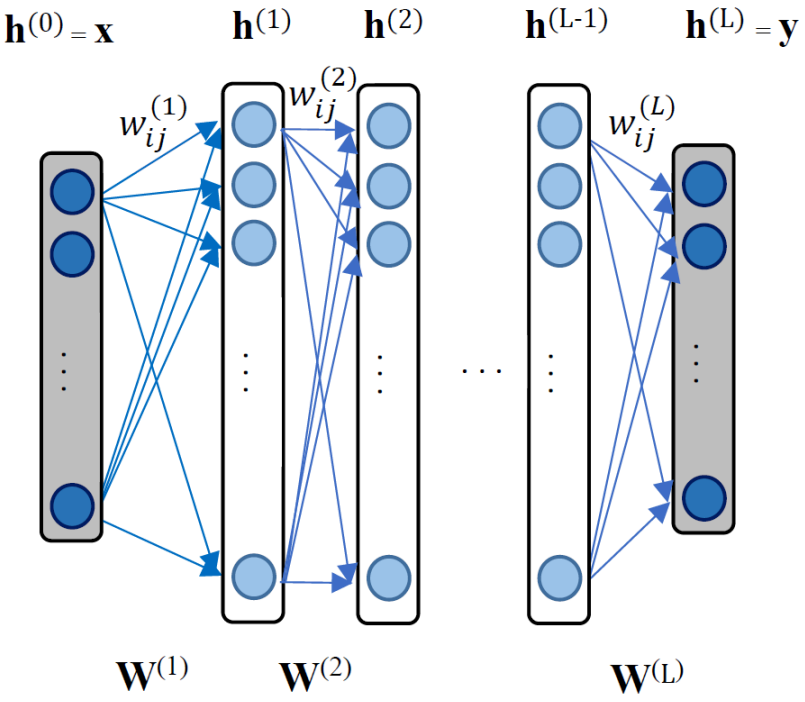

Fig. 2.3 Example of a feed-forward neural network

$c \in \{1, 2, \ldots, C\}$. It is often assumed that the classes are disjoint and that input vector can only belong to a single class. Hence, in the case of classification task, $K = 1$ and the output variable $y_1^{(L)}$ belongs to a finite discrete set. An example is provided by email spam filtering, where the goal of prediction is to determine weather an email is a spam or not. In contrast, the goal of the regression problem is to predict K continuous targets given an I-dimensional input vector $\mathbf{x}$. In other words, if the output variable $y_k^{(L)}$ that we want to predict is a real number, i.e., $y_k^{(L)} \in \mathbb{R}$, the prediction problem is called regression. An example of such a variable could be the speed control of a self-driving car, where the output variable is continuous and positive.

In general, the activation function $f(.)$ is the identity function in the case of regression and a non-linear function for classification models. Each neuron in the hidden and output layers of a FFNN computes a linear combination of its inputs, which can be expressed as,

$$z_i^{(l)} = \sum_{j=0}^{I_{l-1}} w_{ij}^{(l)} \, y_j^{(l-1)} \tag{2.3}$$

where $w_{ij}^{(l)}$ is the (i,j)-th entry of a linear transformation matrix $\mathbf{W}^{(l)} \in \mathbb{R}^{I_l \times I_{l-1}}$ where $l \in \{1, 2, \ldots, L\}$ is the layer index and I_l denotes the number of neurons in the l-th layer. In this notation, the bias values for each layers are absorbed in the weight matrix $\mathbf{W}^{(l)}$, i.e., $\mathbf{W}^{(l)} = [\mathbf{w}_0^{(l)}, \mathbf{w}_1^{(l)}, \ldots, \mathbf{w}_{I_{l-1}}^{(l)}]$, where $\mathbf{w}_i^{(l)} = [w_{1i}^{(l)}, w_{2i}^{(l)} \ldots, w_{I_l i}^{(l)}]^T$ and $y_0^{(l-1)} = 1$.

The output values of each neuron in the l-th layer are then transformed using an activation function, as expressed by,

$$y_i^{(l)} = f(z_i^{(l)}) \tag{2.4}$$

where $f(.)$ is often (but not recursively taken)as a differentiable, non-linear function (see below). Thus, combining these stages yields an overall network function that takes the following form for the l-th layer:

$$y_i^{(l)} = f(\sum_{j=0}^{I_{l-1}} w_{ij}^{(l)}\, y_j^{(l-1)}) \tag{2.5}$$

The role of the activation functions is to map or compress the permissible amplitude range of the output signal to some other more appropriate range. As such, the choice of activation function depends upon the application. Among all types of activation functions used in ANN systems, the following list provides a mathematical description of the most commonly used ones [45, 46]:

- The output of the linear (identity) function is proportional to its input, i.e., the weighted sum from the neurons. It is often used at the output of the network for regression problems. This activation function is expressed as,

$$f_{\text{lin}}(x) = x \tag{2.6}$$

- The rectified linear unit (ReLU) function applies a threshold to its input: it only retains the positive values and outputs zero else. It is less computationally expensive than some of the functions presented below as it involves simple mathematical operations, expressed by,

$$f_{\text{ReLU}}(x) = \max(0, x) = \begin{cases} x, & \text{for } x \geqslant 0 \\ 0, & \text{for } x < 0 \end{cases} \tag{2.7}$$

- The sigmoid or logistic function is the most common activation function; it is continuous, nonlinear, and differentiable. The output of this function takes value in the

range of 0 to 1 through the following definition,

$$f_{\text{sig}}(x) = \frac{1}{1 + e^{-x}} \tag{2.8}$$

- The hyperbolic tangent function is defined as the ratio between the hyperbolic sine and cosine. It shares similar characteristics with the sigmoid function: it is continuous, non-linear and differentiable. However, its output takes values in the range of -1 to 1, through the following definition,

$$f_{\tanh}(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.9}$$

- The unit step and sign (signum) functions change their output state from 0 to 1, and -1 to 1, receptively, depending on the sign of the input. These kinds of step functions are useful for binary classification problems. They are formally defined as,

$$f_{\text{step}}(x) = \begin{cases} 1, & \text{for } x \geqslant 0 \\ 0, & \text{for } x < 0 \end{cases} \tag{2.10}$$

$$f_{\text{sign}}(x) = \begin{cases} 1, & \text{for } x \geqslant 0 \\ -1, & \text{for } x < 0 \end{cases} \tag{2.11}$$

Fig. 2.5 shows the graph of the most common activation functions as discussed above.

2.3 Network Training

The goal of training the ANN is to adapt its weight parameters in order to achieve a desired behavior. In the context of supervised learning, the network is presented with large amount of data and attempts to correct its internal parameters so that its output values can best match the desired values provided in the training data set. In general, to train a FFNN model, one has to decide upon a performance measure or cost function. A cost function provides a quantitative measure of the mismatch or error between the network output values and the desired values; it is used to evaluate how well the model is trained by its training data. Two typical cost functions are the mean-squared error (MSE) and the cross-entropy (CE), which respectively take the form,

$$E_{\text{MSE}}(\{\mathbf{W}^{(l)}\}) = \sum_{i=1}^{K} \parallel y_i^{(L)} - t_i \parallel^2 \tag{2.12}$$

$$E_{\mathrm{CE}}(\{\mathbf{W}^{(l)}\}) = -\sum_{i=1}^{K}\{t_i \ln y_i^{(L)} + (1-t_i)\ln(1-y_i^{(L)})\} \tag{2.13}$$

where $y_i^{(L)}$ and t_i denote the estimated and target values, respectively. The neural network parameters, represented by $\{\mathbf{W}^{(l)} : l = 1, 2, \ldots, L\}$, are estimated by minimizing the chosen cost function $E(\{\mathbf{W}^{(l)}\})$. On the basis of the problem being solved, the activation function and cost function can be chosen accordingly. In the case of regression problems, the MSE function is often used in combination with the linear activation function. However, in the case of classification, the cross-entropy is often used with either the softmax or sigmoid activation function [45].

2.3.1 Backpropagation Algorithm

It is essential for the learning algorithm of an ANN to find the weight parameters, as represented by $\{\mathbf{W}^{(l)}, l = 1, 2, \ldots, L\}$, which minimize the chosen error function $E(\{\mathbf{W}^{(l)}\})$. Many algorithms exist for this purpose which take an iterative form, in which the weight matrices are updated as $\mathbf{W}^{(l)}(\tau+1) = \mathbf{W}^{(l)}(\tau) + \Delta\mathbf{W}^{(l)}(\tau)$, where integer τ is the iteration index, $\mathbf{W}^{(l)}(\tau)$ is the estimated value of $\mathbf{W}^{(l)}$ at iteration τ, and $\Delta\mathbf{W}^{(l)}(\tau)$ is the weight update. While different algorithms make use of different weight updates $\Delta\mathbf{W}^{(l)}(\tau)$ at each iteration, most often the gradient information is involved in the derivation of training algorithms. For instance, the gradient descent algorithm takes the general form,

$$\mathbf{W}^{(l)}(\tau+1) = \mathbf{W}^{(l)}(\tau) - \mu\nabla\mathbf{W}^{(l)}(\tau) \tag{2.14}$$

where the gradient of the cost function, denoted as $\nabla\mathbf{W}^{(l)}(\tau)$, formally defined as,

$$\nabla\mathbf{W}^{(l)}(\tau) = \frac{\partial E(\{\mathbf{W}^{(l)}\})}{\partial\mathbf{W}^{(l)}}\Big|_{W^{(l)}=W^{(l)}(\tau)} \tag{2.15}$$

and $\mu > 0$ is a step size which controls the learning rate.

Some techniques require using the entire training dataset to be processed at each step in order to update $E(\{\mathbf{W}^{(l)}\})$ and the weight matrices $\mathbf{W}^{(l)}(\tau)$; these are referred to as batch methods. On-line algorithms such as the gradient descent iteration, in contrast, update the weight matrices by processing one data point at a time.

The learning rule for FFNN comprises two stages, namely: feed-forward propagation

and feed-backward propagation. In the former stage, input data are supplied to the network input and propagate forward to calculate the output of the hidden layers, ultimately reaching the output layer. In the latter stage, the error propagates back through the network and the weights of each layer are updated in the direction that reduces $E(\{\mathbf{W}^{(l)}\})$. Here, the backpropagation update is derived using the MSE error function and gradient descent optimization as the learning rule.

For simplicity in notation, the error $E(\{\mathbf{W}^{(l)}\})$ will be denoted as E, the iteration index τ will be dropped and the update rule for the bias vectors will not be mentioned explicitly. Let $w_{ij}^{(l)}$ denote the weight connecting the j-th neuron from the $(l-1)$-th layer to the i-th neuron in the l-th layer, and $y_i^{(l)}$ denote the output of the i-th neuron in the l-th layer as in (2.5). Backpropagation provides information about the influence of the weights and biases on the cost function in a network. It involves an iterative procedure for minimizing the error function by updating its internal parameters. In this regard, the partial derivative of the error function with respect to the weights in the output layer, $\frac{\partial E}{\partial w_{ij}^{(L)}}$, are first evaluated. Following this step, the partial derivatives of the error function with respect to the weights in the hidden layers, $\frac{\partial E}{\partial w_{ij}^{(l)}}$, are evaluated.

Beginning with the output layer (i.e. $l = L$), we define the quantity, $\delta_i^{(L)}$, which will represent the accumulated error at each neuron. Specifically, the quantity $\delta_i^{(L)}$, provides a measure of how much the network error varies with the input to neuron i [45, 47, 48]. It is defined as,

$$\begin{aligned}\delta_i^{(L)} &= \frac{\partial E}{\partial z_i^{(L)}} \\ &= \frac{\partial E}{\partial y_i^{(L)}} \frac{\partial y_i^{(L)}}{\partial z_i^{(L)}} \\ &= \frac{\partial E}{\partial y_i^{(L)}} f'(z_i^{(L)})\end{aligned} \tag{2.16}$$

where (2.14) here has been invoked in (2.16), the first term $\frac{\partial E}{\partial y_i^{(L)}}$ indicates the rate of change of the cost function with respect to the i-th output, whereas, the second term $f'(z_i^{(L)})$, represents the rate of change of the activation function $f(.)$ with respect to the input $z_i^{(L)}$. In particular, in the case of the MSE cost function in (2.12), we have:

$$\frac{\partial E}{\partial y_i^{(L)}} = y_i^{(L)} - t_i \tag{2.17}$$

Thus, by inserting (2.16) and (2.17) in (2.15), $\frac{\partial E}{\partial w_{ij}^{(L)}}$ takes the form,

$$\frac{\partial E}{\partial w_{ij}^{(L)}} = y_j^{(L-1)} \delta_i^{(L)} = y_j^{(L-1)} (y_i^{(L)} - t_i) f'(z_i^{(L)}) \tag{2.18}$$

This results resonates with the physical meaning of the error signal: it defines the partial derivative of the error function with respect to the weights in the output layer as a product of the error term $\delta_i^{(L)}$ at neuron i in the L-th layer and the output $y_j^{(L-1)}$ of the j-th neuron in the $(L-1)$-th layer.

Next, the error $\delta^{(l)}$ in any layer l in the network can be represented with respect to the error in the next layer, $\delta^{(l+1)}$, through the following expression,

$$\begin{aligned} \delta_j^{(l)} &= \frac{\partial E}{\partial z_j^{(l)}} \\ &= \sum_{i \in I_{l+1}} \frac{\partial E}{\partial z_i^{(l+1)}} \frac{\partial z_i^{(l+1)}}{\partial z_j^{(l)}} \\ &= \sum_{i \in I_{l+1}} \frac{\partial z_i^{(l+1)}}{\partial z_j^{(l)}} \delta_i^{(l+1)} \end{aligned} \tag{2.19}$$

where

$$z_i^{(l+1)} = \sum_{j \in I_l} w_{ij}^{(l+1)} y_j^{(l)} = \sum_{j \in I_l} w_{ij}^{(l+1)} f(z_j^{(l)}) \tag{2.20}$$

The first term in (2.19), the partial derivative $\frac{\partial z_i^{(l+1)}}{\partial z_j^{(l)}}$, can be expressed as,

$$\frac{\partial z_i^{(l+1)}}{\partial z_j^{(l)}} = w_{ij}^{(l+1)} f'(z_j^{(l)}) \tag{2.21}$$

Furthermore, substituting (2.21) in (2.19), yields,

$$\delta_j^{(l)} = \sum_{i \in I_{l+1}} w_{ij}^{(l+1)} f'(z_j^{(l)}) \delta_i^{(l+1)} \tag{2.22}$$

Thus, the rate of change of the error with respect to each weight in the hidden layer, $w_{ij}^{(l)}$, in the network is given by,

$$\frac{\partial E}{\partial w_{ij}^{(l)}} = y_j^{(l-1)}\delta_j^{(l)} = y_j^{(l-1)} \sum_{k \in I_{l+1}} w_{kj}^{(l+1)} f'(z_j^{(l)})\delta_k^{(l+1)} \tag{2.23}$$

The above derivatives are then used to calculate the adjustments to be made on the weights as in (2.14).

In conclusion, the backpropagation procedure is summarized as follows,

1. Apply the input data to the network and forward propagate through the network to compute the output values for each layer $l \in \{2, , \ldots, L\}$.

2. Evaluate the error $\delta_i^{(L)}$ for all output neurons as in (2.16).

3. Evaluate the error $\delta_i^{(l)}$ for each hidden neurons in layer $l = L-1, L-2, \ldots, 1$ as in (2.22).

4. Update each network bias and weight using the gradient descent algorithm in (2.14).

2.4 Regularization

The goal of machine learning is to properly train the system model from the given data, in order to make a good prediction for the new unseen data. The ability to perform well on the new unseen data, called *generalization*, depends on the size of the training data set, the number of free parameters in the prediction model, and the actual physical property of the data (or true model).

To better illustrate this point, Fig. 2.4 shows three polynomial models of degree 1, 4, and 16, each attempting to approximate a sine function. The models (hypotheses) have polynomial features of different degrees. The linear model, i.e. polynomial with degree 1, is not adequate to fit the samples of the true function. Consequently, it gives a poor representation of the sine function. This is called *underfitting*. However, the 4-th order polynomial model seems to be a good fit to the true function. For higher order polynomials, i.e. polynomial with degree 16, we achieve perfect fit to the samples of the true function with zero error. This is achieved by passing the polynomial curve through each of the

samples. This causes the fitted curve to oscillate and to give a poor representation of the sine function. This latter behavior is known as *overfitting*.

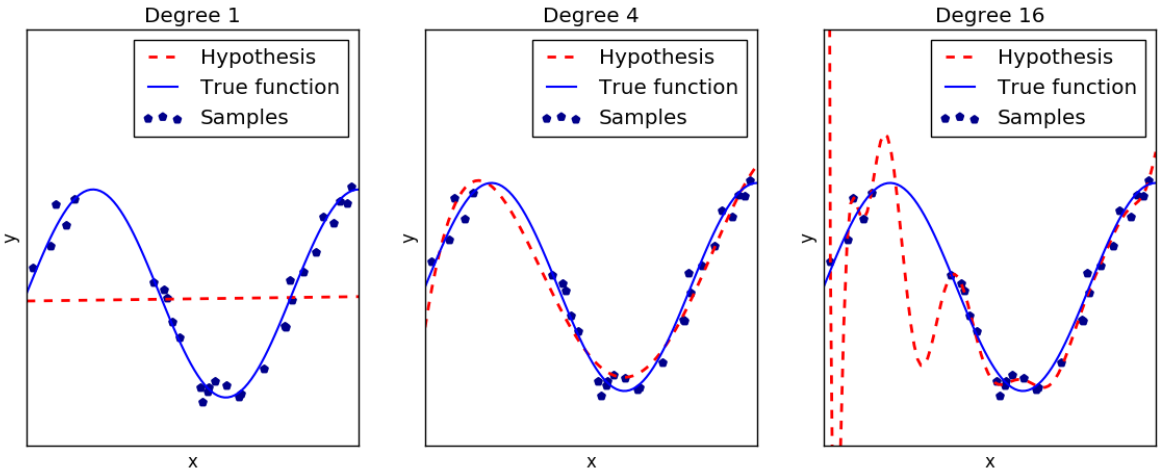

Fig. 2.4 Underfitting and Overfitting

In a DNN structure, the numbers of layers L and hidden neurons I_l in each layer $l \in \{1, 2, \ldots, L-1\}$ control the number of parameters, i.e. weights $w_{ij}^{(l)}$ and biases $b_i^{(l)}$, which can be adjusted to provide the best predictive performance. It is vital to find optimum values of $\{I_1, \ldots, I_L\}$ that give the best generalization performance, corresponding to the optimum balance between underfitting and overfitting. When a model learns too well the details and the noise from the training data, often it will not generalize well on different data that were not seen in the training phase. This is known as overfitting and is one of the problems afflicting neural network. In order to prevent overfiting, regularization techniques can be used. They are achieved by penalizing the size of neural network weights. A weight penalty allows a DNN to discard large unwanted values of the weight parameters. A general representation of regularization as given in [45] can take the following form,

$$E(\{\mathbf{W}^{(l)}\}) = \sum_{i \in K} (y_i^{(L)} - t_i)^2 + \lambda \sum_{l=1}^{L} |\mathbf{W}^{(l)}|^q \tag{2.24}$$

where $\lambda > 0$ denotes the regularization parameter. When $q = 2$, (2.24) refers to the quadrature or the Ridge regularization function, often known as the weight decay [48], while $q = 1$ corresponds to the Lasso regularization [48].

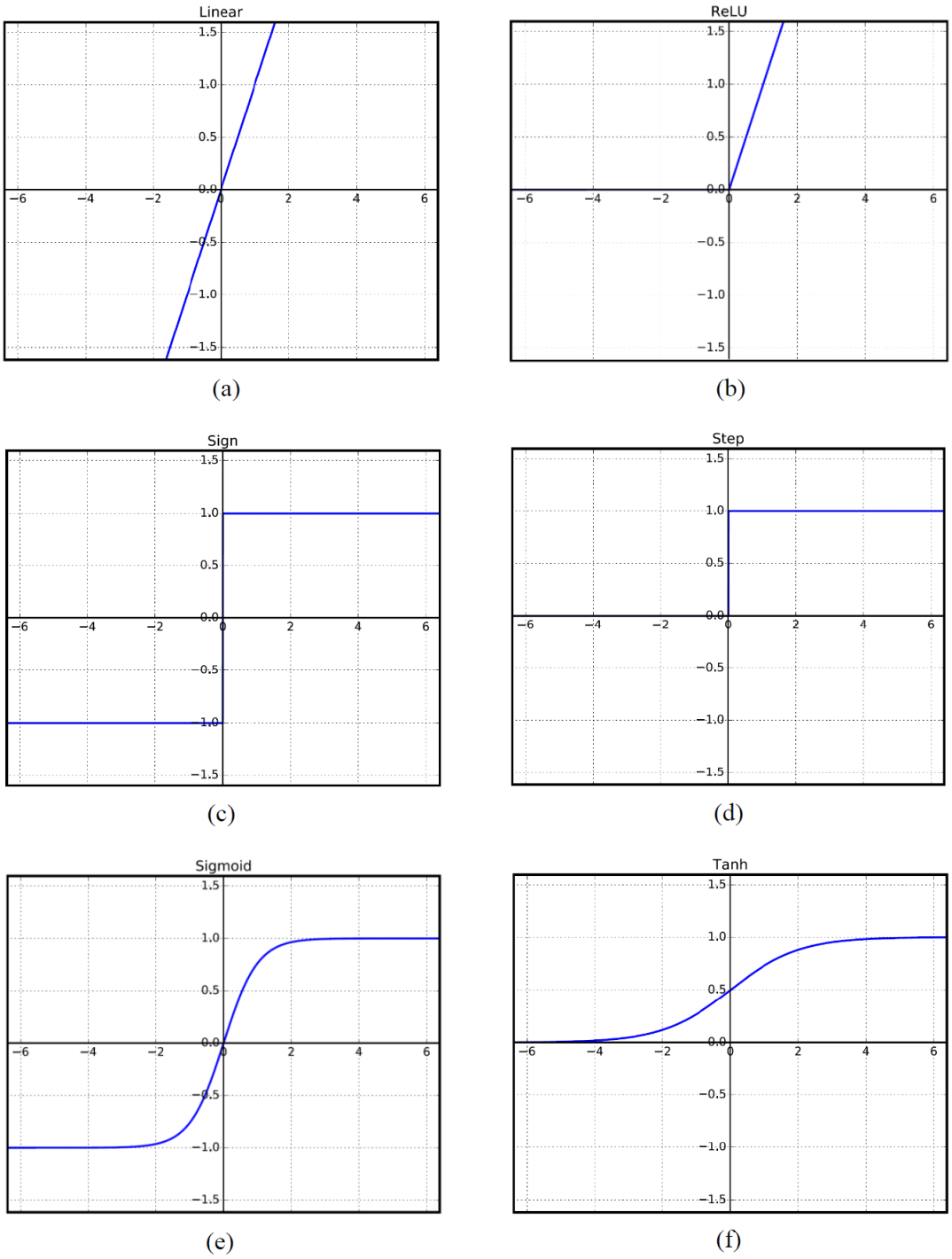

Fig. 2.5 Illustration of the most common activation functions: (a) linear; (b) rectified linear unit (ReLU); (c) sign; (d) step; (e) sigmoid and (f) tanh

Chapter 3

Neural Network for Speech Denoising

In this chapter, we first review the standard audio feature extraction and signal reconstruction used in speech processing. Then the basic features of a STFT-based DNN structure for speech enhancement and associated training procedure are discussed. Following that, different resilient backpropagation techniques are provided.

3.1 Feature Extraction and Speech Reconstruction

The vast majority of single-channel speech enhancement methods implement the analysis-modification-synthesis (AMS) framework in the acoustic frequency domain [49]- [50] which will be presented in this section. In general, a noisy speech signal can be expressed as a combination of additive and convolution noise, or so-called mixed noise, as illustrated in Fig. 3.1. However, in this work for simplicity, the noisy speech signal is assumed to result from the combination of additive noise only with the clean signal, i.e.,

$$y[n] = x[n] + d[n] \tag{3.1}$$

where $y[n]$, $x[n]$ and $d[n]$ respectively denote the noisy speech, the clean speech and the additive background noise, and $n \in \mathbb{Z}$ is the discrete-time index. The noisy speech spectrum, obtained via short-time Fourier transform (STFT), can be expressed as,

$$Y(\nu, k) = X(\nu, k) + D(\nu, k) \tag{3.2}$$

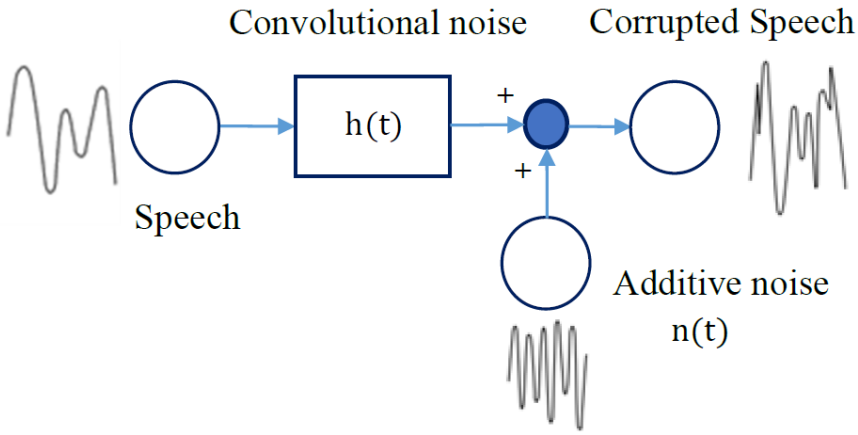

Fig. 3.1 Noise environment structure

where $Y(\nu, k)$, $X(\nu, k)$ and $D(\nu, k)$ refer to the STFT coefficients of the noisy speech, clean speech and noise at the (ν, k)-th time-frequency bin, respectively. Specifically, the STFT is defined by the following relation,

$$Y(\nu, k) = \sum_{n=-\infty}^{\infty} y[n]\phi(n - \nu F)\, e^{j2\pi kn/K} \tag{3.3}$$

where $\nu \in \mathbb{Z}$ refers to the frame index, positive integer F is the frame advance, $k \in \{0, 1, \ldots, K-1\}$ is the frequency index and $\phi(n)$ is a windowing function of length K, chosen as a Hanning window [51] in this work. In effect, the input signal $y[n]$ is first decomposed into consecutive (overlapping) segments of length K. Then for each segment, the window function is applied and a K-point discrete Fourier transform (DFT) is computed.

In this work, the enhanced speech spectrum denoted as $\hat{X}(\nu, k)$, is estimated by means of a DNN structure followed by Wiener filtering, where the detailed enhancement and training procedures will be explained in the sequel. The enhanced speech signal in the time-domain, $\hat{x}(n)$ is then obtained by first applying the inverse STFT to the estimated clean speech spectrum $\hat{X}(\nu, k)$, followed by the overlap-add method (OLA) [50], as given

by the following expressions,

$$\hat{x}_\nu(n) = \frac{1}{K}\sum_{k=0}^{K-1} \hat{X}(\nu,k)e^{j2\pi kn/K} \tag{3.4}$$

$$\hat{x}(n) = \sum_{\nu=-\infty}^{\infty} \hat{x}_\nu(n-\nu F)\,\phi(n-\nu F) \tag{3.5}$$

An important condition for OLA to yield perfect synthesis is that the sum of all the shifted copies of the analysis window adds up to a constant, c, i.e. [52],

$$\sum_{\nu=-\infty}^{\infty} \phi(n-\nu F) = c, \quad \forall n \in \mathbb{Z} \tag{3.6}$$

3.2 DNN Structure

The architecture adopted for the speech enhancement system developed in this thesis is based on a feed-forward DNN consisting of multiple non-linear hidden layers. This architecture, shown in Fig. 3.2, allows to model a highly non-linear regression function, which maps noisy speech features at the input into clean speech features at the output.
Referring to Fig. 3.2, each hidden layer, labeled with index $l \in \{1,..,L-1\}$, where L is the total number of layers, consists of I_l neurons. The output values of the l-th layer are represented by vector $\mathbf{h}^{(l)} \in \mathbb{R}^{I_l}$ and can be expressed as,

$$\mathbf{h}^{(l)} = f\left(\mathbf{W}^{(l)}\,\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}\right) \tag{3.7}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{I_l \times I_{l-1}}$ is a linear transformation matrix with (i,j)-th entry $w_{ij}^{(l)}$, $\mathbf{b}^{(l)} \in \mathbb{R}^{I_l}$ is a bias vector with i-th entry $b_i^{(l)}$, and $f(.)$ represents a non-linear activation function which operates element-wise. Depending on the application, the activation function can be selected accordingly, such as a sigmoid or piecewise linear function [46]. However, the rectified linear unit function, $f_{\mathrm{ReLU}}(.)$ as defined in (2.7) according to [31], turns out to be more effective in our prediction problem.

In the DNN architecture of Fig. 1, the input (bottom) layer consists of the noisy spectrum magnitudes at the ν-th frame. Specifically $\mathbf{h}^{(0)} = \boldsymbol{\mathcal{Y}}_\nu \equiv [\mathcal{Y}_{\nu,0}, \mathcal{Y}_{\nu,1}, \ldots, \mathcal{Y}_{\nu,I_0-1}]^T$ where $\mathcal{Y}_{\nu,k} = |Y(\nu,k)|$ and $I_0 = K$. The output (top) layer in Fig. 1, represented by vector

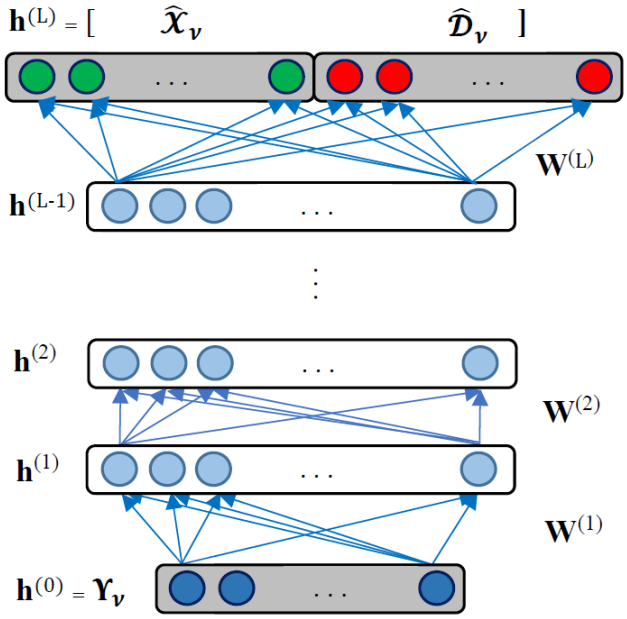

Fig. 3.2 Feed forward DNN

$\mathbf{h}^{(L)} \in \mathbb{R}^{I_L}$, is obtained through a linear regression as,

$$\mathbf{h}^{(L)} = \mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)} \tag{3.8}$$

where $\mathbf{W}^{(L)} \in \mathbb{R}^{I_L \times I_{L-1}}$ and $\mathbf{b}^{(L)} \in \mathbb{R}^{I_L}$. For the output layer, we adopt a special configuration where $I_L = 2K$ and $\mathbf{h}^{(L)} = [\hat{\boldsymbol{\mathcal{X}}}_\nu, \hat{\boldsymbol{\mathcal{D}}}_\nu]$ consists of two K-dimensional prediction vectors. In this notation, $\hat{\boldsymbol{\mathcal{X}}}_\nu = [\hat{\mathcal{X}}_{\nu,0}, \hat{\mathcal{X}}_{\nu,1}, \ldots, \hat{\mathcal{X}}_{\nu,K-1}]^T$ and $\hat{\boldsymbol{\mathcal{D}}}_\nu = [\hat{\mathcal{D}}_{\nu,0}, \hat{\mathcal{D}}_{\nu,1}, \ldots, \hat{\mathcal{D}}_{\nu,K-1}]^T$. The components $\hat{\mathcal{X}}_{\nu,k}$ and $\hat{\mathcal{D}}_{\nu,k}$ provide preliminary estimates of the clean speech and noise spectrum magnitudes, that is $\mathcal{X}_{\nu,k} \equiv |X(\nu,k)|$ and $\mathcal{D}_{\nu,k} \equiv |D(\nu,k)|$, respectively.

The predicted spectrum of the clean speech at the ν-th frame is finally obtained from the DNN output by applying the Wiener filter [53] as given by,

$$\hat{X}(\nu,k) = \frac{P_{\mathcal{X}}(\nu,k)}{P_{\mathcal{X}}(\nu,k) + P_{\mathcal{D}}(\nu,k)}\, Y(\nu,k) \tag{3.9}$$

In this expression, the quantities $P_{\mathcal{X}}(\nu,k)$ and $P_{\mathcal{D}}(\nu,k)$ represent the smoothed clean speech and noise power spectral densities (PSDs) for the k-th frequency bin and ν-th frame. They

are computed recursively over the frame index as,

$$P_{\mathcal{X}}(\nu,k) = \tau_x P_{\mathcal{X}}(\nu-1,k) + (1-\tau_x)\hat{\mathcal{X}}^2_{\nu,k} \tag{3.10}$$

$$P_{\mathcal{D}}(\nu,k) = \tau_d P_{\mathcal{D}}(\nu-1,k) + (1-\tau_d)\hat{\mathcal{D}}^2_{\nu,k} \tag{3.11}$$

where τ_x and τ_d, taken within the interval of $[0,1]$, denote the temporal smoothing factors for the clean speech and noise, respectively.

3.3 Training Procedure

In the training stage, we estimate the weight matrices $\mathbf{W}^{(l)}$ and bias vectors $\mathbf{b}^{(l)}$ for all layer $l \in \{1,2,\ldots,L\}$ by employing training data, represented by the triplet $\{\boldsymbol{\mathcal{Y}},\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{D}}\}$. The latter consists of the input noisy speech matrix $\boldsymbol{\mathcal{Y}} = [\boldsymbol{\mathcal{Y}}_1,\ldots,\boldsymbol{\mathcal{Y}}_N]$, clean speech target $\boldsymbol{\mathcal{X}} = [\boldsymbol{\mathcal{X}}_1,\ldots,\boldsymbol{\mathcal{X}}_N]$, and clean noise target $\boldsymbol{\mathcal{D}} = [\boldsymbol{\mathcal{D}}_1,\ldots,\boldsymbol{\mathcal{D}}_N]$, where N is the total number of frames available for training.

The DNN parameters are estimated by minimizing a suitable cost function. Among the different cost functions available for this task, such as the mean-squared error (MSE), the cross-entropy, the Kullback Leibler divergence and the Itakura Staito divergence, the minimum MSE (MMSE) seems to be an appropriate choice for the speech enhancement problem [22]. The MSE function of the DNN output is calculated as,

$$E = \frac{1}{N}\sum_{n=1}^{N} \| \,[\hat{\boldsymbol{\mathcal{X}}}_n,\hat{\boldsymbol{\mathcal{D}}}_n] - [\boldsymbol{\mathcal{X}}_n,\boldsymbol{\mathcal{D}}_n]\, \|_2^2 + \lambda \sum_{l=1}^{L} \|\mathbf{W}^{(l)}\|_2^2 \tag{3.12}$$

where $[\hat{\boldsymbol{\mathcal{X}}}_n,\hat{\boldsymbol{\mathcal{D}}}_n]$ and $[\boldsymbol{\mathcal{X}}_n,\boldsymbol{\mathcal{D}}_n]$ denote the estimated and target spectral feature vectors of the clean speech and noise pair, respectively. In order to avoid overfitting, as previously discussed in Section 2.4, Ridge regularization is considered through the term $\lambda \sum_{l=1}^{L} \|\mathbf{W}^{(l)}\|_2^2$, where $\lambda > 0$ is the regularization parameter.

3.3.1 Rprop Algorithms

One can use the error backpropagation technique to estimate the parameters that minimize the cost function in (3.12), such as the common stochastic gradient descent algorithm, or more complex methods such as the conjugate gradient or Levenberg-Marquardt algo-

rithms [44]. In addition, there is an interest towards using an additional greedy layer-wise pre-training stage via the RBM [29, 54] or autoencoder techniques [30]. However, these approaches are computationally expensive and do not seem to critically affect the final enhancement performance of the DNN with ReLU activation function, given sufficiently large and varied data set [31]. In this work therefore, we choose an improved version of the resilient backpropagation (Rprop) [55] algorithm, called iRprop$^-$ which is presented in [41]. As an alternative to iRprop$^-$, we have also implemented our proposed DNN approach using Adam optimization [56] in the training stage; however, no significant improvement was obtained in the performance of our model.

Rprop is a first-order iterative learning algorithm, which has been shown to provide a rapid and reliable convergence compared to the conjugate gradient algorithm, yet with much less computations. It performs a local adaptation of the weight-update term according to the behavior of the error function. In the standard backpropagation procedure, the network is trained to minimize the discrepancy between the original clean speech and its estimate at the DNN output. This is achieved by back propagating the MSE value from the output to the input layer and adjusting the weights via gradient descent to reduce the error. In contrast, Rprop uses only the sign of the gradient, as apposed to its magnitude value, which leads to a more robust and faster converging weight update. In essence, it assumes that different weights need different step sizes for their update, which vary throughout the learning process [41, 55]. The main steps of Rprop algorithm are summarized below.

Let $t \in \{0, 1, \ldots, T\}$ denote the iteration index, T the maximum number of iterations, $E(t)$ the value of the cost function in (3.12) at iteration t and $\gamma_{ij}^{(l)}(t) = \partial E(t)/\partial w_{ij}^{(l)}$ the partial derivative of $E(t)$ with respect to $w_{ij}^{(l)}$. During the learning process, the value of each weight $w_{ij}^{(l)}$ at iteration t is updated based on the local information available about the evolution of the error function, as represented by $\gamma_{ij}^{(l)}(t)$. Specifically, a time-varying step-size is first computed as follows,

$$\Delta_{ij}^{(l)}(t) = \begin{cases} \eta^+ \, \Delta_{ij}^{(l)}(t-1), & \text{if } \gamma_{ij}^{(l)}(t-1)\gamma_{ij}^{(l)}(t) > 0 \\ \eta^- \, \Delta_{ij}^{(l)}(t-1), & \text{if } \gamma_{ij}^{(l)}(t-1)\gamma_{ij}^{(l)}(t) < 0 \\ \Delta_{ij}^{(l)}(t-1), & \text{else} \end{cases} \tag{3.13}$$

where $\eta+$ and $\eta-$ are predefined constants in the range of $0 < \eta^- < 1 < \eta^+$. In effect, if the error gradient for a given weight $w_{ij}^{(l)}$ has the same sign in two consecutive iterations,

we increase its step-size $\Delta_{ij}^{(l)}$ by the scaling factor η^+, since the weight's optimal value may be far away. However, if the derivative changes its sign, which confirms that the algorithm jumped over a local minimum, the step size is decreased by the factor η^-. Otherwise, it remains unchanged.

In the literature, Rprop has been proposed with and without weight-backtracking. Weight-backtracking refers to using a previous weight update for some or all weights. In case of Rprop without weight-backtracking, we use a negative superscript to indicate the weight-backtracking is omitted from the standard Rprop and denote it as Rprop$^-$. In particular, using Rprop$^-$ algorithm, there is no need to store the previous weight updates since for all cases they can be computed as,

$$\Delta w_{ij}^{(l)}(t) = -\text{sgn}(\gamma_{ij}^{(l)}(t))\Delta_{ij}^{(l)}(t) \tag{3.14}$$

where the function sgn(.) returns -1 when its argument is negative, +1 when its argument is positive, and 0 otherwise. After adjusting the weight updates as in (3.14), then each weight will be updated accordingly based on the following rule,

$$w_{ij}^{(l)}(t+1) = w_{ij}^{(l)}(t) + \Delta w_{ij}^{(l)}(t) \tag{3.15}$$

Algorithm 1 summarizes Rprop$^-$ in pseudo-code.

As opposed to Rprop$^-$, in iRprop$^-$ algorithm, when a change of sign of the partial derivative occurs, that derivative is set to zero [41], i.e.,

$$\gamma_{ij}^{(l)}(t) \leftarrow 0 \quad \text{if } \gamma_{ij}^{(l)}(t-1)\gamma_{ij}^{(l)}(t) < 0 \tag{3.16}$$

In other words, when the sign of a partial derivative changes, iRprop$^-$ reduces the corresponding step-size and does not modify the weight. Algorithm 2 shows the corresponding pseudo-code.

In the same manner, the bias vector $\mathbf{b}^{(l)} \in \mathbb{R}^{I_l}$ with i-th entry $b_i^{(l)}$, is updated using Rprop$^-$ and iRprop$^-$ algorithms for every layer, i.e. for all $l \in \{1, 2, \ldots, L\}$. However, for the sake of simplicity in notation, the bias vectors are not mentioned explicitly in our further developments [41, 55, 57].

Algorithm 1 Rprop$^-$

Initialization: $\eta^+, \eta^-, \Delta_{max}, \Delta_{min}, \Delta_{ij}^{(l)}(0)$

$\Delta w_{ij}^{(l)}(0) = \frac{\partial E(0)}{\partial w_{ij}^{(l)}} + \Delta_{ij}^{(l)}(0)$

while $t < T$ **do**

for each $w_{ij}^{(l)}$ **do**

if $\frac{\partial E(t-1)}{\partial w_{ij}^{(l)}} \frac{\partial E(t)}{\partial w_{ij}^{(l)}} > 0$ **then**

$\Delta_{ij}^{(l)}(t) = \min\left(\Delta_{ij}^{(l)}(t-1)\,\eta^+, \Delta_{max}\right)$

else if $\frac{\partial E(t-1)}{\partial w_{ij}^{(l)}} \frac{\partial E(t)}{\partial w_{ij}^{(l)}} < 0$ **then**

$\Delta_{ij}^{(l)}(t) = \max\left(\Delta_{ij}^{(l)}(t-1)\,\eta^-, \Delta_{min}\right)$

end if

$\Delta w_{ij}^{(l)}(t) = -\mathrm{sgn}(\frac{\partial E(t)}{\partial w_{ij}^{(l)}})\,\Delta_{ij}^{(l)}(t)$

$w_{ij}^{(l)}(t+1) = w_{ij}^{(l)}(t) + \Delta w_{ij}^{(l)}(t)$

end for

Algorithm 2 iRprop$^-$

Initialization: $\eta^+, \eta^-, \Delta_{max}, \Delta_{min}, \Delta_{ij}^{(l)}(0)$

$\Delta w_{ij}^{(l)}(0) = \frac{\partial E(0)}{\partial w_{ij}^{(l)}} + \Delta_{ij}^{(l)}(0)$

while $t < T$ **do**

for each $w_{ij}^{(l)}$ **do**

if $\frac{\partial E(t-1)}{\partial w_{ij}^{(l)}} \frac{\partial E(t)}{\partial w_{ij}^{(l)}} > 0$ **then**

$\Delta_{ij}^{(l)}(t) = \min\left(\Delta_{ij}^{(l)}(t-1)\,\eta^+, \Delta_{max}\right)$

else if $\frac{\partial E(t-1)}{\partial w_{ij}^{(l)}} \frac{\partial E(t)}{\partial w_{ij}^{(l)}} < 0$ **then**

$\Delta_{ij}^{(l)}(t) = \max\left(\Delta_{ij}^{(l)}(t-1)\,\eta^-, \Delta_{min}\right)$

$\frac{\partial E(t)}{\partial w_{ij}^{(l)}} = 0$

end if

$\Delta w_{ij}^{(l)}(t) = -\mathrm{sgn}(\frac{\partial E(t)}{\partial w_{ij}^{(l)}})\,\Delta_{ij}^{(l)}(t)$

$w_{ij}^{(l)}(t+1) = w_{ij}^{(l)}(t) + \Delta w_{ij}^{(l)}(t)$

end for

Chapter 4

Proposed Framework

In this section, we first introduce the proposed DNN framework for speech enhancement in general terms. Speech feature extraction based upon MFCC is then briefly reviewed, followed by the incorporation of MFCC features in the training and the enhancement stages of the DNN model. In addition, the non-negative matrix factorization method is briefly presented with its application in speech enhancement. Finally, we discuss the computational complexity of the proposed DNN scheme in comparison to a STFT-based DNN approach.

4.1 Proposed Structure and Motivation

Machine learning has received much attention in the field of speech enhancement and automatic speech recognition for a long time. The acoustic feature extraction plays a key role as a pre-processing stage to these tasks. The MFCCs are among the most commonly used features in this context as they provide a spectral representation of speech that incorporates some aspects of audition. The use of MFCC not only has the advantage of mimicking the logarithmic perception of the human auditory system [40], but it also leads to a reduction in the number of features as compared to the conventional STFT approach. In turn, this leads to reduced complexity for the learning and processing tasks.

The process of calculating the MFCC vectors from the observed speech signal includes some non invertible stages. It might be possible to make certain approximations about the information that has been discarded during this process to allow estimating the magnitude spectrum of the input speech as a result of the MFCC inversion process. Yet, it is still a challenge to ensure that the MFCC inversion process will achieve perfect reconstruction

without additional computational complexity.

Therefore, in this thesis, a spectral feature mapping from noisy MFCC to the enhanced STFT spectrum is introduced based on DNN modeling, in order to predict the clean speech signal from a noise corrupted input signal. Mapping the MFCC features directly into the frequency domain allows one to bypass the information loss caused by the inversion of the MFCC process. In addition, implementation of the spectral feature mapping technique using MFCC features has the advantage of reducing the length of the input feature vector. Hence, a smaller DNN model (i.e., with reduced number of nodes) can be employed. Consequently, this leads to a faster convergence time in training and lower computational complexity. These are the main motivations for the proposed approach, whose main processing steps are summarized below.

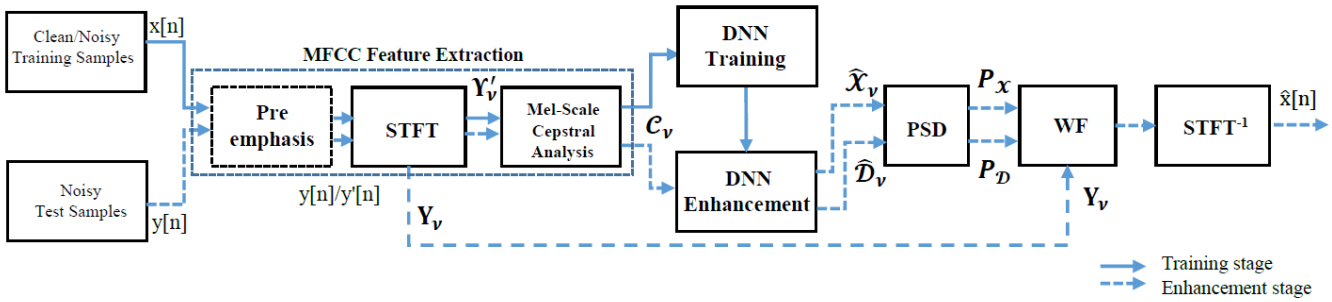

Fig. 4.1 Block diagram of the MFCC-based DNN system for speech enhancement

A block diagram of the proposed DNN approach for speech enhancement is illustrated in Fig. 4.1. The system operation consists of two stages, that is, training and enhancement. In both stages, first the input signal is passed into a MFCC feature extraction module to obtain the speech spectral features. In this module, following a pre-emphasis operation, short-time Fourier analysis is applied to the input speech signal by computing the STFT of overlapping windowed frames. Next, the MFCC feature vector of each frame is computed by applying the mel-scale cepstral analysis to the STFT coefficients. As a result, the high-dimensional input STFT data vector is transformed into a lower dimensional MFCC feature vector.

In the training stage, a regression-based DNN model is trained using the available training data which consists of the input noisy speech MFCC, the clean speech STFT target, and clean noise target, as represented by the triplet $\{\boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{D}}\}$. In the enhancement

stage, the clean speech magnitude spectrum is predicted from processing the noisy speech frames by the well-trained DNN model. Next, in the Wiener filtering module, the phase of the noisy speech is applied to the enhanced output, as explained in Section 3.3. In effect, as in the majority of available methods for speech enhancement, it is assumed that the phase information is not critical for the human auditory perception, so only an estimate of the magnitude spectrum of the speech is required [24]. Finally, to reconstruct the enhanced speech signal into the time domain, the inverse STFT is computed followed by the overlap-add method.

4.2 MFCC Features

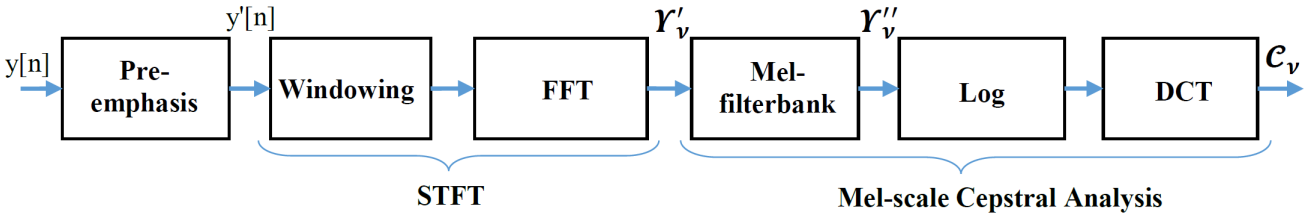

Fig. 4.2 MFCC feature extraction flowchart

This section describes the procedure for calculating the MFCC feature vectors of an audio signal, as illustrated in Fig. 4.2. First, in the pre-emphasis stage, the speech signal is passed through a first order FIR filter that boosts the highband formants. Specifically, the filter output signal, $y'[n]$, is computed as,

$$y'[n] = y[n] - \alpha y[n-1] \tag{4.1}$$

where α is a pre-emphasis coefficient, typically in the range $0.95 \leq \alpha \leq 1$. The boosted speech signal $y'[n]$ is then segmented into consecutive overlapping frames of length K with frame advance F, and each frame is multiplied with an analysis window, which in this work is chosen as a Hanning window [51]. Then for each windowed frame, a DFT is computed as explained in Subsection 3.1. The resulting STFT coefficients are denoted by $\mathcal{Y}'_\nu(k)$, where $\nu \in \mathbb{Z}$ refers to the frame index and $k \in \{0, 1, \ldots, K-1\}$ is the frequency index. For convenience, we define the STFT coefficient vector $\boldsymbol{\mathcal{Y}}'_\nu = [\mathcal{Y}'_\nu(0), \mathcal{Y}'_\nu(1), \ldots, \mathcal{Y}'_\nu(K-1)]$, which is shown in Fig. 4.2. The squared magnitude of the STFT coefficients, i.e., $|\mathcal{Y}'_\nu(k)|^2$,

are then passed through a mel-scale filterbank, consisting of M overlapping triangular pass-band filters [58], indexed by $m \in \{0, 1, .., M-1\}$. Specifically, for the m-th pass-band filter, the filter output denoted as $\mathcal{Y}''_\nu(m)$, is calculated as a weighted sum of the squared magnitude values within the corresponding pass-band [58], as expressed by,

$$\mathcal{Y}''_\nu(m) = \sum_{k=0}^{K-1} |\mathcal{Y}'_\nu(k)|^2 \, \mathcal{W}_{km} \tag{4.2}$$

where $\mathcal{W}_{km} \geqslant 0$ is the (k, m)-th entry of the filterbank matrix $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{K \times M}$. In this formulation, $\mathcal{Y}''_\nu(m)$ can be interpreted as the m-th passband mel-scale filter energy. Next, a logarithmic operation is applied to the filter outputs. Finally, the outputs are further processed by taking the Type III discrete cosine transform (DCT), as expressed by [40, 59],

$$\mathcal{C}_\nu(p) = \sqrt{\frac{2}{M}} \sum_{m=0}^{M-1} (\log_{10} \mathcal{Y}''_\nu(m)) \cos(\frac{p\pi}{M}(m - 0.5)) \tag{4.3}$$

where $\mathcal{C}_\nu(p)$ refers to the p-th MFCC, $p \in \{0, 1, \ldots, P-1\}$, and P is the number of mel-scale cepstral coefficients. For convenience, we define the vector $\boldsymbol{\mathcal{Y}}''_\nu = [\mathcal{Y}''_\nu(0), \mathcal{Y}''_\nu(1), \ldots, \mathcal{Y}''_\nu(M-1)]$ and $\boldsymbol{\mathcal{C}}_\nu = [\mathcal{C}_\nu(0), \mathcal{C}_\nu(1), \ldots, \mathcal{C}_\nu(P-1)]$, also shown in Fig. 4.2.

Although MFCC have been successfully applied in speech processing, it has been shown that the performance of the speech recognition can be improved by applying a liftering operation to the cepstral coefficients [60]. In this operation, the liftered coefficients are obtained as,

$$\mathcal{L}_\nu(p) = w(p)\mathcal{C}_\nu(p) \tag{4.4}$$

where the weights $w(p)$, for $p \in \{1, 2, \ldots, P\}$, define the lifter. Several types of lifters have been proposed in the literature [60], the most commonly used ones being briefly reviewed below:

- The linear lifter which can be expressed as,

$$w(p) = p \tag{4.5}$$

- The statistical lifter takes the form,

$$w(p) = \frac{1}{\sigma(p)} \tag{4.6}$$

where σ denotes the standard deviation of the p-th cepstral coefficient.

- The sinusoidal lifter can be expressed as,

$$w(p) = 1 + \frac{P}{2}\sin(\frac{\pi p}{P}) \tag{4.7}$$

- The exponential lifter is given by,

$$w(p) = p^s \exp(-\frac{p^s}{2\tau^2}) \tag{4.8}$$

where s and τ are constants with typical values of 1.5 and 5, respectively.

In this thesis, we use the sinusoidal lifter, while the other key parameters are chosen as follows:

- Number of mel-scale filterbank channels: $M = 64$
- Frame size for FFT calculation: $K = 1024$
- Number of MFCC coefficients: $P = 22$
- Sampling frequency: $f_s = 16$ KHz

Further details will be provided in the following Chapter where we present the experimental results.

4.3 Incorporation of MFCC within DNN

In the following subsections, we discuss the training and the enhancement procedures of the proposed MFCC-based DNN system.

4.3.1 Training

The proposed low complexity DNN model is illustrated in Fig. 4.3. While the training procedure is the same as that explained in Subsection 3.3, the DNN structure is less complex. Each hidden layer, labeled with index $l \in \{1, 2, \ldots, L-1\}$ consists of $I^{'}$ neurons, where L is the total number of layers. As we will show in Chapter 5, the use of MFCC will make it possible to significantly reduce the number of neurons in each layer. That is, $I^{'} = I/\beta$,

where I is the number of neurons per layer in an STFT-based DNN system with similar performance, and $\beta > 1$ is a complexity reduction factor.

In the training phase, the network is presented with the sequence clean speech target vectors $\boldsymbol{\mathcal{X}}_\nu$, the noise target vectors $\boldsymbol{\mathcal{D}}_\nu$, and the mel scale cepstral coefficient vectors, calculated from the input signal by proceeding as in Subsection 4.2. For convenience, we define the input MFCC matrix $\boldsymbol{\mathcal{C}} = [\boldsymbol{\mathcal{C}}_1, \ldots, \boldsymbol{\mathcal{C}}_N]$, the clean speech target matrix $\boldsymbol{\mathcal{X}} = [\boldsymbol{\mathcal{X}}_1, \ldots, \boldsymbol{\mathcal{X}}_N]$, and the noise target matrix $\boldsymbol{\mathcal{D}} = [\boldsymbol{\mathcal{D}}_1, \ldots, \boldsymbol{\mathcal{D}}_N]$, where N is the total number of frames available for training.

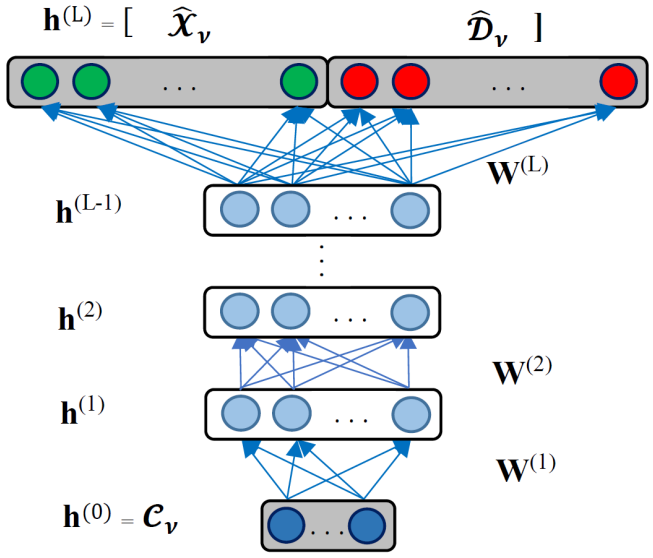

Fig. 4.3 The proposed MFCC-based DNN model

4.3.2 Enhancement

In the enhancement stage, at the ν-th frame, the network is presented with the noisy input MFCC vector $\boldsymbol{\mathcal{C}}_\nu$, in order to predict the clean speech and the noise spectrum magnitudes at the output, represented as $\hat{\boldsymbol{\mathcal{X}}}_\nu$ and $\hat{\boldsymbol{\mathcal{D}}}_\nu$, respectively. The output layer, $\mathbf{h}^{(L)} = [\hat{\boldsymbol{\mathcal{X}}}_\nu, \hat{\boldsymbol{\mathcal{D}}}_\nu]$ consists of two K-dimensional prediction components. In this notation, $\hat{\boldsymbol{\mathcal{X}}}_\nu = [\hat{\mathcal{X}}_{\nu,0}, \hat{\mathcal{X}}_{\nu,1}, \ldots, \hat{\mathcal{X}}_{\nu,K-1}]$, and $\hat{\boldsymbol{\mathcal{D}}}_\nu = [\hat{\mathcal{D}}_{\nu,0}, \hat{\mathcal{D}}v_{\nu,1}, \ldots, \hat{\mathcal{D}}_{\nu,K-1}]$, where the components $\hat{\mathcal{X}}_{\nu,k}$ and $\hat{\mathcal{D}}_{\nu,k}$ provide preliminary estimates of the clean speech and noise spectrum magnitudes, that is $\mathcal{X}_{\nu,k} \equiv |X(\nu,k)|$ and $\mathcal{D}_{\nu,k} \equiv |D(\nu,k)|$, respectively. After DNN processing, the predicted

magnitude spectrum of the clean speech for the ν-th frame is derived by applying a Wiener filter, as explained in Section 3.2.

4.4 Non-negative Matrix Factorization Approach

In addition to DNN, there has been much interest in the application of the non-negative matrix factorization (NMF) techniques to many problems including, but not limited to, speech enhancement [19], source separation [25], speech or object recognition [26] and image processing [61]. The NMF algorithm can be considered as a dimensionality reduction tool, which represents a given data matrix by a product of a basis and activation matrices with non-negative elements.

Let us introduce a non-negative matrix [1] $\mathbf{V} = [V_{pq}] \in \mathbb{R}_+^{P \times Q}$, where $\mathbb{R}_+$ refers to the set of non-negative real numbers. Using the NMF technique, the matrix $\mathbf{V}$ can be approximated as $\mathbf{V} \cong \mathbf{WH}$, where $\mathbf{W} = [W_{pr}] \in \mathbb{R}_+^{P \times R}$ is a basis matrix (also known as dictionary or codebook) and $\mathbf{H} = [H_{rq}] \in \mathbb{R}_+^{R \times Q}$ is an activation matrix (also known as encoding matrix), and R is the number of basis vectors.

In order to achieve this approximation, a suitable cost function $\mathcal{J}(\mathbf{V}, \mathbf{WH})$ needs to be minimized, subject to the non-negative constraints $\mathbf{W} \geqslant 0$, $\mathbf{H} \geqslant 0$, according to the procedures in [25, 62, 63]. To achieve a solution to this optimization problem, one can express the gradient of $\mathcal{J}(\mathbf{V}, \mathbf{WH})$ in terms of two non-negative terms $\nabla^- \mathcal{J}(\mathbf{V}, \mathbf{WH})$ and $\nabla^+ \mathcal{J}(\mathbf{V}, \mathbf{WH})$, such that $\mathcal{J}(\mathbf{V}, \mathbf{WH}) = \nabla^+ \mathcal{J}(\mathbf{V}, \mathbf{WH}) - \nabla^- \mathcal{J}(\mathbf{V}, \mathbf{WH})$, [25, 62, 63]. Using these terms, it is possible to express the multiplicative update rule as:

$$\mathbf{W} \leftarrow \mathbf{W} \otimes \frac{\nabla_W^- \mathcal{J}(\mathbf{V}, \mathbf{WH})}{\nabla_W^+ \mathcal{J}(\mathbf{V}, \mathbf{WH})}, \qquad \mathbf{H} \leftarrow \mathbf{H} \otimes \frac{\nabla_H^- \mathcal{J}(\mathbf{V}, \mathbf{WH})}{\nabla_H^+ \mathcal{J}(\mathbf{V}, \mathbf{WH})} \tag{4.9}$$

where the operator $\otimes$ and the quotient line respectively denote element-wise multiplication and division. Under appropriate conditions, the sequence of matrices $\mathbf{W}$ and $\mathbf{H}$ obtained through repeated application of the update rule 4.9 usually converge to a local minimum of the cost function $\mathcal{J}(\mathbf{V}, \mathbf{WH})$.

The cost function, $\mathcal{J}(\mathbf{V}, \mathbf{WH})$, can be obtained using some measure of distance between two non-negative matrices. Several cost functions have been proposed in the literature, such

[1] We refer to a matrix $\mathbf{V} = [V_{pq}]$ as non-negative if all its entries satisfies $V_{pq} \geqslant 0$, and indicates this through the notation $\mathbf{V} \geqslant 0$.

as the Euclidean distance (FR), the Kullback-Leibler divergence (KL), and the Itakura-Saito divergence (IS) shown in (4.10), (4.11), (4.12), respectively:

$$\mathcal{J}_{\mathrm{FR}}(\mathbf{V},\mathbf{WH}) = \sum_{p=1}^{P}\sum_{q=1}^{Q}(V_{pq} - [\mathbf{WH}]_{pq})^2 \tag{4.10}$$

$$\mathcal{J}_{\mathrm{KL}}(\mathbf{V},\mathbf{WH}) = \sum_{p=1}^{P}\sum_{q=1}^{Q}(V_{pq}\ln\frac{V_{pq}}{[\mathbf{WH}]_{pq}} - V_{pq} + [\mathbf{WH}]_{pq}) \tag{4.11}$$

$$\mathcal{J}_{\mathrm{IS}}(\mathbf{V},\mathbf{WH}) = \sum_{p=1}^{P}\sum_{q=1}^{Q}(\frac{V_{pq}}{[\mathbf{WH}]_{pq}} + \ln\frac{V_{pq}}{[\mathbf{WH}]_{pq}} - 1) \tag{4.12}$$

However, the KL-divergence is widely used and it has been shown that it improves the performance compared to the other functions. Hence, by incorporating the KL-divergence cost function into the NMF, the update rule becomes the following:

$$\mathbf{W} \leftarrow \mathbf{W} \otimes \frac{(\mathbf{V}/\mathbf{WH})\mathbf{H})^T}{\mathbf{1}\mathbf{H}^T}, \qquad \mathbf{H} \leftarrow \mathbf{H} \otimes \frac{\mathbf{W}^T(\mathbf{V}/\mathbf{WH})}{\mathbf{W}^T\mathbf{1}} \tag{4.13}$$

Here, T denotes the matrix transpose operation and $\mathbf{1}$ is a $P \times Q$ dimensional matrix with all entries equal to 1.

As previously mentioned, in single channel speech enhancement, the noisy speech spectrum can be expressed as the sum of clean and noise magnitude spectrum, i.e, $Y(p,q) = X(p,q) + D(p,q)$. Here, we define $\mathbf{V} = [V_{pq}] \in \mathbb{R}_+^{P\times Q}$, where V_{pq} denotes the magnitude of the STFT coefficient of the noisy speech, i.e. $V_{pq} = |Y(p,q)|$, for the p-th frequency bin of the q-th time frame, Q is the number of time frames and P refers to the number of frequency bins. In the following explanation, for convenience in notation, we use subscripts or superscripts Y, X, and D referring to the noisy speech, the clean speech, and noise, respectively.

NMF consists of two stages of training and enhancement. In the training stage, the basis matrix for both the clean and noise, respectively denoted as $\mathbf{W}_X = [W_{pr}^X] \in \mathbb{R}_+^{P\times R_X}$ and $\mathbf{W}_D = [W_{pr}^D] \in \mathbb{R}_+^{P\times R_D}$ are derived by applying the update rule to the training data, $\mathbf{V}_X \in \mathbb{R}_+^{P\times Q_X}$, $\mathbf{V}_D \in \mathbb{R}_+^{P\times Q_D}$. In the enhancement stage, we first form the basis matrices as $\mathbf{W}_Y = [\mathbf{W}_X\mathbf{W}_D] \in \mathbb{R}_+^{P\times(R_X+R_D)}$. Then, the NMF activation function is applied to the noisy speech magnitude spectrum $\mathbf{V}_Y \in \mathbb{R}_+^{P\times Q_Y}$ to estimate the activation matrix of the noisy speech, $\hat{\mathbf{H}}_Y = [\hat{\mathbf{H}}_X^T\hat{\mathbf{H}}_D^T]^T \in \mathbb{R}_+^{(Q_X+Q_D)\times R}$.

In this regard, the activation function is first computed, then the estimated clean speech spectrum is evaluated using Wiener filter as,

$$\hat{\mathbf{S}} = \frac{\mathbf{P}_X}{\mathbf{P}_X + \mathbf{P}_D} \otimes \mathbf{Y} \tag{4.14}$$

where matrices $\mathbf{P}_X = [\hat{P}_X(p,q)]$ and $\mathbf{P}_D = [\hat{P}_D(p,q)]$ contain the estimated values of speech and noise power spectra, with (p,q) being the time-frequency bin. These estimates are computed recursively as,

$$\hat{P}_X(p,q) = \tau_x \hat{P}_X(p,q-1) + (1-\tau_x)([\mathbf{W}_X \hat{\mathbf{H}}_X]_{pq})^2 \tag{4.15}$$

$$\hat{P}_D(p,q) = \tau_d \hat{P}_D(p,q-1) + (1-\tau_d)([\mathbf{W}_D \hat{\mathbf{H}}_D]_{pq})^2 \tag{4.16}$$

In this expression, the parameters τ_x and τ_d, taken in the range $[0,1)$, denote the temporal smoothing factors for the speech and noise signal, respectively. Finally, to reconstruct the enhanced speech in the time domain, the inverse STFT is applied to $\hat{\mathbf{S}}$ followed by the overlap method as explained in Chapter 3.

4.5 Complexity Analysis

The computational complexity of an algorithm indirectly provides a measure of the time taken by that algorithm to run on a given processing platform. It is often measured in terms of the number of computer instructions or operation cycles (e.g., floating point multiplications) needed to execute the algorithm [64]. Here, we measure the complexity in terms of the required number of real multiplications per frame.

The overall computational complexity of the proposed MFCC-based DNN for speech enhancement depends on the implementation of the individual sub-algorithms composing the system. Let us first consider the MFCC acoustic feature extraction module represented in Fig. 4.1. The pre-emphasis and windowing require two multiplications per sample, or $2K$ multiplications per frame, where K is the frame length. The STFT can be implemented using FFT with $K \log_2 K$ complexity, as opposed to a direct realization of the DFT with complexity K^2. The required STFT magnitude coefficients for each frame are then computed at the cost of $2K$. These values are used as input to the mel-scale triangular filterbank with complexity upper bounded by MK, where M is the total number of overlapping tri-

angular pass-band filters. The DCT in the mel-scale cepstral analysis stage is implemented using the fast cosine transform (FCT) algorithm with complexity $M \log_2 M$. Hence, in the acoustic feature extraction module, a total of $K \log_2 K + (M+4)K + M \log_2 M$ multiplications per frame of speech signal are required, where the dominant term is $K \log_2 K$ (since $K \gg M$ in our application).

Now consider the MFCC-based DNN as shown in Fig. 4.3 with L hidden layers, each containing I' neurons for simplicity and I_L output neurons, and assume that the maximum number of training iterations is T. Based on [65], the MFCC-based DNN requires $3(L-2)I'^2T + 3I'I_LT + 2PI'T + 2I_LT$ multiplications per frame in the training stage, where P is the number of computed MFCC. In the enhancement stage, the MFCC-based DNN requires $(L-2)I'^2 + I'I_L + PI'$ multiplications to forward propagate the input frame to the output.

Referring to Fig. 4.1, the signal reconstruction involves the following operations: updating the speech and noise PSD based on (3.10)-(3.11) at the cost of $6K$ per frame; implementing the Wiener filter in (3.9) at the cost $4K$ per frame (this includes incorporation of the noisy speech phase); and signal reconstruction via inverse STFT, which requires $K \log_2 K$ multiplications per frame.

Based upon the above considerations, the overall computational complexity of the proposed MFCC-based DNN system is summarized in Table 4.1.

Table 4.1 Computational complexity of the MFCC-based DNN for speech enhancement

MFCC feature extraction	$K \log_2 K + (M+4)K + M \log_2 M$
DNN Training	$(3(L-2)I'^2 + 3I'I_LT + 2PI' + 2I_L)T$
DNN Enhancement	$(L-2)I'^2 + I'I_L + PI'$
Signal reconstruction	$10K + K \log_2 K$

Now let us consider the STFT-based DNN which is shown in Fig. 3.2. In this case, since the algorithm uses the STFT magnitude coefficients as inputs, the complexity of the feature extraction reduces to $K \log_2 K + 4K$. Denoting by I the number of neurons in each one of the L layers, the STFT-based DNN requires $3(L-2)I^2T + 3II_LT + 2KIT + 2I_LT$ multiplications per frame in the training stage, including the forward and back propagation needed to apply the gradient information to lower layers and update the weights accordingly.

In the enhancement stage, the algorithm requires $(L-2)I^2 + KI + II_L$ multiplications to forward propagate the input frame to the output. The signal reconstruction is identical to the MFCC-based DNN algorithm. Subsequently, the reduced computational workload of the low complexity DNN, that is $I' < I$ and $P < K$, will allow for a faster enhancing and testing running time, as illustrated in the next chapter.

Computational complexity is often used as a performance measure of an algorithm along with its memory requirements. The number representation used for implementing the DNN algorithms for speech enhancement developed and studied in this thesis is the standard double precision floating point numbers in Python, which is based on IEEE 754 floating point representation [66]. A single number represented using this standard takes up $S = 8$ bytes of memory. In the case of the STFT-based DNN with parameters K, L, I and I_L as defined before, the memory requirement to store the weights and biases needed to perform a forward pass in the DNN is approximately $S(KI + I^2 + II_L)$. [2] In the same way, the memory requirement of the proposed MFCC-based DNN for processing a single frame of noisy speech is approximately $S(PI' + I'^2 + I'I_L)$. Again, due to the fact that $I' < I$ and $P < K$, the memory needed by the MFCC-based DNN algorithm will be much smaller than that needed by the STFT one.

[2] This excludes the additional memory needed to store temporary results from each hidden units.

Chapter 5

Simulation Results and Discussion

In this chapter, we first describe the methodology used to assess the performance of the proposed MFCC-based DNN, as well as the standard STFT-based and NMF approaches for speech enhancement. We then evaluate the run time of these algorithms for the training and the enhancement phases. Finally, we compare their performance in the speech enhancement task by presenting the PESQ, segSNR, and SDR results.

5.1 Methodology

In this section, we describe the speech data corpus used to carry out our experiments, the parameter settings of the three algorithms under evaluation, and the performance measures used for their comparative evaluation.

5.1.1 The noisy speech data

The clean speech signals used in our experiments were selected from the TSP-speech database [67] and consisted of 1500 utterances from 25 different male and female speakers (60 utterances per speaker). As for the noise signals, five different types were selected from the NoiseX92 database [2], namely: babble, pink, buccaneer2, factory1, and hfchannel. The noisy speech utterances were generated by adding noise sequences to the clean speech, appropriately scaled to achieve input SNRs of 0, 5 and 10 dB. The sampling frequency of all the signals was set to its original value of 16 KHz.

For each SNR level, the noisy speech utterances generated in this way were divided

into two sets. The first set, referred to as the training and validation set, includes 18750 utterances, corresponding to 11 hours of speech, while the second set referred to as the test set, includes 3750 utterances, corresponding to 2 hours.

5.1.2 Systems Under Comparison

We implemented the proposed MFCC-based DNN for speech enhancement along with two benchmark algorithms, namely the STFT-based DNN and NMF algorithms, to compare the enhanced speech quality. The basic settings for the STFT analysis and synthesis were kept identical for all the benchmark and proposed methods. Specifically, a value of $\alpha = 0.97$ was used in the pre-emphasis filter while a Hanning window was employed in computing the STFT. The length of the window was set to $K = 1024$ (64ms) with a 75% frame overlap for both the analysis and the synthesis. Also the temporal smoothing factors for the speech and noise spectrum estimation, as needed for the final Wiener filtering operation in all three approaches, were set to $\tau_x = 0.4$ and $\tau_d = 0.9$, respectively. The same dataset as mentioned in Subsection 5.1.1 was applied to train all three methods, and this for each noise type and SNR level.

For the implementation of the MFCC-based DNN system, we used $M = 64$ ovelapping filterbank channels spanning the frequency range from 300 to 3700Hz. The filter outputs were used to compute $P = 22$ MFCC coefficients using an M-point DCT, after which a sinusoidal lifter was applied. The first MFCC coefficient, $\mathcal{C}_\nu(0)$, provides a measure of the energy content of each frame. The proposed MFCC-based DNN consists of $L = 2$ hidden layers with $I' = 1024$ hidden neurons each. For each processed data frame, the DNN is input with the corresponding vector of $P = 22$ MFCC, while the output consists of two vectors containing the $K/2 + 1 = 513$ magnitude coefficients [1] of the clean speech and noise signals, respectively. For each noise type and SNR value, the MFCC-based DNN was trained by processing the data in the training/validation set, consisting more precisely of the corresponding noisy speech, clean speech and noise sequences. The optimum number of iterations for training of the DNN was obtained by observing the behavior of the training and validation errors and selecting the value where the validation error starts to increase. In this way, we empirically found that a number of 25 iterations was adequate. We considered the iRprop$^-$ algorithm with parameters set as follows: $\eta^+ = 1.2$, $\eta^- = 0.8$, $\Delta_{max} = 100$,

[1] For real valued signal, only half of the STFT magnitude coefficients are needed to the even symmetry property.

$\Delta_{min} = 0$, $\Delta_{ij}^{(l)}(0) = 0.5$ and the Ridge regularization parameter $\lambda = 0.01$.

The STFT-based DNN consists of $L = 2$ hidden layers with $I = 4096$ hidden neurons each. In our experiment, for each processed data frame, the STFT-based DNN is input with the corresponding vector of $K/2 + 1 = 513$ STFT, while the output consists of two vectors containing the $K/2 + 1 = 513$ magnitude coefficients of the clean speech and noise signals, respectively. We used a regularization parameter of $\lambda = 0.01$.

Regarding the NMF algorithm, we consider a speaker-independent but noise-dependent application, in which one basis matrix is trained for all the clean speech signals and one basis matrix is trained for each type of noise signals. For implementing the NMF, we considered the KL-divergence leading to the update rule (4.13). The number of basis vectors was set to $R = 80$.

The NMF algorithm was implemented in Matlab [68] while both DNN methods were implemented in Matlab and Python. The numerical experiments were run on a computer featuring the Intel(R) Xeon(R) central processing unit (CPU), 2 additional parallel processors operating at the speed of 2.3 GHz, and 64GB of RAM.

5.1.3 Performance Measures

To evaluate the enhanced speech, we use three objective measures, namely, the perceptual evaluation of speech quality (PESQ) [69, 70], the segmental signal-to-noise ratio (SegSNR) [71] and the source-to-distortion ratio (SDR) [72]. A brief overview of these evaluation measures is presented below.

Perceptual Evaluation of Speech Quality

In order to evaluate the quality of the enhanced signal, the perceptual evaluation of speech quality (PESQ) [69, 70] was used. It is a popular measure with typical value in the range from -0.5 to 4.5, used for automated assessment of speech quality as experienced by a listener. PESQ attempts to emulate the results of a subjective listening experiment by predicting the quality of the enhanced signal as it would be perceived by a listener. Larger PESQ values correspond to better overall speech quality.

Segmental Signal-to-Noise Ratio

Segmental signal-to-noise ratio (SegSNR) is a common objective quality measure, and is defined as the average of SNR (dB) values calculated over successive short-time segments of speech (e.g., 16-64ms). The SNR helps to distinguish between the estimation errors that are mostly inflicted by noise. Consider a sequence of speech frame, each of length N and indexed by integer i. Let $\mathbf{x}_i = [x(n_i), \ldots, x(n_i + N - 1)]$ denotes the N-dimensional vector of clean speech samples for the i-th frame, where n_i denotes the index of the first sample. Similarly, let $\hat{\mathbf{x}}_i = [\hat{x}(n_i), \ldots, \hat{x}(n_i + N - 1)]$ denote the corresponding vector of processed speech (e.g. as produce by the MFCC-based DNN) for the i-th frame. The SNR of the processed speech signal for the i-th frame can be formulated as,

$$\mathrm{SNR}_i = \frac{\mathbf{x}_i^T \mathbf{x}_i}{(\mathbf{x}_i - \hat{\mathbf{x}}_i)^T (\mathbf{x}_i - \hat{\mathbf{x}}_i)} \tag{5.1}$$

Considering that SNR values tends to be influenced more significantly by intense fragments of speech (i.e. frames with large power), it is preferable to compute the average SNR in the log domain, by averaging the decibel values of SNR_i over multiple frames. The resulting measure, called segmental SNR (segSNR), averages the performance more equally over the weak and strong segments of the speech. The average SegSNR over all the frames is therefore calculated as,

$$\mathrm{SegSNR} = \frac{1}{N_s} \sum_{i=0}^{N_s - 1} \mathrm{SNR}_i \Big|_{\mathrm{dB}} \tag{5.2}$$

$$= \frac{1}{N_s} \sum_{i=0}^{N_s - 1} 10 \log_{10} \frac{\mathbf{x}_i^T \mathbf{x}_i}{(\mathbf{x}_i - \hat{\mathbf{x}}_i)^T (\mathbf{x}_i - \hat{\mathbf{x}}_i)} \tag{5.3}$$

where N_s refers to the number of available speech frames. Higher SegSNR values indicate lesser background noise, hence a better performance of the enhancement method.

Source-to-Distortion Ratio

The source-to-distortion ratio (SDR) measures the amount of speech distortion in the enhanced speech signal. It reflects the overall separation quality on a dB scale, considering both the speech distortion and the noise reduction aspects [73]. In our work, the SDR value

is computed by employing the BSS-Eval toolbox [72].

5.2 Performance Evaluation and Discussion

As explained earlier, the PESQ, SegSNR and SDR measurements are used as the performance metrics in our experiments. Five noise types, i.e., babble, pink, buccaneer2, factory1, and hfchannel noise, are considered for the evaluation of the various algorithms under study. The clean speech signal is corrupted by these background noise types at three SNR levels: 0, 5 and 10dB. In addition to the proposed approach, two other benchmark methods are considered in the experiments. For conciseness, we refer to these methods as follows,

- DNN-MFCC: the proposed DNN approach using MFCC as inputs.
- DNN-STFT: a more conventional DNN approach using the STFT magnitude coefficients as input [22].
- NMF: the speaker independent NMF approach of Section 4.

5.2.1 Parameter Selection and Run Times

The first step in implementing the DNN approaches for speech enhancement was to determine the number and size of the hidden layers. Based upon several preliminary experiments, we found that a number of $L = 2$ hidden layers was adequate to achieve good performance. For this value of L, we have also compared the speech enhancement performance of both MFCC and STFT-based DNN models trained with different sizes of hidden layers, i.e., 1024, 2048, and 4096. In Table 5.1, we show average PESQ values for the two DNN methods versus the size of the hidden layers as obtained for pink noise at 5dB. Based on these and other similar results, we found that the optimum number of hidden layers, resulting in the best performance of the STFT and the MFCC-based DNN methods were $I = 4096$ and $I' = 1024$, thereby 4096 and 1024, respectively. Hence, according to our discussion in Section 4.3, we have $I' = I/\beta$, where $\beta = 4$.

Fig. 5.1 gives the average PESQ results for different numbers P of the MFCC coefficients for pink noise at 5 dB input SNR. Based on these and other similar results, the optimal value of $P = 22$ is chosen for the proposed method.

Table 5.1 Comparing the average PESQ values of different DNN structures for pink noise at 5dB

Hidden layers size	1024	2048	4096
DNN-STFT	2.17	2.18	**2.21**
DNN-MFCC	**2.23**	2.19	2.18

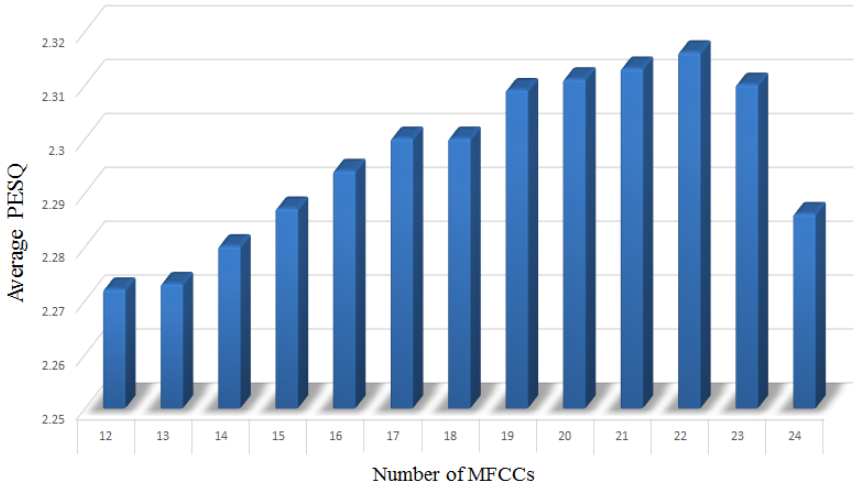

Fig. 5.1 Average PESQ results for different numbers of MFCCs

Table 5.2 demonstrates the running time comparison of different algorithms. In the DNN-STFT approach, we trained a DNN model with 2 hidden layers of size $I = 4096$ neurons each and used the STFT magnitudes as input features. In the DNN-MFCC approach, the MFCC input features were applied to a DNN model with 2 hidden layers of size $I' = 1024$ neurons each. For both STFT and MFCC-based DNN approaches, the training and validation set includes 18750 utterances, corresponding to 11 hours of speech, while the test set includes 3750 utterances, corresponding to approximately 2 hours. In the case of the speaker independent but noise dependent NMF approach, the training and validation set, includes 1250 utterances, corresponding to approximately 42 minutes of speech, while the test set, includes 3750 utterances, corresponding to approximately 2 hours.

From the discussion in Section 4.3, since $I' = I/\beta$ where $\beta = 4$, the proposed DNN structure using the MFCC features as input can lead to a significant reduction in computa-

tional complexity when compared to the DNN model using conventional STFT features [22]. This is evidenced by the results in Table 5.2 where the runtime for the training phase of the DNN-MCCC is reduced by a factor of approximately 5 (4.75) compared to DNN-STFT. While NMF takes less time than both DNN models for training, mainly because in the later case, longer training data sets obtained by adding various noise types at different SNR to the clean speech are employed. Nevertheless, the enhancement phase for NMF takes much longer time than for the DNN models, mainly due to the use of an iterative process in the estimation of the activation matrix. The running time for the enhancement phase for both DNN-STFT and DNN-MFCC are almost the same, as they forward propagate the same data and compute the predicted signal in the same manner.

Table 5.2 Running time including the training and the enhancement stages

Time	NMF	DNN-STFT	DNN-MFCC
Training	5min	38min	8min
Enhancement	15min	2min	2min

5.2.2 Enhancement Performance

The average perceptual evaluation results, as given by the PESQ values for the benchmark and the proposed algorithms for each noise type and SNR levels are presented in Tables 5.3 to 5.7. It can be observed from these results that the proposed DNN-MFCC method achieves the best performance for all the cases under study. Compared to NMF, the use of the DNN-MFCC leads to significant gain in PESQ values, especially at lower SNR (i.e. 0 and 5dB).

Tables 5.8 to 5.12 present the values of the segSNR measurements while Tables 5.13-5.17 present the SDR measurements for the same noise types and SNR levels as mentioned above. It can be observed from these results that the proposed MFCC-based DNN method performs better than the NMF and STFT-based DNN methods in most cases, for the objective performance metrics under consideration.

Informal listening tests tend to concur with the above objective results. In particular, the NMF method seem to suppress the stationary elements present in the background noise but fail to reduce the non-stationary elements properly. This is evident when listening to

the enhanced signals processed by this method. The proposed DNN-MFCC method is able to suppress the non-stationary elements in the background noise better than the NMF and DNN-STFT methods in most cases. However, this seems to come at the expense of some additional distortion in the resulting enhanced speech.

For illustrative purposes, Fig. 5.2 presents the time domain waveforms of the clean, noisy and enhanced speech, as well as the corresponding spectrograms, for a representative sentence from the TSP database [2]. In this case, the noisy is obtained by adding pink noise with 5dB SNR, while the enhanced speech is that obtained with the proposed DNN-MFCC method. It is seen from both the time domain waveforms and the spectrograms that a significant portion of the noise has been removed, while most of the clean speech structure has been preserved.

To summarize our experimental findings, the above objective results show that the proposed DNN approach using MFCC coefficients can lead to a better enhancement performance than the NMF and DNN-STFT. More importantly, when compared to the DNN-STFT, the use of MFCC in the proposed method leads to a significant reduction of the model complexity, which translates into a corresponding reduction of the algorithm run time during the training phase.

Table 5.3 Average PESQ values for pink noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	PESQ	1.31	1.70	2.07	**2.12**
5dB	PESQ	1.68	2.13	2.21	**2.23**
10dB	PESQ	2.08	2.45	2.31	**2.46**

Table 5.4 Average PESQ values for babble noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	PESQ	1.45	1.52	1.81	**1.92**
5dB	PESQ	1.83	1.92	2.20	**2.22**
10dB	PESQ	2.01	2.24	2.33	**2.39**

Table 5.5 Average PESQ values for buccaneer2 noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	PESQ	1.13	1.67	1.90	**2.03**
5dB	PESQ	1.52	2.07	2.08	**2.26**
10dB	PESQ	1.90	2.30	2.21	**2.42**

Table 5.6 Average PESQ values for factory1 noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	PESQ	1.36	1.60	1.89	**1.93**
5dB	PESQ	1.73	2.05	2.20	**2.24**
10dB	PESQ	2.11	2.30	2.33	**2.36**

Table 5.7 Average PESQ values for hfchannel noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	PESQ	1.19	1.62	2.00	**2.09**
5dB	PESQ	1.45	1.80	2.09	**2.17**
10dB	PESQ	1.79	2.22	2.14	**2.24**

Table 5.8 Average segSNR values for pink noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	segSNR	-3.98	0.03	0.05	**0.08**
5dB	segSNR	-1.5	3.05	3.03	**3.06**
10dB	segSNR	2.38	6.23	6.13	**6.25**

Table 5.9 Average segSNR values for babble noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	segSNR	-3.98	**-0.66**	-0.71	-0.68
5dB	segSNR	-1.06	2.22	2.25	**2.26**
10dB	segSNR	3.3	**4.54**	4.48	4.53

Table 5.10 Average segSNR values for buccaneer2 noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	segSNR	-4.2	-0.18	-0.12	**-0.12**
5dB	segSNR	-1.13	2.12	2.18	**2.23**
10dB	segSNR	2.5	**6.7**	6.65	6.68

Table 5.11 Average segSNR values for factory1 noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	segSNR	-4.35	-0.52	-0.55	**-0.51**
5dB	segSNR	-1.33	2.69	2.73	**2.73**
10dB	segSNR	2.56	5.81	5.79	**5.88**

Table 5.12 Average segSNR values for hfchannel noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	segSNR	-4.5	**0.47**	0.35	0.46
5dB	segSNR	-1.5	4.13	4.09	**4.14**
10dB	segSNR	2.4	**6.78**	6.61	6.72

Table 5.13 Average SDR values for pink noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	SDR	0.02	5.16	5.23	**5.36**
5dB	SDR	5.01	10.08	10.31	**10.38**
10dB	SDR	10.00	14.30	14.57	**14.77**

Table 5.14 Average SDR values for babble noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	SDR	0.03	2.42	2.53	**2.60**
5dB	SDR	5.02	7.28	7.43	**7.44**
10dB	SDR	10.02	11.45	11.68	**11.71**

Table 5.15 Average SDR values for buccaneer2 noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	SDR	0.01	3.89	4.03	**4.11**
5dB	SDR	5.01	8.64	8.32	**8.39**
10dB	SDR	10.01	13.07	13.8	**13.83**

Table 5.16 Average SDR values for factory1 noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	SDR	0.01	4.05	4.23	**4.24**
5dB	SDR	5.00	8.84	**8.90**	8.89
10dB	SDR	10.00	13.26	13.29	**13.32**

Table 5.17 Average SDR values for hfchannel noise

SNR	Eval.	Noisy	NMF	DNN-STFT	DNN-MFCC
0dB	SDR	0.01	5.45	5.53	**5.61**
5dB	SDR	5.00	10.01	11.30	**11.41**
10dB	SDR	10.01	14.03	14.12	**14.33**

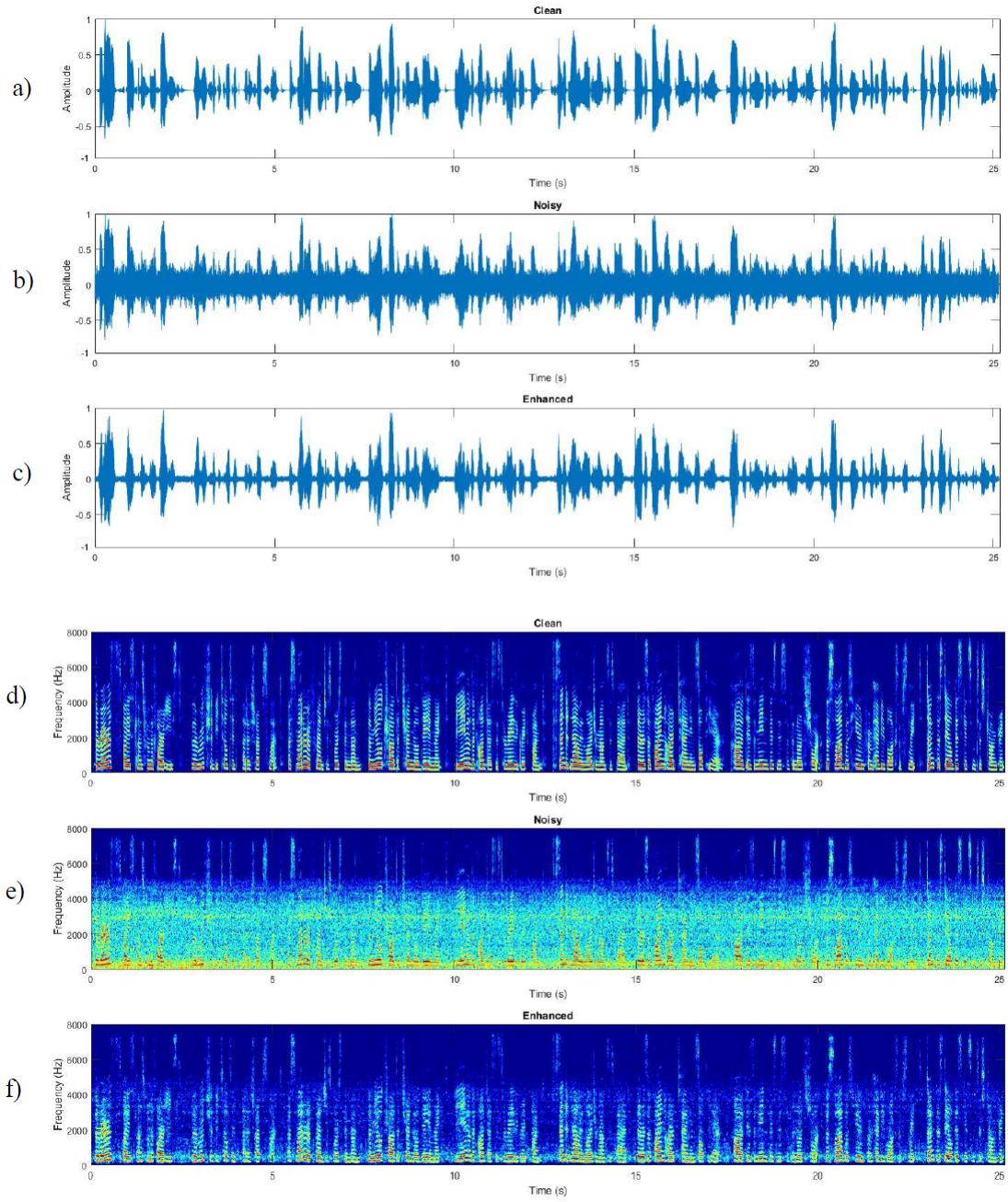

Fig. 5.2 Time domain representations of: a) clean speech signal; b) noisy speech signal (5 dB SNR pink) and c) enhanced speech signal; and frequency domain (spectrogram) representations of d) clean speech signal; e) noisy speech signal and f) enhanced speech signal. The corresponding speech utterances are selected from [2]: "The rarest spice comes from the far East. The roof should be tilted at a sharp slant. A smatter of French is worse than none. The mule trod the treadmill day and night. The aim of the contest is to raise a great fund. To send it now in large amounts is bad. There is a fine hard tang in salty air. Cod is the main business of the north shore. The slab was hewn from heavy blocks of slate. Dunk the stale biscuits into strong drink."

Chapter 6

Conclusion and Future Work

This chapter provides some concluding remarks about the research presented in this thesis. Specifically, Section 6.1 presents a brief summary of the thesis work and contributions, while Section 6.2 lists suggestions for possible future work in this active area.

6.1 Thesis Overview and Contributions

During the last decade, deep neural network (DNN) has been a subject of great interest in many fields of study, such as image processing, speech processing and email classification. While a recently proposed DNN technique based on STFT coefficients [22] offers many advantages for the speech enhancement task, it is characterized by high system complexity and implementation cost, especially with regards to the training phase. The main objective of this thesis was to reduce the DNN complexity by introducing a spectral feature mapping from the noisy mel frequency cepstral coefficients (MFCC) to the enhanced short time Fourier transform (STFT) spectrum. Consequently, by exploiting the lower dimensionality of the MFCC input feature vector, as compared to the STFT feature vector, a low-complexity DNN model is presented, in order to efficiently perform noise suppression for the purpose of a single channel speech enhancement. Below, we present a chapter-wise sequential overview of the main topics discussed in this work.

- In Chapter 1, a comprehensive summary of the speech enhancement problem was presented. This was followed by a literature survey on the conventional and more recent techniques of speech enhancement, including non-negative matrix factorization

(NMF) and DNN methods.

- In Chapter 2, a brief review of deep learning was first presented. Then, motivated by biological networks, artificial neural networks were reviewed. This was followed by the presentation of DNN and backpropagation techniques.

- In Chapter 3, a detailed description of the STFT audio feature extraction and signal reconstruction was given. The STFT based DNN structure presented in [22] along with its training procedure were presented. Towards the end of the chapter, the different Rprop techniques were reviewed.

- In Chapter 4, the proposed DNN algorithm for speech enhancement was presented. This framework delves upon a special spectral feature mapping from MFCC to STFT features. A brief review of the NMF approach was also provided for the purpose of comparison.

- In Chapter 5, simulation results based on three well-accepted objective measures, that is, SegSNR, PESQ and SDR, were presented and discussed. The results showed that compared to STFT-based DNN model, the proposed MFCC-based DNN model offers improved quality in the enhanced speech, while significantly reducing the processing complexity, i.e. run time for the training phase reduced by a factor of ~ 5.

In our work, we considered the regression based feed-forward DNN with MFCC as input vector for the problem of single channel speech enhancement. The DNN model was trained with the help of the improved resilient back propagation algorithm and the MMSE cost function. We implemented the proposed DNN model with different numbers of MFCC coefficients and sizes of network structure and were able to achieve a significant reduction in runtime by a factor of 4.75 when compared to a recently proposed STFT-based DNN approach. The system performance was evaluated using the PESQ, segSNR and SDR scores, thereby showing that the proposed scheme can outperform the NMF and DNN-STFT which were used as benchmark approaches.

6.2 Future Research Directions

In this section, we point out some possible directions for future research work.

There are other families of deep learning algorithms which could be generally applied to the problem of noise reduction or speech enhancement. In this work, we have introduced a low-complexity feed forward DNN which was able to achieve a significant reduction in the computational complexity and run time by exploiting vectors of MFCC features with reduced dimensionality as inputs. It is possible to explore other families of deep learning using the same low-complexity approach. Specifically, the proposed method can be extended to convolutional neural network (CNN) [44] or recurrent neural network (RNN), i.e., long short-term memory (LSTM) [38].

In addition, as mentioned in Section 4.2, there are different types of lifter functions that can be applied to the calculation of MFCC features. One interesting approach would be to generate MFCC features using different types of lifters (or even without) and compare the results to find the best candidate according to their performance.

In this work, we have proposed a MFCC-based DNN and compared its performance with a STFT-based DNN. However, it is possible to combine these two networks with a third, i.e. higher level neural network, to be trained based upon the output of the STFT-based DNN and the MFCC-based DNN, with the aim to further enhance the speech quality. Such a system is expected to improve the enhancement performance, since the third neural network can further learn the complex relationship between the noisy input features and the desired speech.

References

[1] M. Mobarhan, "Creating a 3d neuron scene in blender," CINPLA , Uinoversity of Oslo, Tech. Rep., Sep. 2015.

[2] P. Kabal, "Tsp speech database," *McGill University, Database Version*, vol. 1, no. 0, pp. 09–02, 2002.

[3] J. Benesty, M. M. Sondhi, and Y. Huang, *Speech Enhancement*. Berlin, Germany: Springer Science and Business Media, 2005.

[4] R. Balan and J. Rosca, "Microphone array speech enhancement by Bayesian estimation of spectral amplitude and phase," in *Proc. Sensor Array and Multichannel Signal Process. Workshop*, Aug. 2002, pp. 209–213.

[5] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 27, no. 2, pp. 113–120, Apr. 1979.

[6] N. Virag, "Single channel speech enhancement based on masking properties of the human auditory system," *IEEE Trans. Speech Audio Process.*, vol. 7, no. 2, pp. 126–137, Mar. 1999.

[7] J. S. Lim and A. V. Oppenheim, "Enhancement and bandwidth compression of noisy speech," vol. 67, no. 12, Dec. 1979, pp. 1586–1604.

[8] P. Scalart and J. V. Filho, "Speech enhancement based on a priori signal to noise estimation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 2, May 1996, pp. 629–632.

[9] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 6, pp. 1109–1121, Dec. 1984.

[10] E. Plourde and B. Champagne, "Auditory-based spectral amplitude estimators for speech enhancement," *IEEE Trans. Audio, Speech, Language Process.*, vol. 16, no. 8, pp. 1614–1623, Nov. 2008.

[11] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 2, pp. 443–445, Apr. 1985.

[12] P. C. Loizou, "Speech enhancement based on perceptually motivated bayesian estimators of the magnitude spectrum," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 5, pp. 857–869, Sept 2005.

[13] K. Paliwal and A. Basu, "A speech enhancement method based on Kalman filtering," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 12, Apr. 1987, pp. 177–180.

[14] R. Ishaq, B. G. Zapirain, M. Shahid, and B. Lövström, "Subband modulator Kalman filtering for single channel speech enhancement," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2013, pp. 7442–7446.

[15] Y. Ephraim and H. L. V. Trees, "A signal subspace approach for speech enhancement," *IEEE Trans. Speech Audio Process.*, vol. 3, no. 4, pp. 251–266, Jul. 1995.

[16] K. Hermus and P. Wambacq, "A review of signal subspace speech enhancement and its application to noise robust speech recognition," *EURASIP J. Adv. Signal Process.*, vol. 2007, no. 1, pp. 1–15, 2006.

[17] J. F. and B. Champagne, "Incorporating the human hearing properties in the signal subspace approach for speech enhancement," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 11, no. 6, pp. 700–708, Nov. 2003.

[18] A. Chaudhari and S. B. Dhonde, "A review on speech enhancement techniques," in *Proc. Int. Conf. Pervasive Computing*, Jan. 2015, pp. 1–3.

[19] N. Mohammadiha, P. Smaragdis, and L. Arne, "Supervised and unsupervised speech enhancement using nmf," *IEEE Trans. Audio, Speech, Language Process.*, vol. 21, no. 10, pp. 2140–2151, 2013.

[20] H. Chung, E. Plourde, and B. Champagne, "Discriminative training of NMF model based on class probabilities for speech enhancement," *IEEE Signal Process. Lett.*, vol. 23, no. 4, pp. 502–506, Apr. 2016.

[21] G. Hinton, L. Deng, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[22] D. Liu, P. Smaragdis, and K. Minje, "Experiments on deep learning for speech denoising," in *Interspeech*, Singapore, 2014, pp. 2685–2689.

[23] S. Tamura, "An analysis of a noise reduction neural network," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 3, May 1989, pp. 2001–2004.

[24] J. Rigelsford, "Handbook of neural networks for speech processing," *Sensor Review*, vol. 23, no. 4, Dec. 2003.

[25] T. Virtanen, "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria," *IEEE Trans. on Audio, Speech, and Language Process.*, vol. 15, no. 3, pp. 1066–1074, Mar. 2007.

[26] B. Schuller, F. Weninger, M. Wöllmer, Y. Sun, and G. Rigoll, "Non-negative matrix factorization as noise-robust feature extractor for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 2010, pp. 4562–4565.

[27] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the itakura-saito divergence: With application to music analysis," *Neural Comput.*, vol. 21, no. 3, pp. 793–830, Mar. 2009. [Online]. Available: http://dx.doi.org/10.1162/neco.2008.04-08-771

[28] G. J. Mysore and P. Smaragdis, "A non-negative approach to semi-supervised separation of speech from noise with the use of temporal dynamics," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2011, pp. 17–20.

[29] Y. Xu, J. Du, L. R. Dai, and C. H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 23, no. 1, pp. 7–19, Jan. 2015.

[30] L. Ugang, T. Yu, M. Shigeki, and H. Chiori, "Speech enhancement based on deep denoising autoencoder," in *Proc. INTERSPEECH Conf.*, 2013.

[31] W. Chan and I. Lane, "Deep recurrent neural networks for acoustic modelling," *arXiv preprint arXiv:1504.01482*, Apr. 2015.

[32] A. Narayanan and D. Wang, "Investigation of speech separation as a front-end for noise robust speech recognition," *IEEE Trans. Audio, Speech, Language Process.*, vol. 22, no. 4, pp. 826–835, Apr. 2014.

[33] Y. Tachioka, S. Watanabe *et al.*, "Sequence discriminative training for low-rank deep neural networks," in *Proc. IEEE Global Conf. Signal Info. Process.*, Dec. 2014, pp. 572–576.

[34] T. G. Kang, K. Kwon *et al.*, "Nmf-based target source separation using deep neural network," *IEEE Signal Process. Letters*, vol. 22, no. 2, pp. 229–233, Feb. 2015.

[35] J. Du, Q. Wang, T. G., X. Yong, D. Li-Rong, and L. Chin-Hui, "Robust speech recognition with speech enhanced deep neural networks." in *Proc. INTERSPEECH Conf.*, 2014, pp. 616–620.

[36] P. S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2014, pp. 1562–1566.

[37] A. Maas, Q. V. Le, M. Tyler, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent neural networks for noise reduction in robust ASR," in *INTERSPEECH*, 2012.

[38] F. Weninger, H. Erdogan, S. Watanabe, E. Vincent, J. Le Roux, J. R. Hershey, and B. Schuller, "Speech enhancement with lstm recurrent neural networks and its application to noise-robust asr," in *Proc. Int. Conf. Latent Variable Analysis Signal Separation*. Springer, Aug. 2015, pp. 91–99.

[39] S. Shin, K. Hwang, and W. Sung, "Fixed-point performance analysis of recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Shanghai, China, Mar. 2016, pp. 976–980.

[40] D. O'Shaughnessy, *Speech Communications: Human and Machine*. Wiley-IEEE Press, 1999.

[41] C. Igel and H. Michael, "Improving the Rprop learning algorithm," in *Proc. 2nd Int. Symp. Neural Computation*. Citeseer, 2000, pp. 115–121.

[42] T. M. Mitchell, *The discipline of machine learning*, 2006, vol. 9.

[43] G. A. Carpenter, S. Grossberg, and D. Rosen, "Art 2-a: an adaptive resonance algorithm for rapid category learning and recognition," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. ii, Jul. 1991, pp. 151–156.

[44] S. Haykin, *Neural networks and learning machines*. Pearson Upper Saddle River, NJ, USA:, 2009, vol. 3.

[45] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[46] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized mlp architectures of neural networks," *Int. J. Artificial Intel. Expert Syst.*, vol. 1, no. 4, pp. 111–122, 2011.

[47] M. A. Nielsen, "Neural networks and deep learning," 2015.

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

[49] J. Allen, "Short term spectral analysis, synthesis, and modification by discrete fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 3, pp. 235–238, Jun. 1977.

[50] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 2, pp. 236–243, Apr. 1984.

[51] A. V. Oppenheim, *Discrete-time signal processing*. Pearson Education India, 1999.

[52] T. F. Quatieri, *Discrete-time speech signal processing: principles and practice*, pearson ed. ed. Pearson Education India, Nov. 2006.

[53] E. M. Grais, M. U. Sen, , and H. Erdogan, "Deep neural networks for single channel source separation," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2014, pp. 3734–3738.

[54] E. Dumitru, B. Yoshua, C. Aaron, M. Pierre-Antoine, V. Pascal, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learning Research*, vol. 11, no. Feb., pp. 625–660, 2010.

[55] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the Rpop algorithm," in *Proc. IEEE Int. Conf. Neural Networks*, vol. 1, 1993, pp. 586–591.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[57] C. Igel and M. Hüsken, "Empirical evaluation of the improved Rprop learning algorithms," *Neurocomputing*, vol. 50, pp. 105–123, 2003.

[58] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, and D. Povey, "The htk book," *Cambridge university engineering department*, vol. 3, p. 175, 2002.

[59] X. Huang, A. Acero, and H. W. Hon, *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[60] K. K. Paliwal, "Decorrelated and liftered filter-bank energies for robust speech recognition," in *Sixth European Conference on Speech Communication and Technology*, 1999.

[61] S. Zafeiriou, A. Tefas, I. Buciu, and I. Pitas, "Exploiting discriminant information in nonnegative matrix factorization with application to frontal face verification," *IEEE Transactions on Neural Networks*, vol. 17, no. 3, pp. 683–695, 2006.

[62] E. M. Grais and H. Erdogan, "Regularized nonnegative matrix factorization using gaussian mixture priors for supervised single channel source separation," *Computer Speech & Language*, vol. 27, no. 3, pp. 746–762, 2013.

[63] N. Bertin, R. Badeau, and E. Vincent, "Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 538–549, 2010.

[64] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[65] P. Vincent, A. de Brébisson, and X. Bouthillier, "Efficient exact gradient update for training deep networks with very large sparse targets," in *Advances in Neural Information Processing Systems*, 2015, pp. 1108–1116.

[66] M. Lutz, *Learning Python*, 2nd ed. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2003.

[67] R. University, "Signal processing information base: noise data," Nov. 1993. [Online]. Available: Available online: http://spib.rice.edu/spib/select-noise.html

[68] T. P. Krauss, L. Shure, and J. N. Little, "Signal processing toolbox for use with matlab," 1994.

[69] A. Rix, J. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (pesq), an objective method for end-to-end speech quality assessment of narrowband telephone networks and speech codecs," *ITU-T Recommendation*, p. 862, Feb. 2001.

[70] P. C. Loizou, *Speech enhancement: theory and practice*. Boca Raton, FL.: CRC press, Feb. 2013.

[71] Voicebox, "Speech processing toolbox for MATLAB." [Online]. Available: Available online: http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html

[72] C. Févotte, R. Gribonval, and E. Vincent, "Bss eval, a toolbox for performance measurement in (blind) source separation," 2010. [Online]. Available: Available online: http://bass- db.gforge.inria.fr/bss eval

[73] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, Jul. 2006.