Tampere University

Farhad Javanmardi

# GENERATING SPEECH IN DIFFERENT SPEAKING STYLES USING WAVENET

Master of Science Thesis
Faculty of Information Technology and Communication Sciences (ITC)
Examiners: Prof. Paavo Alku
Prof. Okko Räsänen
May 2020

# ABSTRACT

Farhad Javanmardi: Generating speech in different speaking styles using WaveNet
Master of Science Thesis
Tampere University
Audio-Visual Signal Processing
May 2020

Generating speech in different styles from any given style is a challenging research problem in speech technology. This topic has many applications, for example, in assistive devices and in human-computer speech interaction. With the recent development in neural networks, speech generation has achieved a great level of naturalness and flexibility. The WaveNet model, one of the main drivers in recent progress in text-to-speech synthesis, is an advanced neural network model, which can be used in different speech generation systems. WaveNet uses a sequential generation process in which a new sample predicted by the model is fed back into the network as input to predict the next sample until the entire waveform is generated.

This thesis studies training of the WaveNet model with speech spoken in a particular source style and generating speech waveforms in a given target style. The source style studied in the thesis is normal speech and the target style is Lombard speech. The latter corresponds to the speaking style elicited by the Lombard effect, that is, the phenomenon in human speech communication in which speakers change their speaking style in noisy environments in order to raise loudness and to make the spoken message more intelligible. The training of WaveNet was done by conditioning the model using acoustic mel-spectrogram features of the input speech. Four different databases were used for training the model. Two of these databases (Nick 1, Nick 2) were originally collected at the University of Edinburgh in the UK and the other two (CMU Arctic 1, CMU Arctic 2) at the Carnegie Mellon University in the US. The different databases consisted of different mixtures of speaking styles and varied in number of unique speakers.

Two subjective listening tests (a speaking style similarity test and a MOS test on speech quality and naturalness) were conducted to assess the performance of WaveNet for each database. In the former tests, the WaveNet-generated speech waveforms and the natural Lombard reference were compared in terms of their style similarity. In the latter test, the quality and naturalness of the WaveNet-generated speech signals were evaluated. In the speaking style similarity test, training with the Nick 2 yielded slightly better performance compared to the other three databases. In the quality and naturalness tests, we found that when the training was done using CMU Arctic 2, the quality of Lombard speech signals were better than when using the other three databases. As the overall results, the study shows that the WaveNet model trained on speech of source speaking style (normal) is not capable of generating speech waveforms of target style (Lombard) unless some speech signals of target style are included in the training data (i.e., Nick 2 in this study).

Keywords: Speech generation, Lombard style, Speaking style, WaveNet

# PREFACE

I would like to extend my deepest thanks to my supervisors Prof. Paavo Alku and Prof. Okko Räsänen for showing me a brilliant research path and guiding me during this process. They gave me freedom and encouragement to explore, and I am very glad our collaboration did not end here.

I would like to thank each and every member of our Speech Communication Technology team, especially Lauri Juvela and Sudarsana Kadiri. They always provided me great motivation, guidance and fun research atmosphere.

I would like to thank my dear friends Mehrdad Nahalparvari and Sina Rahimi Motem who were very supportive and caring and tolerated my many grumpy days.

I would like to thank my aunt, Sahar Zafari, and my uncle, Aidin Hassanzadeh for helping me to set sail on this unknown world called Finland and follow my dreams.

I dedicate this thesis to my dear parents and siblings whom without their love, support and encouragement, I would not be who I am today.

Tampere, 6th May 2020

Farhad Javanmardi

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# LIST OF SYMBOLS AND ABBREVIATIONS

| | |
|---|---|
| $F_0$ | Fundamental frequency |
| ANN | artificial neural network |
| BGMM | Bayesian Gaussian mixture model |
| CE | cross entropy |
| CNN | convolutional neural network |
| DBLSTM-RNN | deep bidirectional long short-term memory recurrent neural network |
| DBN | deep belief networks |
| DNN | deep neural network |
| DTW | dynamic time warping |
| FNN | feedfroward neural network |
| LSF | line spectral frequency |
| MLP | neural network |
| MOS | mean opinion score |
| MSE | mean squared error |
| NN | neural network |
| PSOLA | pitch synchronous overlap and add |
| RBM | restricted boltzmann machines |
| SGD | stochastic gradient descent |
| SSC | speaking style conversion |
| TTS | text-to-speech |
| VC | voice conversion |
| VT | voice transformation |

# 1 INTRODUCTION

Speech is the most important means to communicate between people. It is also an efficient way to transfer information. In addition to its linguistic message, speech also contains rich information about various speaker traits such as age, gender, state of health and native language. Thus, this type of information motivates researchers to study and analyze speech and consequently, develop methods to generate different types of speech signals artificially. Speech generation (artificially) refers to the production of high-quality speech from various inputs such as text, but also from other forms such as acoustical parameters or from other speech signals. For instance, people with hearing disorders as well as people with speech or language disorders can take advantage of these applications in the form of screen readers and digital personal assistants.

The emergence of neural networks has enabled building effective speech generation technologies for human-computer speech interaction. Examples of such technologies are text-to-speech (TTS), particularly in the form of statistical parametric speech synthesis [1], voice conversion (VC) [2] and speaking style conversion (SSC) [3]. Among these technologies, TTS is the most popular one employing speech generation. In TTS systems, text is first converted to linguistic features. The linguistic features are converted to acoustic features which are mapped to acoustic speech signals using speech generation. Even though some of the technologies mentioned have been studied for many years, the recent progress due to the emergence of deep learning has improved the performance of these technologies. In the case of speaking style conversion, the technology closely associated with the topic of this thesis, conventional signal processing methods (vocoders) have been used to analyze and synthesize the speech signal, and statistical methods such as standard Gaussian mixture models (SGMMs) [3] or Bayesian Gaussian mixture models (BGMMs) [3] or neural networks [4] have been used to convert the vocoder parameters from one style to another. Examples of speaking style conversion using conventional methods are the studies published in [3, 5].

Speech generation has been subject to a remarkable progress in the past five years due to advancements in generative deep learning models such as Tacotron [6], generative adversarial networks (GANs) [7] and WaveNet [8]. These methods take acoustic features as input and generate raw speech waveforms as output. WaveNet is one of the most widely used speech generation tools in recent TTS studies [9]. Conversion between speech of normal style and Lombard style (i.e. the speaking style that natural talkers adopt to when speaking in noise [10]) was studied in TTS recently using adaptation methods [11].

However, there are no studies in generating Lombard speech directly using the WaveNet model trained on normal speech. Besides, generating speech in an arbitrary style would help us to have more data for some topics in which it is difficult to gather training data. Thus, we can use the WaveNet model as a data augmentation method. For instance, training WaveNet on a database containing multiple talkers (healthy and Parkinson's disease) and generate speech signals for Parkinson's disease.

In this thesis, the WaveNet model is used to generate Lombard style from normal speaking style without using mapping models (as in speaking style conversion) or conventional signal processing methods (as in vocoding in TTS). Thus, the thesis studies acoustic-to-acoustic mapping (as in speaking style conversion) by focusing on one part of the system, speech generation, without using the other main component, parameter conversion. In other words, the goal of this thesis is to study the generation of speech waveforms of different speaking styles using the popular generative WaveNet model. We decided to use WaveNet because of its recent progress in generating speech samples. This autoregressive model generates the speech waveforms by predicting conditional probability distribution of a new sample given the past generated samples. Training of WaveNet is done by conditioning the model by using mel-spectrogram as an auxiliary feature. When the learning process is done, generating speech in target style is carried out. Thereafter, subjective evaluations are conducted to assess the performance of the model in the generation of Lombard speech.

The thesis is organized as follows. Chapter 2 presents the concept of speech generation as well as describes the speech conversion technologies developed earlier. Chapter 3 explains the theoretical background of neural networks, the convolutional neural network and WaveNet. Chapter 4 describes methodologies including pre-processing, feature extraction, model architecture and speech generation. The evaluation procedure details, the description of the databases used and the WaveNet training and testing are presented in Chapter 5. Chapter 6 reports the results of both the speaking style similarity test as well as the quality and naturalness test. Finally, Chapter 7 draws conclusions based on the presented results and discusses possible future works.

# 2 BACKGROUND

In this chapter, we will first give a short, general description of the characteristics of speech signals and then focus slightly more on the key speech communication attribute of this thesis, speaking style. After these parts, an introduction to speech conversion technologies will be given.

## 2.1 Characteristics of Speech Signals

The speech signal includes many types of information. In addition to its main component, the linguistic message, the speech signal includes lots of information about the speaker and about the environment where the speaker is. The speaker-specific information includes, for example, acoustic cues about the speaker's gender, age, emotional state and state of health. In general, the characteristics of speech signals can be analysed based on different speech features. These features can be categorized into the following two groups:

- **Segmental features:** The sound or timbre of a person's voice is defined by segmental features whose acoustic descriptors are formants (their frequency and bandwidth), and time domain energy. These features are influenced by both the emotional state of the speaker and by the physical properties of the speaker's speech organs [12]. Segmental features also depend on the linguistic content, as that is the primary driver of timbre/lower formants.
- **Suprasegmental features:** These features characterize the prosodic features related to the speaking styles, namely, fundamental frequency ($F_0$), intonation, energy (stress) and phone durations over the utterance. These features depend on the social and psychological status of speaker [13]. Suprasegmental features such as prosody also depend on intended message, structure of the given language etc., and hence are also guided by the linguistic structure of the language.

Speech signals show huge dynamics due to changes in linguistic contents, speaker, language and emotion. This thesis will focus on one speech attribute, the style of speaking. The general goal of the technology studied is to convert speech signals from one speaking style into another, e.g., from normal to whisper or alternatively from normal to Lombard, while maintaining the linguistic contents of the speech signal and the speaker identity. Since generating Lombard speech is the main scope of this work, we briefly introduce it in the following section.

### 2.1.1 Lombard Speech

The Lombard effect takes place when talkers change their speaking style in order to generate more intelligible speech in noisy environments [10]. The speaking style used in this context is called Lombard speech or speech-in-noise. Lombard speech shows changes in both acoustic and phonetic features compared to speech of normal style. Related to acoustic features, the Lombard effect causes an increase in formant amplitudes [14] and a decrease in formant bandwidths [14]. In addition, the Lombard effect raises $F_0$ and vocal intensity and decreases spectral tilt [15, 16]. Changes in phonetic properties include, for example, increased prominence in the production of vowels compared to consonants, and in the production of vowels and consonants compared to semivowels [17, 18].

## 2.2 Speech Generation

Speech generation refers to a process where acoustic speech signals are generated from different forms of information such as text or acoustic parameters. Examples of technologies that use speech generation are TTS, voice conversion and speaking style conversion. The process of speech generation can be considered generally to consist of three processes (analysis, manipulation and synthesis) as shown in Fig. 2.1. Vocoder for speech analysis involves the parameterization of speech in terms of acoustic features (i.e., feature extraction) which are feasible for manipulation and reconstruction of speech (i.e., vocoder for synthesis). For example, in the case of speaking style conversion, the source speaking style features are extracted by the vocoder in the analysis stage, then manipulated, and finally reconstructed by the vocoder in the synthesis stage to obtain the speech signal of the target style.

Speech synthesis typically refers to TTS, but it can be generalized to refer to any artificial generation of speech waveforms. One commonly used approach for building speech synthesizers in TTS is statistical parametric speech synthesis, in which text and speech are analyzed to get acoustic and linguistic features. Thereafter, the linguistic features are mapped to acoustic features (vocoder for analysis) through an acoustic model. Finally, at the synthesis stage, a previously unseen text is first converted to acoustic features by the acoustic model and is then processed through a waveform synthesis method (vocoder for synthesis) to generate target speech. Vocoders are used in both the speech analysis and synthesis stages because they describe a speech waveform using a parametric representation (a set of acoustic features). Therefore, vocoding enables the modification of speech to, for example, to increase its intelligibility [19].

There are two main groups of vocoders that can be used in speech generation: (1) conventional signal processing knowledge-based vocoders (which have been developed for

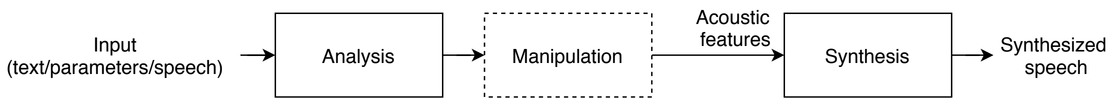


***Figure 2.1.*** *Typical framework for speech generation.*

TTS) such as STRAIGHT [20] and glottal vocoders [21, 22] and (2) learning-based neural vocoders such as the WaveNet vocoder [8]. Both glottal vocoders and STRAIGHT employ the source-filter model in which the speech signal is produced by convolving a source signal with a vocal tract filter. In the STRAIGHT vocoder, a source signal is spectrally flat consisting of impulses and noise, and the spectral envelope information is parameterised using mel-generalized cepstral coefficients [23]. In glottal vocoders, the speech signal is divided into the glottal excitation (i.e. the estimate of the true glottal volume velocity waveform generated by the vocal folds) and a vocal tract filter. The glottal excitation is not spectrally flat due to the different vibration modes of the vocal folds. The vocal tract filter is parameterized in glottal vocoders using line spectral frequencies (LSFs).

Today, neural vocoders have become the most popular vocoding methods for generating raw waveforms. In general, a neural vocoder is a trainable system which receives acoustic features as input and generates speech waveforms as output. An example of a trainable system is WaveNet, which learns to generate speech waveforms by conditioning the system on acoustic features and by modeling the distribution of the samples of the time-domain speech waveforms [8, 24, 25, 26]. Despite the fact that WaveNet has demonstrated its ability to generate high-quality speech, it is worth pointing out that the model uses an autoregressive architecture which calls for a long processing time in the system's learning and generating stages. Thus, simplifications of the WaveNet architecture have been proposed, including systems such as FFTnet [24] and WaveRNN [25]. This is discussed in more detail in Section 3.4.

## 2.3 Speech Conversion Technologies

Due to the emergence of deep learning, there is increasing interest in speech technology for different speech conversion technologies. Generally, speech conversion means converting speech of one type (e.g. speaker identity, emotion, speaking style) to speech of another type. The most well-known area of speech conversion is voice conversion (VC) which refers to changing the speaker identity characteristics of speech signals by keeping the linguistic contents unchanged. Another area of speech conversion technology is speaking style conversion (SSC). Unlike in VC, SSC aims to preserve both the linguistic contents and the speaker identify but to change the speaking style of the underlying talker. With the recent progresses of neural network methods, these two speech conversion technologies have shown success in their ability to conduct the underlying conversion task without compromising naturalness and quality of the speech signal [27, 28].

### 2.3.1 Voice Conversion

Voice conversion (VC) is a sub-field of speech conversion technologies aiming at mapping the speaker identity by keeping the linguistic content intact [29]. VC techniques manipulate speech timbre and its prosodic features such as intonation, $F_0$ and duration. In recent years, VC research has achieved considerable results with the help of advanced deep learning in applications such as transforming speaker identity [30], speech-to-speech translation [31] and personalizing TTS systems [32].

The overall framework of the VC system is illustrated in Fig. 2.2 and it is divided into two operation steps: (1) the training phase, which is an offline process, and (2) the conversion phase, which is an online process. In the training phase, a mapping function of speaker-dependent features——referred to as F(.)——is computed between source and target. There are some parts where the speech signal has to be processed before achieving the mapping function F(.). Input data are pairs of source and target features corresponding to speech signals of the same linguistic contents. First, both the source and target signals are processed to extract features such as spectral envelope, $F_0$, and aperiodic component in the speech analysis stage. Typically, these components are processed using two feature extraction methods, either generalized cepstral coefficients [23] or LSFs [33]. After the feature extraction, dynamic time warping (DTW) is used to align the features. Thereafter, the conversion function F(.) is obtained which can be applied to conduct the parameter mapping operation.

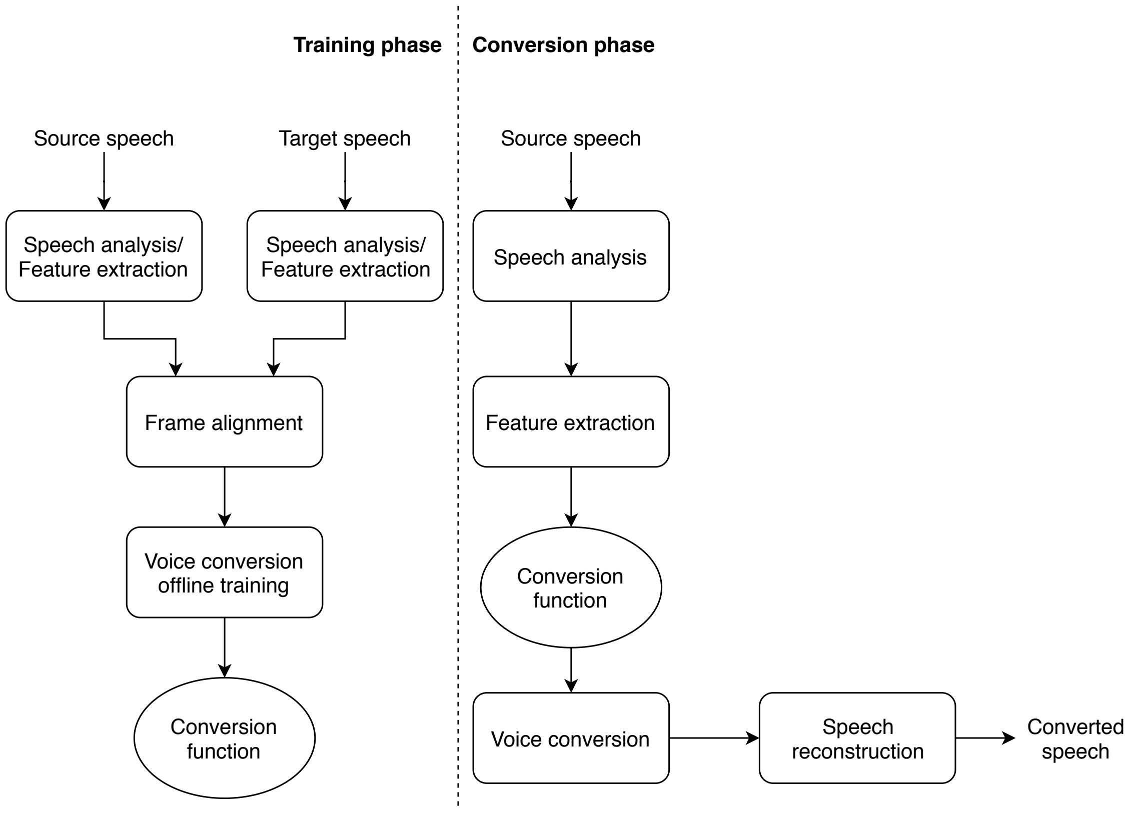


***Figure 2.2.*** *The overall framework for a voice conversion system.*

In the conversion phase, the speech analysis and feature extraction modules receive only the source speech signal. The conversion function F(.) takes these features as input and produces converted features as output. Finally, the converted features are processed by the reconstruction module to produce the converted speech signal. Both the speech analysis and reconstruction modules play an important role in VC, mainly by using some popular speech production models such as harmonic plus noise [34], the WORLD vocoder [35] and the STRAIGHT vocoder [20].

Many methods have been used in VC to build conversion functions. Some of the most popular techniques used are vector quantization [36, 37], Gaussian mixture models [38, 39, 40], unit selection methods [41], and neural networks-based methods, e.g., restricted Boltzmann machines (RBM) and its variations [42, 43, 44], deep belief networks (DBN) [45], deep bidirectional long short-term memory recurrent neural network (DBLSTM-RNN) [46].

### 2.3.2 Speaking Style Conversion

Speaking style conversion (SSC) is another example of speech technology where speech conversion is used. SSC aims at converting speech signals uttered by the speaker in one style to sound like the same speaker's speech produced using another style (e.g., from normal speaking style to shouting). In SSC, both the speaker identity and linguistic contents of the speech signal should remain unchanged. SSC can be used in different applications, e.g., in emotion conversion [47, 48], in speech intelligibility improvement [49, 50], and in TTS. In the last one, SSC can be used to expand the number of speaking styles the synthesis system can generate when the system is trained using speech from only one style.

Speaking style conversion can be done using two different approaches. The first approach is non-parametric and it corresponds simply to performing a direct transformation such as filtering to the source signal to achieve the conversion [51]. The second approach is to use a parametric, vocoder-based technique. A general block diagram of the parametric approach is shown in Fig. 2.3 and its idea is explained as follows. The parametric system has three main parts including feature extraction, mapping model and synthesis. First, the features from the input speech are extracted by the vocoder. There are various vocoders that can be used in this stage, namely, STRAIGHT [20], WORLD [35]), GlottHMM [22], GlottDNN [21]), Quasiharmonic model [52] and dynamic sinusoidal model [53]). After this process, the features are fed to the mapping model in order to produce a new set of features. The model learning can be divided into parallel learning which contains utterance pairs of source and target with the same linguistic contents, and non-parallel learning in which the target and source speech are of different linguistic contents. Moreover, the mapping model can be trained either in a supervised or unsupervised manner. Bayesian Gaussian mixture model (BGMM) [3] and feed-forward deep neural network are examples of parallel learning and cycleGAN [4] is an example of a recently used technique for non-parallel learning. In the final stage, the vocoder receives the mapped features and synthesizes the target speech of the desired style. Fig. 2.3 demonstrates the block diagram of a parametric SSC system.

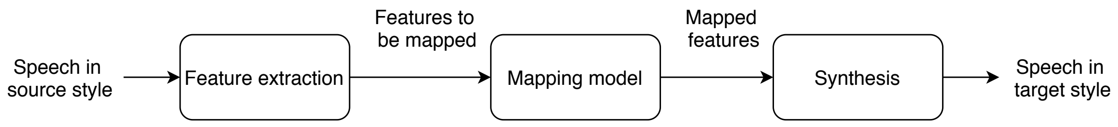


***Figure 2.3.*** *The general framework for a parametric speaking style conversion system.*

# 3 NEURAL NETWORKS

In recent years, many studies in speech processing applications have been conducted using a new family of models: artificial neural networks (ANNs). Example of applications that use ANNs are speech enhancement, automatic speech recognition, voice conversion, and more importantly for this thesis, speech generation in different styles. Therefore, the concepts of neural network, convolutional neural network and WaveNet are discussed in this chapter.

In machine learning, artificial neural network (ANN), also known as neural network (NN), is inspired by how the human brain processes information. ANN is determined by many parallel interconnected networks of adaptive components in which their organizations tend to act in a similar manner as in biological nervous systems [54]. In its basic structure, an ANN is composed of a network with simple processing units called neurons. An ANN is composed of simple processing units called neurons. Each neuron is joined to the preceding layer of neuron through weighted connection to transmit the information signal in the network. Its similarity with the brain demonstrates two characteristics of ANN: 1) neurons learn to represent regularities in the data, and 2) the regularities are stored in the connections of the neurons [55].

Neural networks methods are utilized to solve many machine learning tasks, such as classification, regression, clustering and time-series prediction. Over the years, there have been a variety of ANN architectures introduced in which neurons, layers and activation functions are the common elements.

**Neuron:** A neuron, also called a unit or a node, is the principal component of ANNs. In the brain, synapses transfer information (stimulus) between biological neurons. The amount of stimuli determines whether neuron can either generate electrical impulse or not. This phenomenon is implemented by ANNs in such a way that each neuron receives weighted inputs through its connections from other neurons. In other words, neurons (except those at the input layer) receive their inputs from the previous layer and compute outputs for the next layer of neurons. Each input is multiplied by weight $w$ and the sum of the weighted inputs is calculated. Finally, the output is produced by an artificial neuron depending on its activation function. For example, perceptron is one of the most important artificial neurons introduced by Frank Rosenblatt [56] in 1958. A perceptron receives several inputs $\{x_1, x_2, ..., x_i\} \in \mathbb{R}$ and computes the weighted sum of these inputs with weights $w_i$, and then produces the output $y$ depending on a threshold value (interchangeably called as bias) $b$. The output is calculated as:

$$y(x) = \begin{cases} 1, & \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0, & \sum_{i=1}^{n} w_i x_i + b \leq 0 \end{cases} \tag{3.1}$$

where $n$ is the number of inputs ($x_i$) to the perceptron, $w_i$ represents the weights, $b$ is the bias and $y$ denotes the output. The step function is used as an activation function in perceptron. Fig 3.1 shows the structure of a simple perceptron.

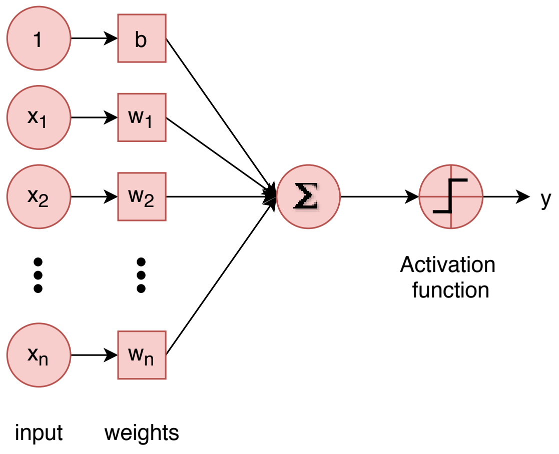


***Figure 3.1.*** *A perceptron structure. $x_i$ represents the inputs, $y$ is the output, $w_i$ are the weights and $b$ represents bias. A perception uses step function as an activation function.*

**Layers:** Layers are composed of a group of artificial neurons in the network. Neural networks typically consist of three types of different layers: input layer, hidden layer and output layer. Input layer contains $D$ passive neurons, where $D$ is the dimensionality of the input data. Each neuron gets one sample of input $x$ and delivers the output to the next layer of neurons. The hidden layers are between input and output layers and they are in charge of executing intermediate computation in the network. The hidden layers are the main factor in the learning process. An ANN with more than one stacked hidden layers is known as a deep neural network (DNN). The complexity of the network is mainly determined by the design of the hidden layers (number of nodes, activation functions etc.). DNN is explained in detail in Section 3.1. The output layer is composed of neurons that compute the posterior probabilities of the output for input $x$. For example, in classification tasks, the output of neuron $z$ is the posterior probability for the input belonging to class $z$. These probabilities are in the range of [0 1] and can be converted to binary outputs using a certain threshold value, i.e., "Class $k$ or Class $z$". An ANN with three different layers is depicted in Fig. 3.2.

**Activation Function:** In an artificial neural network, activation functions are considered as essential components that convert the input signal of a neuron into an output signal. Activation functions introduce non-linearity in the network. Fig. 3.3 illustrates some important activation functions more commonly utilized in ANNs.

**Logistic function:** Logistic function is also known as sigmoid function and it is mathe-

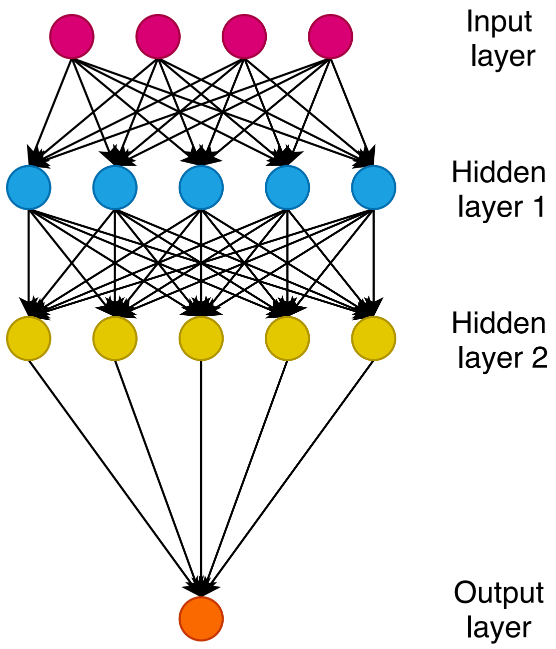


***Figure 3.2.*** *The representation of artificial neural network.*

matically expressed as:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{3.2}$$

Sigmoid neuron is a popular artificial neuron that uses sigmoid activation function $\sigma$. This function is a smooth approximation of the step function used in perceptrons. The output value is between 0 and 1 and thus it is generally employed in classification tasks.

**SoftMax:** SoftMax function is a more generalized form of the Logistic activation function and it is particularly useful in multiclass classification tasks because the output of softmax is interpreted as the probability distributions of a list of possible outcomes. The output of softmax is defined as:

$$\mathrm{softmax}(x_i) = \frac{e^{x_i}}{\sum_n e^{x_n}}. \tag{3.3}$$

**Rectified linear unit (ReLU):** ReLU has the activation function:

$$ReLU(x) = max(0, x) \tag{3.4}$$

where $x$ is the weighted sum of the inputs to the ReLU. This activation function is similar to the characteristics of a biological neuron, because it promotes sparse representations in the network [57]. This function grows unbounded for positive values of $x$ and is 0 for negative values of $x$. The unboundedness feature of ReLU implies that the neuron does not saturate for large values and hence it converges faster.

**Hyperbolic tangent (tanh):** This function can be used as an alternative function to the Logistic function which scales the output to the range of [-1 1]. The output of the hyperbolic tangent function is defined by

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{3.5}$$

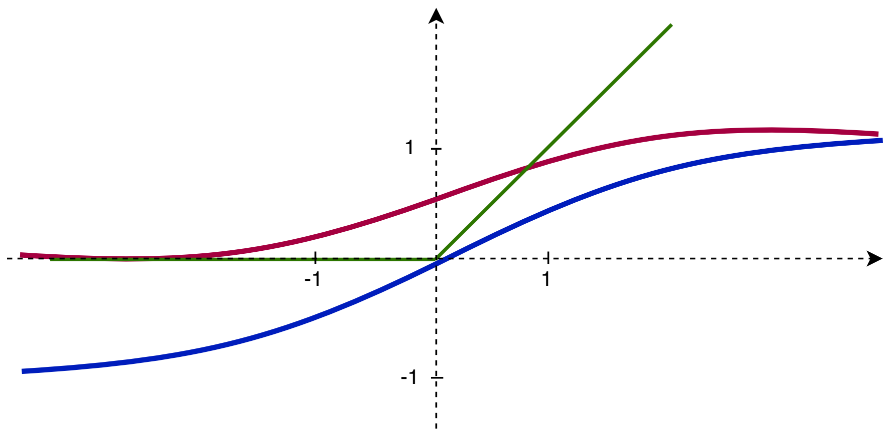


***Figure 3.3.*** *Visual representation of activation functions, rectified linear unit (ReLU) in green, sigmoid function in red and hyperbolic tangent in blue.*

Next, we will shortly describe multi-layer perceptrons (MLPs) which are prototypical neural networks [58, 59]. The universal approximation theorem [60] promotes MLP's expressive power and states that for any bounded continuous and nonlinear function, it is always possible to approximate the function to an arbitrary degree of accuracy using an MLP with a hidden layer. MLPs with multiple hidden layers work much better in representing complex and structured functions in comparison to shallow networks. This is because MLPs with multiple hidden layers are able to divide the whole training input space into more linear parts exponentially [66, 56]. Some pattern recognition tasks such as handwritten digit recognition [61] and speech recognition [62] have shown good results using MLPs.

In this neural network architecture, all the nodes are organized in sequential layers that only take inputs from the previous layer of nodes. Fig 3.2 represents a simple MLP with input, output and two hidden layers. In this prototype case, all layers are fully connected via a weight matrix and each neuron has a bias and a non-linear activation function. An MLP calculates the hidden activation vector $h$ and the output $\hat{y}$ for input vector $x$ as:

$$h = F(W^{ih}x + b^h) \tag{3.6a}$$

$$\hat{y} = G(W^{ho}h + b^{\hat{y}}) \tag{3.6b}$$

where $W$ is the weight matrix, i.e., $W^{ih}$ are the weights from input to hidden layer and $W^{ho}$ from hidden layer to output layer, $b$ is the bias vector, and $F$ and $G$ are activation functions that are always computed element-wise. Note that notations $x$ and $W$ indicate vectors and matrices respectively.

In the network, the output layer computes a prediction $\hat{y}$ for an input $x$ that is compared to the original output $y$ using a cost function $E(W, b; x, y)$, or just $E$ for the sake of simplicity.

The network is trained to minimize cost function $E$ for all training samples $x$. The cost function (also called loss function) evaluates the performance of the network. One of the most common cost functions is cross entropy (CE) which is generally used in classification tasks with the following formula:

$$CE = \frac{1}{N}\sum_{n=1}^{N} y_n \log \hat{y_n} + (1-y_n)\log(1-\hat{y_n}) \tag{3.7}$$

where $N$ is the number of training examples, and $y_n$ and $\hat{y_n}$ are the target and prediction outputs of sample $x_n$, respectively. The mean squared error (MSE) is another main cost function to train ANNs:

$$E_{MSE} = \frac{1}{N}\sum_{n=1}^{N} \|y_n - \hat{y_n}\|^2 . \tag{3.8}$$

Since the cost function $E(W,b)$ always depends on $W$ and $b$, gradient descent methods are utilized to minimize the cost function during training. The idea is to find the gradient for a given random weight, and repeatedly update the weights by taking small steps towards the negative direction of the gradient. Backpropagation is an effective algorithm mainly used to calculate the gradients for all the weights. The backpropagation algorithms [58, 59] are applied through a method called the chain rule for partial derivatives along the network.

In order to explain how the backpropagation algorithm works on an MLP, the following notation is used: $w'_{ji}$ is the weight between the $i^{th}$ neuron in layer $l-1$ and the $j^{th}$ neuron in layer $l$. Moreover, $z^l_j$ is the weighted input to the $j^{th}$ neuron in layer $l$, and $F'$ is the first derivatives of the activation function $F$. Finally, $h^{l-1}_i$ is the activation of the $i^{th}$ neuron in layer $l-1$, i.e.,

$$z^l_j = \sum_i w^l_{ji} F(z^{l-1}_i) + b^l_j = \sum_i w^l_{ji} h^{l-1}_i + b^l_j \tag{3.9}$$

where $h^{l-1}_i = F(z^{l-1}_i)$.

As previously explained in this chapter, we need to perform gradient descent to train the parameters in the network. Since these parameters are differentiable, cost function $E$ can be minimized using gradient descent so that it computes the derivatives of cost function with respect to the weights $W$ and bias $b$ terms, i.e., $\frac{\partial E}{\partial w^l_{ji}}$ and $\frac{\partial E}{\partial b^l_j}$. After computing these gradients, the weights and biases are updated by moving a small step in the direction of the negative slope. That is, in the case of using stochastic gradient descent(SGD),

$$w \equiv w - \eta \nabla E(w) \tag{3.10a}$$

$$\Delta w_i(\tau+1) = -\eta \nabla E(w_i) = -\eta \frac{\partial E}{\partial w_i} \tag{3.10b}$$

where $\Delta w_i(\tau+1)$ is the weight update, $\tau$ is the index of training (epochs), and $\eta$ is the learning rate that specifies how much the weights can change on each update. The same update rule applies to the bias by replacing $b$ with $w$.

Backpropagation is a technique that gradient descent uses to calculate the gradients of the cost function by computing the relationship between the error term and all weights and biases in the network. This can be done by propagating the errors at the output layer backwards through the network. First, for each node in the output layer $L$, the backpropagated error $\partial_j^L$ is obtained as:

$$\partial_j^L \equiv \frac{\partial E}{\partial z_j^L} = \frac{\partial E}{\partial h_j^L}\frac{\partial h_j^L}{\partial z_j^L} \tag{3.11}$$

Then, the backpropagated errors $\partial_j^L$ in the $l^{th}$ layer with respect to the backpropagated errors $\partial_{j+1}^L$ in the next layer is calculated:

$$\partial_j^l \equiv \frac{\partial E}{\partial z_j^l} = \sum_i \frac{\partial E}{\partial z_i^{l+1}}\frac{\partial z_i^{l+1}}{\partial z_j^l} = \sum_i w_{ij}^{l+1}\partial_i^{l+1}F'(z_j^l) \tag{3.12}$$

where we used Equation (3.9) to derive

$$\frac{\partial z_j^{l+1}}{\partial z_j^l} = \frac{\partial}{\partial z_j^l}\sum_i w_{ij}^{l+1}F(z_j^l) + b_i^{l+1} = \sum_i w_{ij}^{l+1}F'(z_j^l) \tag{3.13}$$

and from the definition in 3.11

$$\partial_i^{l+1} \equiv \frac{\partial E}{\partial z_i^{l+1}} \tag{3.14}$$

For the gradient $\frac{\partial E}{\partial w_{ji}^l}$ in terms of error $\partial_j^l$, we have

$$\frac{\partial E}{\partial w_{ji}^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{ji}^l} = h_i^{l-1}\partial_j^l \tag{3.15}$$

where we used the equivalence

$$\frac{\partial z_j^l}{\partial w_{ji}^l} = \frac{\partial}{\partial w_{ji}^l}\sum_i w_{ji}^l h_i^{l-1} + b_j^l = h_i^{l-1} \tag{3.16}$$

and for the gradient $\frac{\partial E}{\partial b_{ji}^l}$

$$\frac{\partial E}{\partial b_j^l} = \frac{\partial E}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \partial_j^l \tag{3.17}$$

where the $h_i^{l-1}$ term is eliminated when computing

$$\frac{\partial z_j^l}{\partial b_j^l} = \frac{\partial}{\partial b_j^l}\sum_i w_{ji}^l h_i^{l-1} + b_j^l = 1. \tag{3.18}$$

While stochastic gradient descent is used as a popular optimization strategy, learning with it can lead to slow convergence. This is because frequent updates might lead the gradient descent into competing directions which means that it takes a longer time for the network to minimize the loss function. Batch-based optimization is a technique that allows to speed up the learning by calculating the gradient and updating the parameters of the network once a small sample of randomly chosen training inputs has passed through the network. There are also several optimization methods that can increase convergence speed, such as momentum, Adam [63], adagrad [64], adadelta [65], RMSprop [66], Nesterov accelerated gradient [67]. Here we briefly explain stochastic gradient descent with Adam, which is used in the next sections of this project.

**Adam:** The Adam optimization algorithm [63] is an alternative optimizer to classical stochastic gradient descent that can be utilized to iteratively adjust the weights and biases during training. In this method, individual adaptive learning rates are calculated for network parameters using the estimation of first-order and second-order moments of the gradient. More specifically, an exponential moving average of the gradient and the squared gradient are computed by the algorithm under control of two parameters $\beta1$ and $\beta2$. The algorithm benefits from two other extensions of SGD; adagrad which works well on tasks with sparse gradients, and RMSprop which works well on online and non-stationary tasks. The Adam as adaptive learning rate optimization decreases the fluctuations of gradient in irrelevant directions of features space in comparison to vanilla SGD. Moreover, the speed of convergence is faster for Adam optimizer [63].

In this section, we have introduced the MLP network which is referred to as feedforward neural network (FNN) described in Section 3.2. Another types of FNN, convolutional neural networks (CNNs) and the WaveNet, which is composed of a deep CNN are presented in detail since they are used in the subsequent sections of this work. Before presenting the details of the mentioned networks, we briefly explain deep neural networks.

## 3.1 Deep Learning

ANNs with more than one hidden layer are known as deep neural networks (DNNs) [68, 69]. DNNs have recently shown discriminative and representation learning capabilities in several application domains such as computer vision, automatic speech recognition and natural language processing.

An ANN with a single hidden layer can approximate any function. However, it may fail to perform in some complicated tasks where the data is not large or the input does not have sufficient features. Moreover, some experiments have shown that shallow network architectures can not efficiently find desirable representations of their inputs [70, 71] Furthermore, limited training data represents a sparse subset of samples from the whole population in real life. Since ANNs are data-driven networks, they can only approximate the true distribution based on the training data. Therefore, the evaluation of ANNs using unseen samples from under-represented parts of distribution will lead to a dissatisfying performance of ANNs, in other words poor generalization [72]. On the other hand, DNNs are the expanded version of ANNs are better in terms of extracting hidden patterns from the training inputs due to the extensive number of parameters [69]. This benefits the

DNNs with the greater ability of learning higher level representation of input data, and subsequently with the superior potential of approximating the under-represented regions of population in training data. Thus, the DNNs can better generalize to unseen samples from under-represented spaces in the training data [63].

The use of deep architectures has shown significant success which is due to both technological reasons and theoretical factors. For the former, developments in computational technology paved the way for deep architectures. In particular, multi-processor graphics cards or GPUs enhanced computational power to speed up the learning process with large datasets, enabling involving millions of network parameters in the models. For the theoretical factor, the inventions of new algorithms such as unsupervised pre-training, ReLU [57] and dropout [73] have resulted in significant improvements in classification tasks in image [74] and speech recognition [75].

Despite the obvious advantages of DNNs, training the DNNs can be problematic. As previously mentioned in this chapter, stochastic gradient descent is typically applied with the backpropagation algorithm to update the network parameters by minimizing the cost function in training. This algorithm is guaranteed to reach a local minimum regardless of the depth of the networks. However there is no guarantee that the training reaches the global minimum of the cost function for large and deep networks. Instead, a local minimum, which is typically close to the global minimum error will be reach for networks [76, 77]. In addition, random initialization plays an important role in the performance of the deep networks, in which case the network weights and biases are initialized with small random values for all neurons. Poor initialization can affect the learning process by increasing the time to converge the network training to a desirable accuracy [68].

## 3.2 Feedforward Neural Networks

One of the most common ANNs is the feed-forward neural network. ANN is called FFN when there is no cycle between the connections in the network. In the other words, in this kind of networks, the information is always propagating forward from the first layer of neurons to the last layer of neurons which produces the outputs. FNNs are mainly fully connected layers, see Fig 3.2, so that a neuron is only connected to neurons from the previous and to following layers. MLP described earlier in this chapter and CNNs described in Section 3.3 are the most widely used types of FNNs [78, 79].

## 3.3 Convolutional Neural Networks

Convolutional neural networks (CNNs), also known as ConvNet, are another type of FNN, proposed in the late 1990s by Le Cun for handwritten digit recognition [80]. CNNs have been shown to be highly powerful neural networks yielding excellent results both in generative and discriminative tasks in image, speech, audio, text and video.

As it is evident from the name, CNNs utilizes a mathematical linear operation which is known as *convolution*. In other words, CNNs are the neural networks in which convolution operation is applied instead of general matrix multiplication in at least one of their convolutional layers. Generally, the convolution operation between two real-valued functions is the integral of the product between one of the functions and the reversed and shifted version of the other one. Since the data is discretized in a computer and represented as integer values, the convolution operation between two discrete functions $x$ and $k$ is defined as:

$$x * k = \sum_{\tau=-\infty}^{\infty} x(\tau)k(t-\tau) \tag{3.19}$$

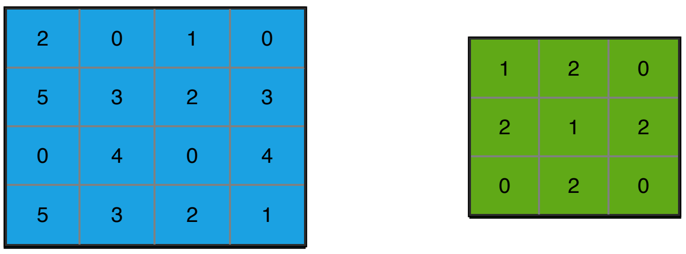
where the asterisk ($*$) denotes the convolution operation. According to CNNs terminology, the first function $x$ represents the input and the second function $k$ is referred to a learnable filter or kernel.

In CNNs, convolution operation is done when a kernel with a specific size, e.g., 5x5, is passed through the input—image—so that the elements of the kernel are convolved with the elements of the input and the resulting products are summed up to produce the output. Through this procedure, kernel is learned and optimized in a way that they could capture important features of the input. Mathematically, we can define a discrete two-dimensional convolution operation as:

$$Y(i,j) = (X * K)(i,j) = \sum_{m}\sum_{n} X(i+m, j+n)K(m,n) \tag{3.20}$$

where $K$ is the filter, $X$ is the input and $Y$ is the output which is also called the feature map. Fig 3.4 shows an example of a 2-D input image and a kernel. In order to clarify how convolution operation works, we demonstrate an example of a 2-D convolution of the input image with a kernel in Fig 3.5. First, the elements of the orange region—the receptive field in input—are multiplied by the elements of the kernel. Then the results are added to compute the output of convolution for the corresponding region which is demonstrated in red. This process is continued until producing the final output. Since the dilated causal convolution is employed in the neural model studied in this thesis, the WaveNet model, we also briefly introduce the causal and dilated convolution operations.

**The causal convolution operation:** In this type of convolution operation, each element



| | | | |
|---|---|---|---|
| 2 | 0 | 1 | 0 |
| 5 | 3 | 2 | 3 |
| 0 | 4 | 0 | 4 |
| 5 | 3 | 2 | 1 |

| | | |
|---|---|---|
| 1 | 2 | 0 |
| 2 | 1 | 2 |
| 0 | 2 | 0 |

***Figure 3.4.*** *The blue table represents a (4 x 4) input and the green table represents a (3 x 3) kernel.*

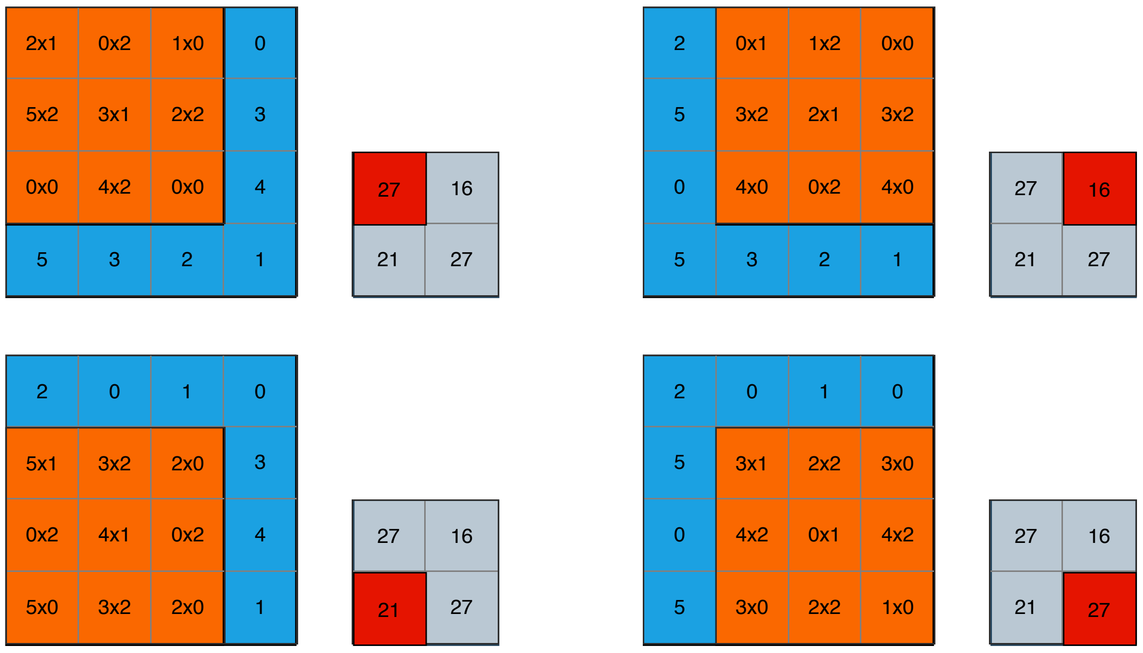


***Figure 3.5.*** *Convolution operation between an input image and a kernel presented in Figure 3.4.*

of the output is computed from the present and past elements in the input. In other words, the output value does not depend on future input values. For simplicity, we present an example of causal convolution for a 1-D input in Fig. 3.6. The output values (shown in red) are produced by computing the dot products of the kernel with the corresponding elements of the input.

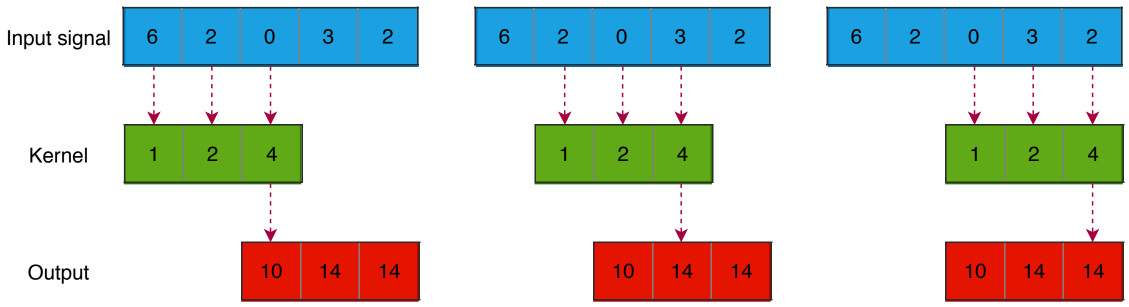


***Figure 3.6.*** *1-D causal convolution operation between an input signal and a kernel.*

**The dilated convolution operation:** In the dilated convolution, a dilation factor $d$ defines which elements of the input are skipped in the convolution operation. For instance, $d = 2$ specifies that every $2^{nd}$ element of the input is skipped when convolving by the elements of the kernel. Fig 3.7 shows a dilated convolution with $d = 2$ and a kernel size $3$. In this example, a new kernel size is generated by adding zeros between the values of the kernel in order to skip every other element of the input. Then the output is simply calculated by summing the resulting products of the input's elements and the kernel's elements.

**Stride:** The stride is one of the methods that can be performed in a convolution operation. The stride defines how the filter moves from one point to the next point: when the stride

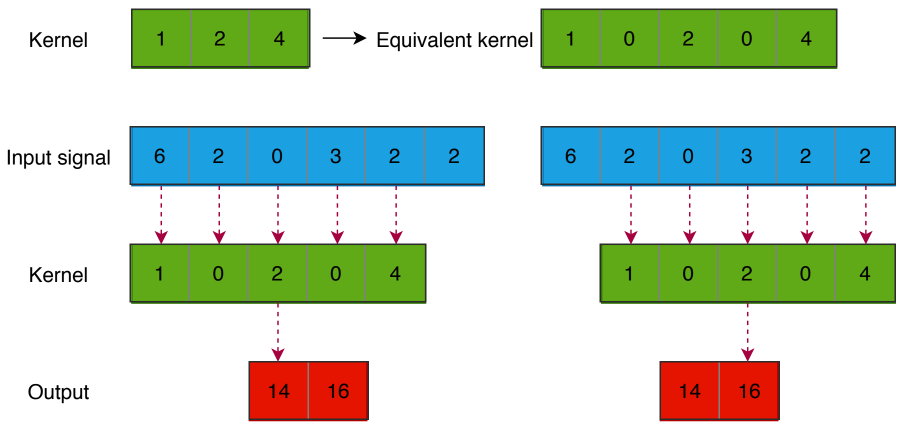


***Figure 3.7.*** *1-D dilated convolution operation between an input signal and a kernel.*

is larger than one, some parts of the input are ignored by the convolution operation. In Fig 3.5 we used 1 x 1 strides in the convolution.

**Zero padding:** Zero padding is another important method that specifies the size of feature maps. This method pads the input with zeros around the border before convolution. The convolution operation without zero padding——known as the valid convolution—— might lead to shrinkage of the feature maps in each convolution operation. Alternatively, with zero padding (adding zeros in each axis of the input), the kernel gains access to the bordering elements in the input, and thus, the feature maps will be of the same size as the input.

In addition to convolution, there are two more operations that are commonly used in CNNs: ReLU activation function and pooling operation. ReLU is mostly applied after every convolution operation so that the output of a convolutional layer is received to the activation function in order to produce a non-linear output. In a CNN, since convolution is a linear operation, ReLU is used to introduce non-linearity in the network. Hence, non-linear real-world phenomena can be modeled by the CNN. Thereafter, pooling operation reduces the feature map size yet maintaining the main information. Typically, It helps to make the output of the convolution operation invariant to small changes of the input. This layer operates on the width and height of the feature map and resizes it using two different pooling operations:

- **Max Pooling:** In this method, a small window of an arbitrary size is specified (for example, a 2 X 2 window) and the largest value of each window on the feature map is the output of max pooling operation.
- **Average Pooling:** This method calculates the average value of each window on the feature map.

An example of max pooling and average pooling operations are shown in Fig 3.8.

Besides, there is an output layer at the end of CNNs which receives the feature map from several convolution and pooling operations. This feature map preserves high-level features of the input. Therefore, the output layer uses this high-level information to produce

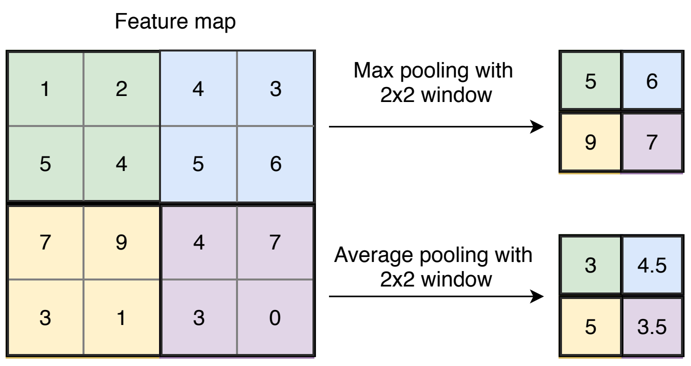


***Figure 3.8.*** *The table in top right presents max pooling operation and the table in bottom right shows average pooling operation.*

the output. As an example for classification task, the output layer uses a softmax activation function to compute the probability of each class given the input data. In other words, it classifies the input data into the existing classes. Fig 3.9 demonstrates an example of CNN in image classification.

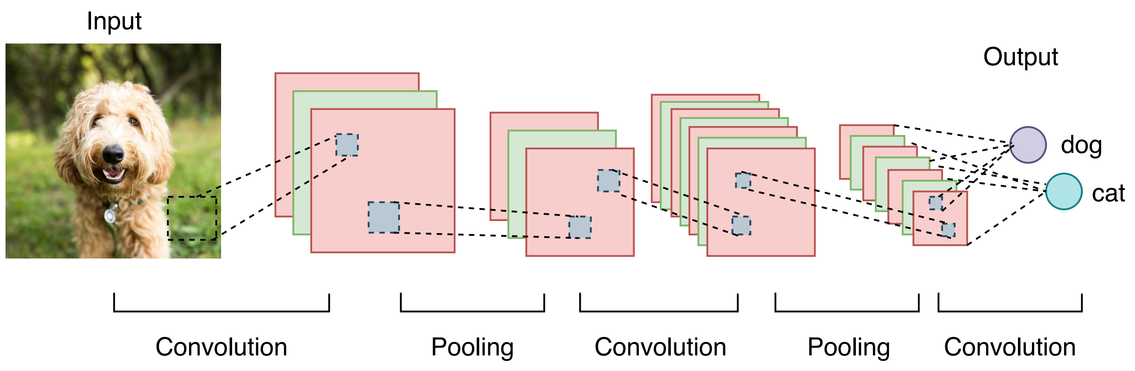


***Figure 3.9.*** *A schematic diagram of convolutional neural networks in image classification. Figure inspired from [81].*

In summary, the CNN can be characterized by the following issues:

- Typically, CNNs uses three different operations, e.g. convolution, activation (ReLU), pooling.
- CNNs are widely used in classification tasks where they take the input, process it and assign it to certain classes.
- CNNs are able to eliminate useless parameters while retaining necessary information.
- CNNs are implemented in such a way that convolutional layers receive 1-D, 2-D or 3-D inputs and correspondingly produce 1-D, 2-D or 3-D outputs.
- The size of the output depends on the input size, zero padding, the stride and the kernel size.
- Convolution operation has parameters, but activation (ReLU) and pooling operation do not.

## 3.4 WaveNet

WaveNet, introduced by Google in [8], is one of the most well-known deep generative neural network models utilizing CNNs. WaveNet has in recent years become a remarkably effective technique to solve complex tasks in speech processing. It has shown extensive progress in many areas of speech technology including TTS [9], speech enhancement [82] and voice conversion [83, 84]. WaveNet was inspired by PixelCNN [85]. Unlike PixelCNN, which automatically generates the contents of a 2-D image by predicting pixels from its nearest neighbors, WaveNet operates on 1-D time-series audio data to generate raw signal waveforms. Therefore, it has quickly become a popular tool in speech generation because of its flexibility to generate time-domain speech waveforms using acoustic features in conditioning the model.

The WaveNet generative model is capable of learning probability distributions of the input data. In other words, it computes the conditional probability distribution for sample $x_n$ given previous predicted samples $\{x_1, ..., x_{n-1}\}$. Thus, the probability of a waveform $x$ is expressed as:

$$p(x) = \prod_{n=1}^{N} p(x_n | x_1, ..., x_{n-1}). \tag{3.21}$$

This formula expresses two main aspects of WaveNet. The first aspect refers to the fully probabilistic property of WaveNet in which it calculates a probability distribution and chooses a discrete value with the highest probability from the distribution. The second aspect is WaveNet's autoregressive structure where the past generated samples are used to produce the next sample. The WaveNet architecture is shown in Fig. 3.10.

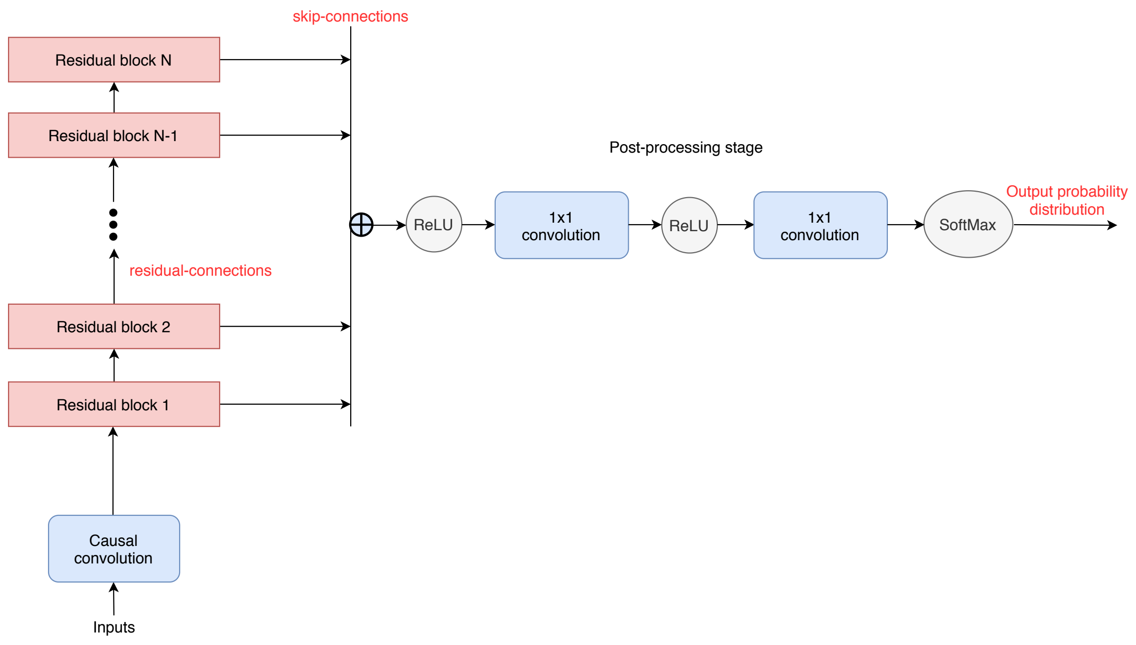


***Figure 3.10.*** *The WaveNet architecture. Figure adapted from [8].*

According to Fig 3.10, WaveNet is composed of a stack of residual blocks including dilated

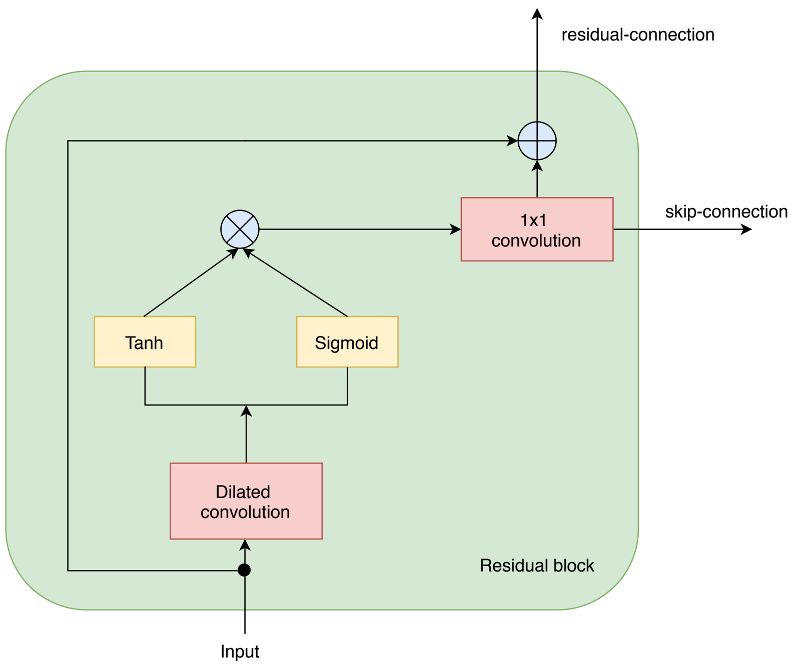


***Figure 3.11.*** *The residual block architecture.*

causal convolutions which are responsible for extracting features, and a post-processing part that receives information from each residual block and processes it to produce the output.

**Residual block:** The residual block uses two shortcut connections, i.e., residual and skip connections, to speed up the convergence and shorten the training time. In addition, both residual and skip connections ease the process of gradient propagating for all layers which helps to avoid the vanishing gradient problem. Moreover, both connections carry features from data, residual connection is passed to the next layer and skip connection to the model output. Fig 3.11 demonstrates the structure of the residual block.

**Dilated causal convolution:** Since the causal convolution guarantees that generating a new sample is dependent on previous samples, it requires many layers to increase the receptive fields which are necessary to generate a waveform. Thus, WaveNet uses dilated causal convolution by doubling the dilation for each layer and resets it at certain intervals to provide large receptive fields with a few layers. Furthermore, this architecture decreases the computational cost.

**Gated activation function:** This unit is responsible for introducing non-linearity in the network after the dilated causal convolution operation. It is formulated as:

$$z = tanh(W_{f,k} * x) \odot \sigma(W_{g,k} * x) \tag{3.22}$$

where $W * x$ represents a dilated convolution operation, $\odot$ is an element-wise multiplication, $f$ and $g$ are the hyperbolic tangent and sigmoid activation functions, respectively, $k$ is layer index and $W$ represents learnable kernels.

**WaveNet with conditioning:** WaveNet predicts the conditional probability distribution of a sample based on both previous generated samples and auxiliary information. Conditioning WaveNet with extra inputs aids to control the characteristics of the generated speech utterances. For example, in the WaveNet reported in [8], linguistic features and/or speaker codes were conditioned to train the network aiming to generate speech while maintaining certain speaker characteristics. Equation 3.21 can be written as follows:

$$p(x|h) = \prod_{n=1}^{N} p(x_n|x_1, ..., x_{n-1}, h) \tag{3.23}$$

where $h$ is auxiliary information. Training WaveNet without $h$ would generate sample $x_n$ that has the highest probability value depending on the past predicted samples, which means that the final result corresponds to a generalization of what WaveNet has learnt to generate [86]. For instance, in order to generate a sequence of speech for a single speaker, WaveNet would generate a mixed sequence of phones from multiple speakers' voices. Fig 3.12 shows a conditional WaveNet architecture and Equation 3.23 with conditioning is written as:

$$z = tanh(W_{f,k} * x + C_{f,k} * H) \odot \sigma(W_{g,k} * x + C_{g,k} * H) \tag{3.24}$$

where the $C$ is the learnable kernel, $C_{f,k} * H$ represents 1x1 convolution operation and $H$ is the transformed feature. It should be noted that both input speech and $H$ have the same time-resolution after computing $H$ = $f(h)$.

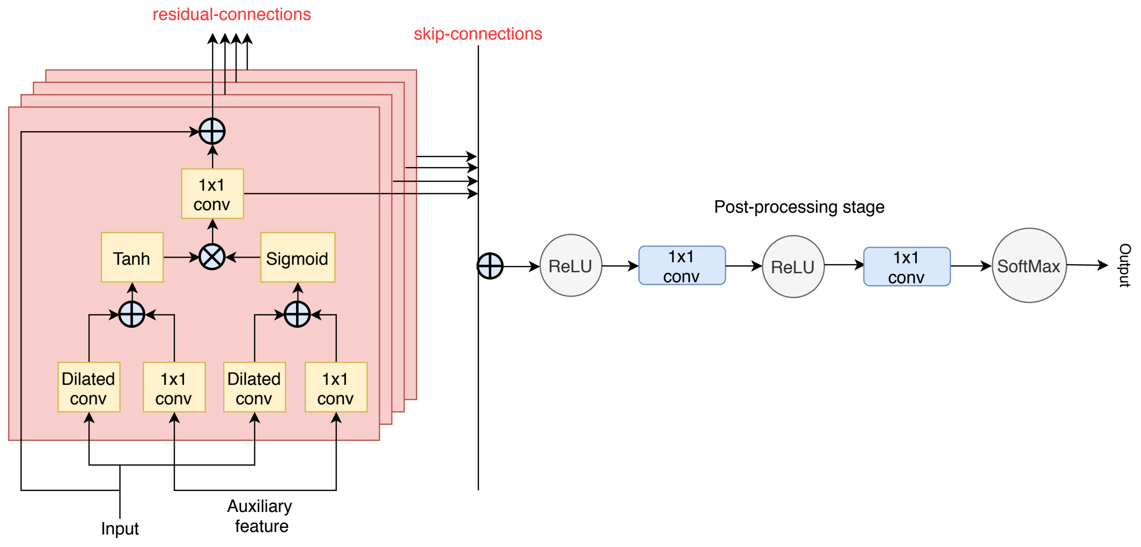


***Figure 3.12.*** *The conditional WaveNet architecture.*

# 4 WAVENET-BASED GENERATION OF SPEECH IN DIFFERENT SPEAKING STYLE

In this chapter, we describe the approach studied for generating speech in different speaking style using WaveNet. The WaveNet model receives a speech uttered in a normal style as input and outputs a speech utterance in Lombard style. Furthermore, we train the model by conditioning it using a time-frequency representation of speech, i.e. the mel-spectrogram.

## 4.1 System Overview

Fig. 4.1 depicts the block diagram of the system which is split into the training and test parts.

During the training phase, the input speech signal is converted to $25$-ms frames with an overlap of $80$%. Then, a time-frequency domain representation, the mel-spectrum, is calculated for each frame as an acoustic feature for conditioning. The obtained data is divided into the training, validation and test sets. Before feeding the training and validation sets to the network, the input waveforms are quantized into 256 possible values using the $\mu$-law transformation. Finally, the conditioned WaveNet is trained with the encoded input speech signal and its corresponding acoustic features to predict the conditional probability for a new sample given the previous generated samples. Once the learning process is completed, the system is ready to generate speech waveforms from the test set mel-spectrograms using the trained model. The evaluation procedure, experiments and results will be discussed in the following chapters.

## 4.2 Feature Extraction

Time-frequency representations are widely used in speech processing applications. The WaveNet model takes advantage of auxiliary information to learn the prosodic features of speech (e.g. intensity, $F_0$, etc.). Thus, we need to process the speech signal to compute these acoustic features. This is because each speech signal contains the characteristic pattern of spectral magnitudes over time which can be utilized as auxiliary information in the speech generation process.

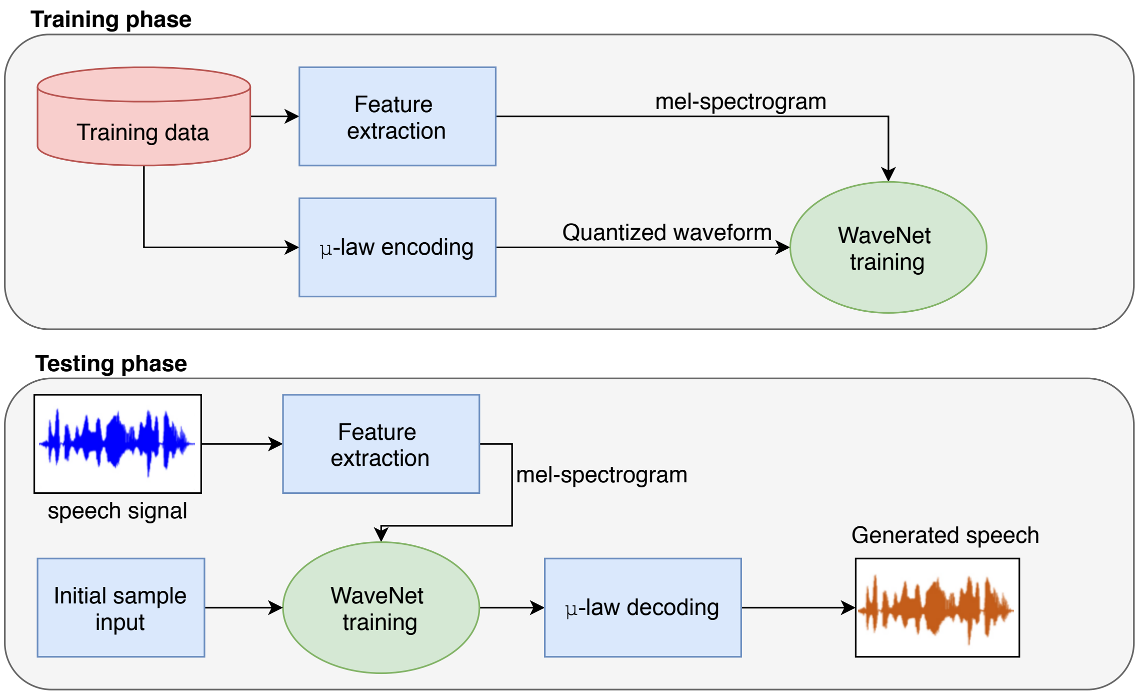


***Figure 4.1.*** *The overview of the system in the training and testing phases.*

There are some steps, referred to as pre-processing, before extracting features from a speech signal. In the pre-processing phase, the speech signal needs to be split in frames in order to capture the spectral information using the Fourier transform. Computation of the frame-based Fourier involves windowing the speech waveform with a soft window such as the Hamming, Hann or triangle window. In this work, we use the Hamming window with a duration of $25$ ms and and with an overlap of $80\%$ defined as:

$$w_n = 0.54 - 0.46\cos(\frac{2\pi n}{N-1}) \tag{4.1}$$

where $N$ is the length of the Hamming window and $n$ = 1,2,...,$N$. In conditioning WaveNet, the spectrum is transformed to a perceptual frequency domain representation using log mel energies, which will be describe next.

### 4.2.1 Log Mel Energies

The mel scale is designed to simulate the frequency-selectivity of the human auditory system. It is also a scale of pitches in which the frequency interval between each mel band is perceptually equal. The human auditory system has higher resolution for low frequencies and therefore frequency intervals in successive mel bands are wider for high frequencies. Conversion of frequency from the $f_{Hz}$ scale to the mel scale $m$ is computed using the following formula:

$$m = 2595\log_{10}(1 + \frac{f_{Hz}}{700}) \tag{4.2}$$

where mel is the scale of pitch.

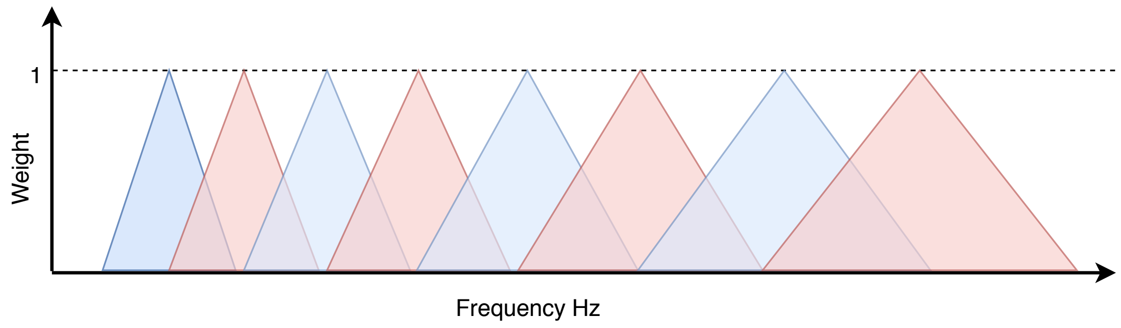


***Figure 4.2.*** *The triangular mel filters.*

In order to calculate the log mel energies, the speech signal is divided into short frames of $25$ ms using an overlap of $80$%. After this process, we calculate the short-time Fourier transform (STFT) of each frame to convert the time domain speech signal to the frequency domain. Next, the square of the absolute value of the STFT is computed to obtain the power spectrum. Then, the mel energies are obtained by filtering the power spectrum using a filterbank (e.g., triangular filters shown in Fig. 4.2). In order to cover the whole frequency range and to obtain fine resolution, a filterbank with $20$, $40$ or $80$ bands is typically used. In this thesis, $80$ mel bands are used. Finally, by taking the logarithm of the mel-band energies we obtain the log mel energies as acoustic features for WaveNet. The feature extraction process is shown in Fig. 4.3.

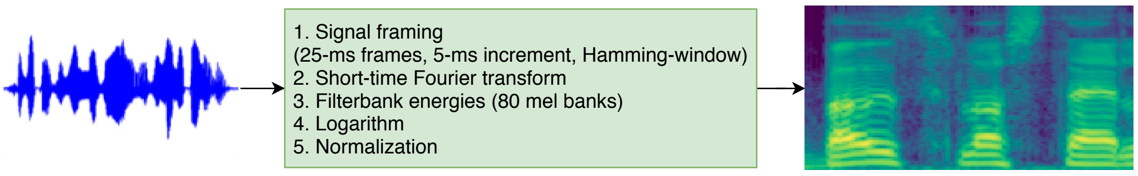

1. Signal framing
(25-ms frames, 5-ms increment, Hamming-window)
2. Short-time Fourier transform
3. Filterbank energies (80 mel banks)
4. Logarithm
5. Normalization

***Figure 4.3.*** *The process of mel-spectrogram calculation.*

### 4.2.2 $\mu$-law Transformation

In addition to the feature extraction process, we also need to apply another transformation to make input speech suitable for the network. Typically, digital time domain waveforms of the speech signal are represented using $16$-bit integer values, hence WaveNet has to predict probabilities of $65,536$ levels for each sample to model all possible signal values. This would result in an excessively complex network. This problem can be avoided by applying the *$\mu$-law transformation* to quantize the input speech to a 8-bit form before feeding to WaveNet. The $\mu$-law transformation is defined as

$$f(x_t) = sign(x_t) \times \frac{\ln 1 + \mu|x_t|}{\ln 1 + \mu} \tag{4.3}$$

where $\mu = 255$ and $-1 < x_t < 1$. This quantization allows the softmax layer to predict the probability distribution for only $256$ possible values, and also present a proper representation of the speech signal to WaveNet. Moreover, the reconstructed quantized speech is sufficiently similar to the original one, as stated in [8].

## 4.3 WaveNet Architecture

In this study, we train WaveNet by conditioning it on mel-spectrogram of the speech signal of a particular speaking style (i.e. normal style) in order to generate speech of a desired style (i.e. Lombard style). We also use thousands of utterances spoken by different speakers as training examples and adjust hyperparameters to train the WaveNet model.

As described in Section 3.4, WaveNet is composed of several stacked residual blocks including dilated causal convolution, gated activation function, residual and skip connections. For this thesis we design $30$ residual blocks in such a way that dilation factor of dilatation causal convolution is reset after 10 layers. Thus, the dilation pattern $1$,$2$,$4$,...,$512$ is repeated three times to form $30$ dilated causal convolution layers. Moreover, the number of channels for the residual block and skip connection is $512$ and $256$, respectively. For the loss function and optimizer, we utilize cross entropy and Adam optimizer with a fixed learning rate of $0.001$.

Before training the model, we choose short segments of the quantized speech signals $x$ and their corresponding mel-spectrograms $h$ as inputs to WaveNet instead of the whole-length signals due to memory restriction and longer computation time involved in the latter case. These segments are then converted to an one-hot representation with $L$ rows (i.e., the number of quantization levels) and $T$ columns (i.e., the timesteps). After this process, it is important to perform the time-resolution adjustment before feeding the data to WaveNet. Therefore, we add an upsample layer to the network so that when the mel-spectrogram is passed through it, its length becomes equal to that of the input segment.

In the training phase, one batch containing the one-hot representation and acoustic feature is fed into WaveNet at every training epoch. Each residual block provides residual and skip connections as output so that residual connection acts as the input to the next block and the skip connection is stored. When the process is done in the last residual block, skip connections are merged in order to pass into the WaveNet output. Finally, the softmax layer predicts the probability distribution for the next sample in the sequence. It should be noted that the WaveNet output has the same size as the input segment.

## 4.4 Speech Generation

Speech generation is carried out by using the trained WaveNet model. As previously explained, WaveNet is trained using plenty of speech data in normal style aiming at generating speech in Lombard style. Once the learning process is done and the trained WaveNet is capable of modelling prosodic features of speech, we store the WaveNet parameters and continue with the generation step.

Speech generation in WaveNet is a sequential process so that the sample $x_{t+1}$ is predicted using a sequence of the predicted past samples $x_1, x_2, ..., x_t$. We start generating speech in target style using an initial input value (e.g., $128$) and acoustic features of the test speech signal. First, we prepare the one-hot representation of the initial input, and then we upsample the mel-spectrogram of the test speech signal in order to feed WaveNet. The softmax function in the output layer is used to output conditional probability distribution of a sample for $L$ possible classes. A discrete value with the maximum likelihood from the distribution is selected as a new sample. This new sample is appended at the output vector and its one-hot representation with acoustic features of the test signal is fed back into the WaveNet to generate the next sample. We continue this process until the test utterance has been generated over its whole length. Fig. 4.4 depicts the speech generation process.

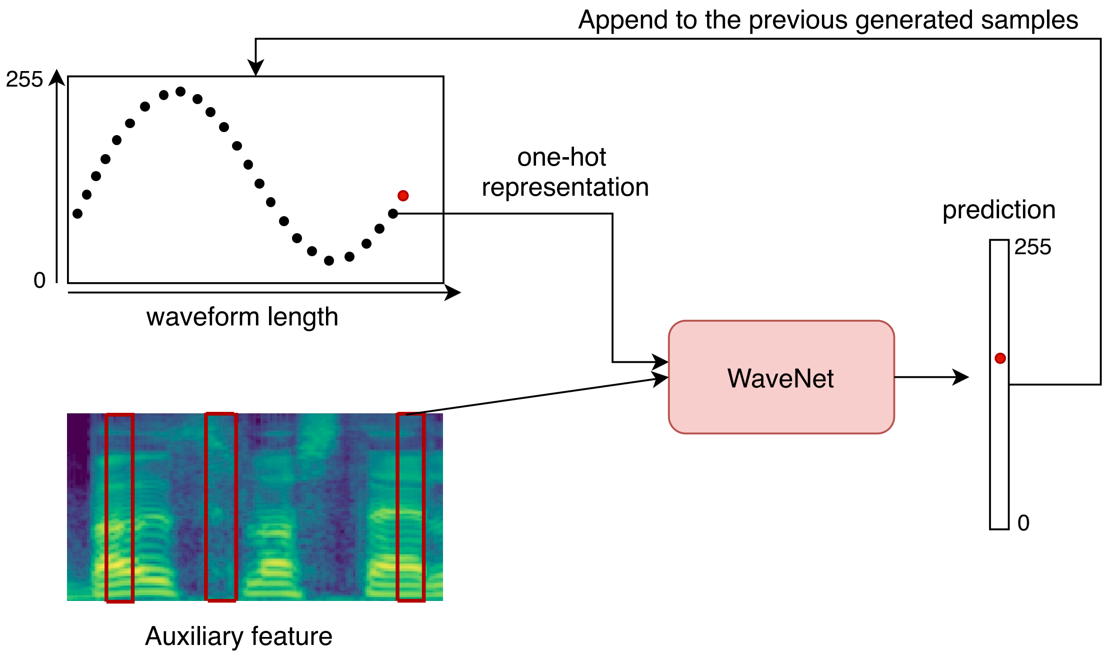


***Figure 4.4.*** *The process of speech waveform generation. After predicting the sample, it is fed back to the network as input to predict sequentially the next sample.*

The system development and speech generation are done in Python. In addition, the Pytorch library——an advanced machine learning library written in Python——is used for implementing and training the WaveNet model.

# 5 EXPERIMENTS

In this work, we aim to train the WaveNet model using both a large amount of normal style data from multiple speakers, and a small amount of normal and Lombard styles data from single-speaker to study how well WaveNet is capable of generating utterances of two speaking styles (normal and Lombard). Therefore, in this chapter we explain the procedures used to evaluate the performance of WaveNet and present the speech databases used in this thesis. We then describe in detail all the experiments that we have carried out to train and test the WaveNet model.

## 5.1 Databases

Two speech databases were used to train and evaluate WaveNet. These databases were the CMU Arctic speech database [87] and the Hurricane Challenge speech database [88]. As previously explained, the data was split to the training, validation and test sets. For this purpose, we used both the CMU Arctic and Hurricane databases for training and validation sets while the test set contained only speech from the Hurricane database.

**The CMU Arctic database** The CMU Arctic speech database was collected at the Language Technologies Institute at Carnegie Mellon University. These databases were originally developed for TTS. The CMU Arctic database contains seven different databases so that each database includes nearly 1132 phonetically balanced English utterances that are approximately $3$ s long. Each utterance is recorded from a single speaker in a specified style. The recordings have been conducted in a soundproof booth using a sampling rate of $16$ kHz. The seven CMU Arctic databases are identified by the initials of the corresponding speaker. The databases are as follows: BDL (an US English male speaker), RMS (an US English male speaker), SLT (an US English female speaker), CLB (an US English female speaker), KSP (an Indian English male speaker), AWB (a Scottish English male speaker) and JMK (an US English male speaker of Canadian ascent). The CMU Arctic databases are described in Table 5.1.

**The Hurricane Challenge speech database** The Hurricane Challenge database was collected using 72 phonetically balanced lists of 10 sentences of the Harvard corpus [89]. The data was spoken by a male British professional talker named Nick, who produced the entire set of the 720 Harvard utterances. Therefore, this data is referred to as "the Nick data" in this thesis. The Nick data was recorded both in normal and Lombard styles

***Table 5.1.*** *Details of CMU Arctic databases*

| Database | Speaker | # utterances | Duration(approx.) |
|---|---|---|---|
| BDL | 1 US male | 1132 | 56 min. |
| RMS | 1 US male | 1132 | 56 min. |
| SLT | 1 US female | 1132 | 56 min. |
| CLB | 1 US female | 1132 | 56 min. |
| KSP | 1 Indian male | 1132 | 56 min. |
| AWB | 1 Scottish male | 1138 | 57 min. |
| JMK | 1 US male | 1132 | 55 min. |

***Table 5.2.*** *Details of Nick data*

| Database | Styles | # utterances | Duration(approx.) |
|---|---|---|---|
| Nick data | Normal | 2544 | 110 min. |
| Nick data | Lombard | 720 | 35 min. |

in quiet and in the presence of noise. The sampling frequency was $16$ kHz. In total, there are $2544$ utterances of normal style and $720$ utterances of Lombard style in the Nick data. Table 5.2 shows the summary of the Nick data.

From now on, we will refer to the five male speakers of CMU Arctic data as *CMU Arctic 1*, all seven speakers of CMU Arctic data as *CMU Arctic 2*, normal speech of Nick data as *Nick 1*, both normal and Lombard styles of Nick data as *Nick 2*.

## 5.2 Experimental details

In order to perform our experiments, we decided to first find a baseline configuration for all experiments. To do this, we started the first experiment using Nick 1 to study how different elements such as the input length, receptive field size, number of dilated causal convolutions and batch size influence the performance of the WaveNet model in generating speech waveforms. Thus, we trained and tested the WaveNet model (see Fig. 3.12) several times with various configurations. Table 5.3 shows the obtained system configuration after several tests. Training WaveNet with the setting shown in Table 5.3 gave a lower loss value in the training and validation phases. This is because we used more dilated causal convolution layers, a longer segment length and a shorter hop-size in this configuration. Decreasing the hop-size results in a higher frame rate for the mel-spectrogram, thus we can extract more information from the speech signal, and the model can learn better the underlying structure of the signal. Moreover, for the shorter segment lengths, we used zero-padding to fill the receptive field which makes the beginnings of sequences less reliable for training. Therefore, by increasing the training segment length the model sees a larger proportion of valid signals in the training phase.

***Table 5.3.*** *System configuration for training WaveNet in the final experiments.*

| | |
|---|---|
| Filter length | 3 |
| Batch size | 1 |
| epoch | 150 |
| # residual blocks | 30 |
| dilation pattern | 1,2,4,8,...,512 |
| loss function | categorical crossentropy |
| Optimizer | Adam |
| Learning rate | 0.001 |

Since the configuration shown in Table 5.3 indicated a reasonable computation time and acceptable performance in predicting speech samples, we selected it as a baseline configuration for running all experiments in order to generate speech in normal and Lombard styles. The experiments carried out in this thesis are:

- **Experiment 1.** In this experiment, 2444 utterances of Nick 1 (1 hour and 45 minutes of normal style) were used to train and evaluate WaveNet so that training set consisted of 2344 speech examples and validation set contained 100 speech files.
- **Experiment 2.** 5660 speech signals of CMU Arctic 1 (4 hours and 40 minutes of normal style) were assigned for training and validation sets.
- **Experiment 3.** This experiment included 7930 utterances from CMU Arctic 2 (6 hours and 36 minutes of normal style) from which we used 7830 speech samples for training and 100 samples for validation.
- **Experiment 4.** We used 3064 utterances from Nick 2 (1 hour and 45 minutes of normal style and 25 minutes of Lombard style) in such a way that the training set consisted of 2964 utterances and the validation set contained 100 utterances.

We used the common test set for all the four experiments. The test set contains 100 utterances of Nick data in normal style and 100 utterances of Nick data in Lombard style as unseen data for generating speech waveforms. These experiments enabled us to study the performance of WaveNet when trained with the larger amount of normal style data, and smaller amount of normal and Lombard styles data regarding generating speech waveforms of the target style (Lombard). The description of four experiments is summarized in Table 5.4.

***Table 5.4.*** *Details of four experiments used for training and testing WaveNet.*

| Experiments | Database | styles | Training set utterances | Test set utterances |
|---|---|---|---|---|
| 1 | Nick 1 | Normal | 2444 | 1. 100 utterances of Nick data in normal style<br>2. 100 utterances of Nick data in Lombard style |
| 2 | CMU Arctic 1 | Normal | 5660 | |
| 3 | CMU Arctic 2 | Normal | 7930 | |
| 4 | Nick 2 | Normal and Lombard | 3064 | |

All these experiments were run on GPU in Triton, the high-performance computing cluster provided by Aalto University. We trained our model, conditioned on the mel-spectrogram, for each experiment using the configuration presented in Table 5.3 and generated speech in both normal and Lombard styles. We generated 100 speech waveforms in Lombard style and 100 speech waveforms in normal style for all the experiments. The reason of generating speech in normal style was to study the performance of WaveNet in different system configurations and to find fixed parameters for further training. Moreover, by comparing the generated normal speech utterances to Lombard speech we could analyze whether WaveNet is capable of mimicking the Lombard effect. Finally, randomly selected samples were taken from the generated data to be used in the two listening tests described in section 5.3.

## 5.3 Evaluation Procedure

In the assessment of speech generation methods such as the WaveNet vocoder, subjective evaluations are widely used. In this work, we apply two different subjective listening tests: the speaking style similarity test and mean opinion score (MOS) test. In the former, we ask subjects (untrained listeners) to evaluate the speaking style similarity between natural Lombard speech and their WaveNet-generated counterparts. In the latter, we evaluate the overall quality and naturalness of the WaveNet-generated speech utterances.

**Speaking Style Similarity Test.** We train WaveNet using speech data of normal style (the source style) and generate speech in Lombard style (the target style) without changing the linguistic contents of speech. Therefore, it is important to evaluate the similarity between the generated Lombard utterances and their natural counterparts (i.e both natural normal speech and natural Lombard speech). Since there are several similarity tests that have been used in the literature [90], we decided to employ the evaluation setup used in [11] to measure style similarity. In a speaking style similarity test, a pair of utterances is used in the assessment: one is a natural speech reference in either normal or Lombard style and the other one is the generated speech sample. The listener is asked to answer the following question “Do you think that these two samples have been produced using the same speaking style? Some of the samples may sound slightly distorted. Please try to ignore the distortion and concentrate on identifying the speaking style." The listener chooses an answer from the following four options:

- Speaking style sounds the same, I’m absolutely sure
- Speaking style sounds the same, but I’m not completely sure
- Speaking style sounds different, but I’m not completely sure
- Speaking styles sounds different, I’m absolutely sure

**Quality and Naturalness Test.** The mean opinion score (MOS) test is the most popular listening test for evaluating the quality and naturalness of speech, for example, in TTS and in speech coding. This test aims to describe how much the quality and/or naturalness of a speech utterance is affected by the underlying processing (e.g. speech coding or generation). In the MOS test conducted in the thesis, the listeners were asked to rate the quality of the generated speech sample using the following scale:

- "5" for excellent
- "4" for good
- "3" for fair
- "2" for poor
- "1" for bad

The MOS was finally calculated by taking the mean value of the answers.

In speaking style similarity test, we randomly chose 16 pairs utterances from all experiments described in section 5.2 so that first utterance is either natural normal reference or natural Lombard reference and second utterance is WaveNet-generated speech waveforms in normal and Lombard styles. In MOS test, 16 utterances of WaveNet-generated speech signals were randomly selected from all experiments (8 normal style and 8 Lombard style). In addition, we also chose two utterances from natural normal reference and two utterances from natural Lombard reference. Totally, 20 utterances were used in MOS test.

In order to conducted the listening tests, we made two simple HTML webpages for both tests. In a non-English speaking country (Finland), it is difficult to find native English speakers. Therefore, we recruited listeners among students of Aalto University who have been studying at least 5 years in a program whose language of instruction is English. The listening tests were conducted by 20 listeners who were situated in a quite office room using high-quality headphones in the evaluations. The subjects rated first the style similarity of 16 pairs utterances after that the quality and naturalness of 20 utterances were rated. The obtained results for speaking style similarity and MOS tests are presented in Chapter 6.

# 6 RESULTS

This chapter presents the results of the subjective evaluations. The results of the speaking style similarity test are described first in section 6.1 after which the results of the quality and naturalness evaluation are described in section 6.2.

## 6.1 Speaking Style Similarity Test

The results of speaking style similarity are shown in Fig 6.1. The plots depict the results of the style similarity test between the WaveNet-generated utterances and the natural reference utterances in Lombard style (left) and in normal style (right). The four databases that were used in the WaveNet training are shown in different rows. In the ideal case, the listener should distinguish the generated normal speaking style samples from the corresponding reference samples of Lombard style. Moreover, in the ideal case, the WaveNet-generated Lombard samples should be distinguished from the corresponding natural samples of normal style. These ideal cases indeed manifest themselves in Fig. 6.1 by the blue bars which show high values.

Fig 6.2 shows the performance of WaveNet trained with four databases (Nick 1, Nick 2, CMU Arctic 1 and CMU Arctic 2) regarding speaking style similarity for WaveNet-generated Lombard speech. Left figure presents the style similarity between WaveNet-generated utterances in Lombard style and natural normal reference. Right figure represents style similarity between WaveNet-generated utterances in Lombard style and natural Lombard reference. As can be seen, WaveNet trained with Nick 2 performed better than other three databases in generating Lombard speech signals. In addition, non-parametric Mann-Whitney *U* test [91] with Bonferroni correction was conducted to

***Table 6.1.*** *Mann-Whitney U test p-values with Bonferroni correction for generated Lombard speech using different databases. The significance level is* $0.05$ *and significant if* $P < 0.05$.

| | Nick 1 | Nick 2 | CMU Arctic 1 | CMU Arctic 2 |
|---|---|---|---|---|
| Nick 1 | – | **0.000** | 0.452 | 1.0 |
| Nick 2 | **0.011** | – | **0.001** | **0.000** |
| CMU Arctic 1 | 0.452 | **0.001** | – | 1.0 |
| CMU Arctic 2 | 1.0 | **0.016** | 1.0 | – |

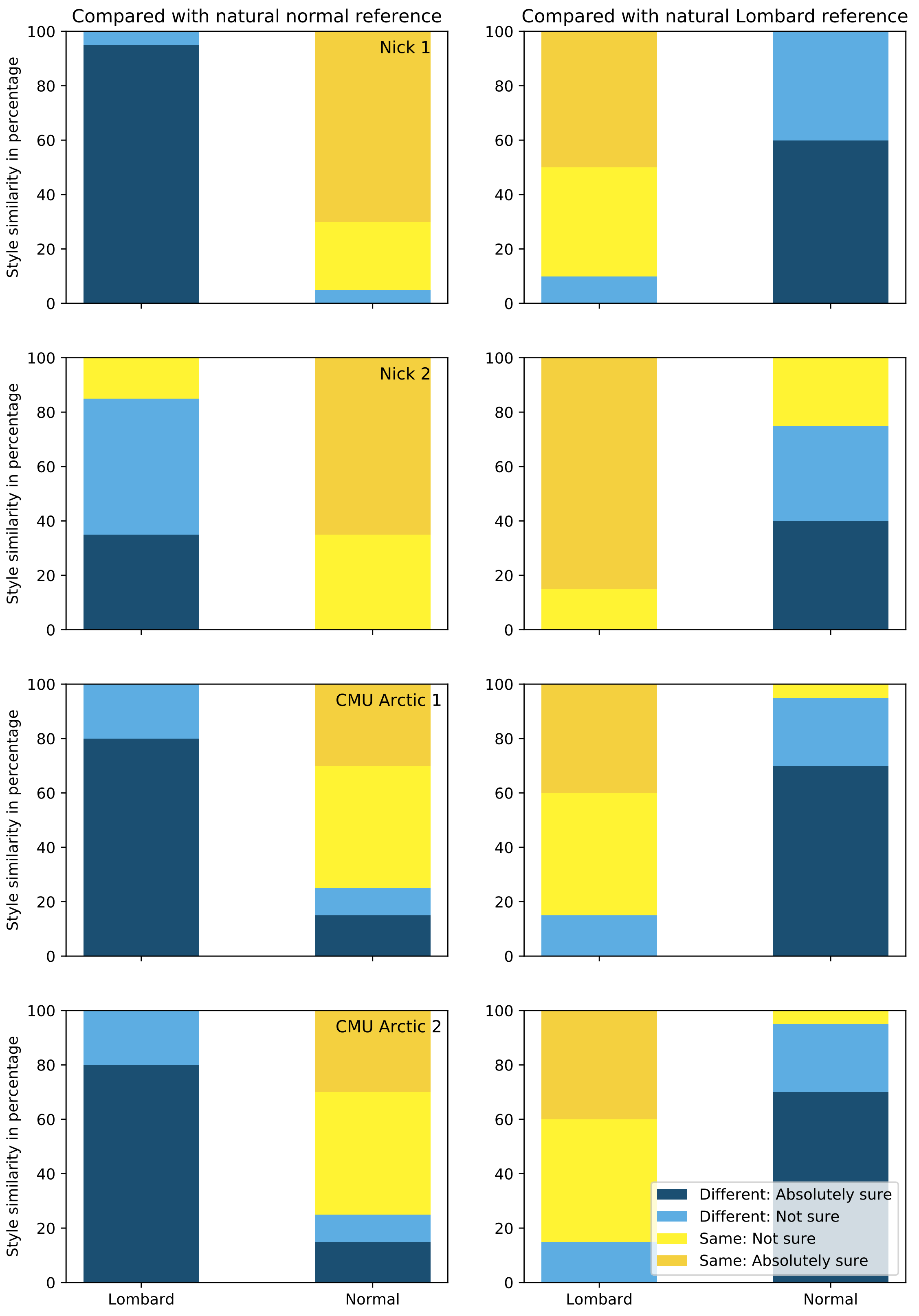


***Figure 6.1.*** *The speaking style similarity results for WaveNet trained with Nick 1 (first row), Nick 2 (second row), CMU Arctic 1 (third row) and CMU Arctic 2 (fourth row). The Y-axis shows the style similarity in percentage. The left column shows the WaveNet-generated utterances in Lombard style (left bars) and in normal style (right bars) compared to the natural normal reference. The right column shows the WaveNet-generated utterances in Lombard style (left bars) and in normal style (right bars) compared to the natural Lombard reference.*

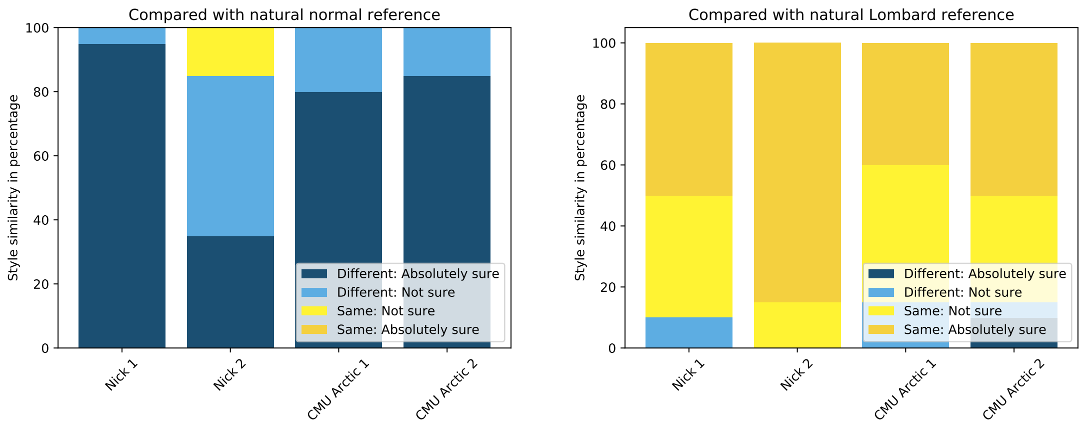


***Figure 6.2.*** *Results of the style similarity test for WaveNet-generated utterances in Lombard style. The X-axis presents results for WaveNet trained with Nick 1, Nick 2, CMU Arctic 1 and CMU Arctic 2. Y-axis shows the style similarity in percentage. Left figure represents the results of style similarity between WaveNet-generated utterances in Lombard style and natural normal reference. right figure represents the results of style similarity between WaveNet-generated utterances in Lombard style and natural Lombard reference.*

analyze whether the database used showed a statistically significant effect in similarity. The results in Table 6.1 indicate that the difference between the Nick 2 and the other databases is significant. On the other hand, Nick 1, CMU Arctic 1 and 2 are not significantly different.

## 6.2 Quality and Naturalness Test

Fig. 6.3 depicts the MOS results for generated speech of Lombard style. This plot shows the mean scores with $95\%$ confidence intervals for all experiments.

It can be seen that the WaveNet vocoder trained with the CMU Arctic 1, CMU Arctic 2 and Nick 2 received high ratings, while the WaveNet trained with Nick 1 clearly failed to generate speech of decent quality and naturalness in Lombard style. This is because Nick 1 has less data per speaker in comparison to the other databases. Consequently, WaveNet training could not see sufficient variations in the data to generate high-quality speech in Lombard style. Instead, the model training using the CMU Arctic 2 results in higher quality and naturalness compared to the other databases. The main reason for this is that CMU Arctic 2 includes a large amount of data which shows different variation due to, for example, multiple speakers, accents and genders. In addition, Table 6.2 shows non-parametric Mann-Whitney *U* test with Bonferroni correction for quality of the generated Lombard utterances. The *U* test indicates that CMU Arctic 2 and natural Lombard reference (Ref) are not significantly different. Likewise, there is no statistically significant differences between CMU Arctic 1 and Nick 2, however, the generated Lombard speech from CMU Arctic 1 received slightly higher rating as also indicated by Fig. 6.3.

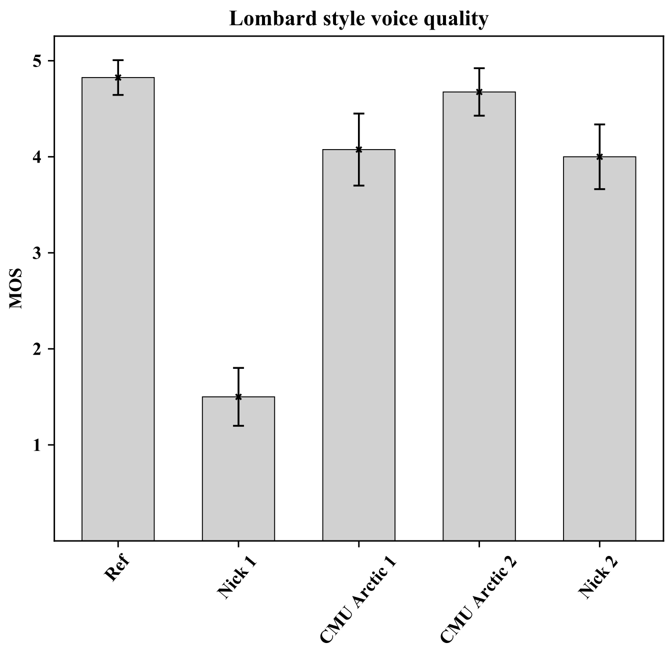


***Figure 6.3.*** *Mean opinion score results for Lombard style speech quality. In X-axis, "Ref" represents natural Lombard reference and the rest corresponds to databases used for training the WaveNet model. Y-axis indicates the mean scores with $95\%$ confidence interval for all experiments.*

***Table 6.2.*** *Mann-Whitney U test p-values with Bonferroni correction for generated Lombard speech using different databases. The significance level is $0.05$ and significant if $P < 0.05$.*

| | ref | Nick 1 | Nick 2 | CMU Arctic 1 | CMU Arctic 2 |
|---|---|---|---|---|---|
| ref | – | **0.000** | **0.001** | **0.005** | 1.0 |
| Nick 1 | **0.000** | – | **0.000** | **0.000** | **0.000** |
| Nick 2 | **0.001** | **0.000** | – | 1.0 | **0.001** |
| CMU Arctic 1 | **0.005** | **0.000** | 1.0 | – | **0.004** |
| CMU Arctic 2 | 1.0 | **0.000** | **0.001** | **0.004** | – |

The results of the MOS test for generated speech of normal style shown in Fig. 6.4 demonstrate that all the databases achieved closer mean scores to the natural normal reference (Ref) mean score. Among the databases, it was observed (see Fig. 6.4) that training WaveNet with Nick 1 and Nick 2 seems to perform better than other two databases (CMU Arctic 1 and 2). Besides, the results of *U* test for quality of the generated normal style speech shown in the Table 6.3 confirms that the differences between all databases and natural normal reference are not statistically significant.

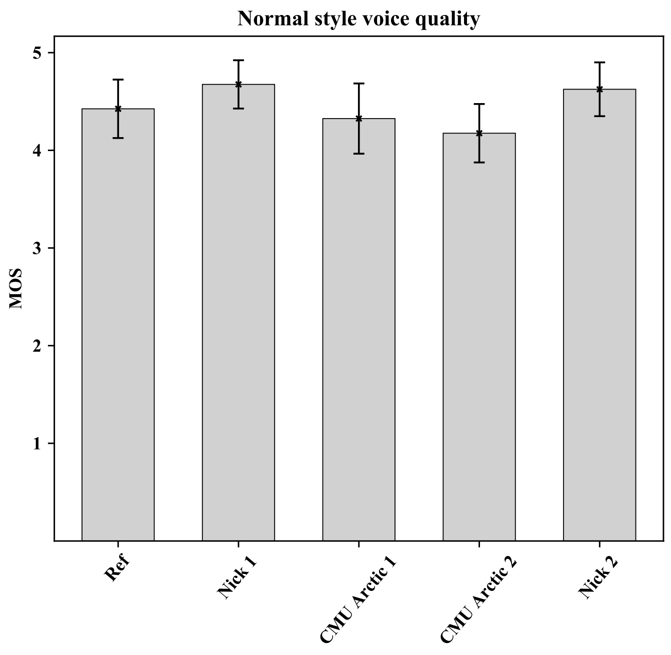


***Figure 6.4.*** *Mean opinion score results for normal style speech quality. In X-axis, "Ref" represents natural normal reference and the rest corresponds to databases used for training the WaveNet model. Y-axis indicates the mean scores with $95\%$ confidence interval for all experiments.*

***Table 6.3.*** *Mann-Whitney U test p-values with Bonferroni correction for generated normal speech using different databases. The significance level is $0.05$ and significant if $P < 0.05$.*

| | ref | Nick 1 | Nick 2 | CMU Arctic 1 | CMU Arctic 2 |
|---|---|---|---|---|---|
| ref | – | 0.611 | 1.0 | 1.0 | 0.733 |
| Nick 1 | 0.611 | – | 1.0 | 0.341 | **0.003** |
| Nick 2 | 1.0 | 1.0 | – | 1.0 | **0.011** |
| CMU Arctic 1 | 1.0 | 0.341 | 0.7 | – | 1.0 |
| CMU Arctic 2 | 0.733 | **0.003** | **0.011** | 1.0 | – |

# 7 CONCLUSIONS

In this thesis, speech generation related to a specific attribute of natural speech communication, speaking style, is studied. The study focuses on the WaveNet model——a recently developed advanced neural vocoder——which was trained using speech spoken in a source style (normal) to generate speech of a target style (Lombard). Training of WaveNet was conducted by conditioning the model on a time-frequency representation (the mel-spectrogram) of the input speech.

We trained the WaveNet model using four different speech databases, namely, Nick 1 (1 hour and 45 minutes of normal style), Nick data (2 hours and 10 minutes of normal and Lombard styles), CMU Arctic 1 (4 hours and 40 minutes of normal style) and CMU Arctic 2 (6 hours and 36 minutes of normal style). These databases consist of corpora that include both large amounts data from multiple speakers (CMU Arctic 1, CMU Arctic 2) and also sets with small amounts of speech from single speakers (Nick 1, Nick data). These databases enabled us to study the performance of WaveNet when trained with the larger amount of normal style data, and smaller amount of normal and Lombard styles data regarding generating speech waveforms of the target style (Lombard).

Two subjective evaluations (a speaking style similarity test, and a MOS test) were conducted to evaluate the performance of the WaveNet model for each of the database. In the speaking style similarity test, the style similarity between the natural Lombard reference and WaveNet-generated speech waveforms were compared. In the MOS test, we assessed the quality and naturalness of the WaveNet-generated speech signals. When the WaveNet model was trained using a small amount of speech consisting of normal and Lombard styles of a single speaker (i.e. the Nick data), we found a large style similarity between the WaveNet-generated Lombard signals and their natural Lombard references. However, the corresponding similarity was clearly smaller when the WaveNet model was trained using speech data from the other three databases consisting of normal style of multiple speakers. On the other hand, when WaveNet was trained using a large amount of normal style data from multiple speakers (i.e., CMU Arctic 2) we found that the quality and naturalness of WaveNet-generated Lombard speech signals were superior compared to the WaveNet-generated speech signals from the other three databases.

In summary, the study shows that WaveNet is an effective tool to generate speech of a given target style (i.e. Lombard in the case of the current study) using the mel-spectrogram. However, the training strategy of WaveNet affects both the style similarity between the generated speech signals and their natural reference as well as the quality and naturalness of the generated signals. In particular, it was observed that WaveNet trained using a small amount of Lombard speech of a single speaker gave better results in terms of speaking style similarity than using a large amount of normal speech from multiple talkers. On the other hand, the model training using a large amount of data in normal style improved WaveNet's performance in terms of speech quality and naturalness. Overall, we can conclude that the WaveNet model trained on speech of normal style is capable of generating speech waveforms of Lombard style when the training data has some speech signals in Lombard style.

# REFERENCES

[1] Zen, H., Tokuda, K. and Black, A. W. Statistical parametric speech synthesis. *Speech Communication* (2009), pp. 1039–1064.

[2] Mohammadi, S. H. and Kain, A. An overview of voice conversion systems. *Speech Communication* (2017), pp. 65–82.

[3] Seshadri, S., Juvela, L., Räsänen, O. and Alku, P. Vocal Effort Based Speaking Style Conversion Using Vocoder Features and Parallel Learning. *IEEE Access* (2019), pp. 17230-17246.

[4] Seshadri, S., Juvela, L., Yamagishi, J., Räsänen, O. and Alku, P. Cycle-consistent adversarial networks for non-parallel vocal effort based speaking style conversion. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), pp. 6835–6839.

[5] López, A., Seshadri, S., Juvela, L., Räsänen, O. and Alku, P. Speaking Style Conversion from Normal to Lombard Speech Using a Glottal Vocoder and Bayesian GMMs. *Proceedings of Interspeech 2017, Interspeech: Annual Conference of the International Speech Communication Association* (2017), pp. 1363-1367.

[6] Wang, Y., Skerry-Ryan, R., Stanton, D., Wu, Y., Weiss, R. J., Jaitly, N., Yang, Z., Xiao, Y., Chen, Z., Bengio, S. et al. Tacotron: Towards end-to-end speech synthesis. *Interspeech* (2017).

[7] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. Generative adversarial nets. *Advances in Neural Information Processing Systems* (2014), pp. 2672–2680.

[8] Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *ArXiv* (2016).

[9] Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., Chen, Z., Zhang, Y., Wang, Y., Skerrv-Ryan, R. et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 4779–4783.

[10] Lombard, E. Le signe de l'elevation de la voix. *Ann. Mal. de L'Oreille et du Larynx* (1911), pp. 101–119.

[11] Bollepalli, B., Juvela, L., Airaksinen, M., Valentini-Botinhao, C. and Alku, P. Normal-to-Lombard Adaptation of Speech Synthesis Using Long Short-Term Memory Recurrent Neural Networks. *Speech Communication* (2019).

[12] KAIN, A. High resolution voice transformation. PhD thesis. 2001.

[13] Kuwabara, H. and Sagisak, Y. Acoustic characteristics of speaker individuality: Control and conversion. *Speech communication* (1995), pp. 165–173.

[14] Lu, Y. and Cooke, M. Speech production modifications produced by competing talkers, babble, and stationary noise. *The Journal of the Acoustical Society of America* (2008), pp. 3261–3275.

[15] Hansen, J. H. L. Analysis and compensation of speech under stress and noise for environmental robustness in speech recognition. *Speech Communication* (1996), pp. 151-173.

[16] Summers, V., Pisoni, D., Bernacki, R., Pedlow, R. and Stokes, M. Effects of noise on speech production: Acoustic and perceptual analyses. *The Journal of the Acoustical Society of America* (1988), pp. 917-28.

[17] Junqua, J.-C. The Lombard reflex and its role on human listeners and automatic speech recognizers. *The Journal of the Acoustical Society of America* (1993), pp. 510–524.

[18] Garnier, M., Bailly, L., Dohen, M., Welby, P. and Lœvenbruck, H. An acoustic and articulatory study of Lombard speech: Global effects on the utterance. *Ninth International Conference on Spoken Language Processing* (2006).

[19] Cooke, M., King, S., Garnier, M. and Aubanel, V. The listening talker: A review of human and algorithmic context-induced modifications of speech. *Computer Speech & Language* (2014), pp. 543–571.

[20] Kawahara, H., Masuda-Katsuse, I. and De Cheveigne, A. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based F0 extraction: Possible role of a repetitive structure in sounds. *Speech communication* (1999), pp. 187–207.

[21] Airaksinen, M., Bollepalli, B., Juvela, L., Wu, Z., King, S. and Alku, P. GlottDNN - A Full-Band Glottal Vocoder for Statistical Parametric Speech Synthesis. *Interspeech* (2016).

[22] Raitio, T., Suni, A., Yamagishi, J., Pulakka, H., Nurminen, J., Vainio, M. and Alku, P. HMM-Based Speech Synthesis Utilizing Glottal Inverse Filtering. *Audio, Speech, and Language Processing, IEEE Transactions on* (2011), pp. 153–165.

[23] Tokuda, K., Kobayashi, T., Masuko, T. and Imai, S. Mel-generalized cepstral analysis-a unified approach to speech spectral estimation. *Third International Conference on Spoken Language Processing* (1994).

[24] Jin, Z., Finkelstein, A., Mysore, G. J. and Lu, J. FFTNet: A real-time speaker-dependent neural vocoder. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 2251–2255.

[25] Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A. van den, Dieleman, S. and Kavukcuoglu, K. Efficient Neural Audio Synthesis. *Proceedings of the 35th International Conference on Machine Learning* (2018), pp. 2410–2419.

[26] Prenger, R., Valle, R. and Catanzaro, B. Waveglow: A Flow-based Generative Network for Speech Synthesis. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), pp. 3617-3621.

[27] Saito, Y., Ijima, Y., Nishida, K. and Takamichi, S. Non-parallel voice conversion using variational autoencoders conditioned by phonetic posteriorgrams and d-vectors. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 5274–5278.

[28] Tobing, P. L., Wu, Y.-C., Hayashi, T., Kobayashi, K. and Toda, T. Non-Parallel Voice Conversion with Cyclic Variational Autoencoder. *Proc. Interspeech 2019* (2019), pp. 674–678.

[29] Childers, D., Wu, K., Hicks, D. and Yegnanarayana, B. Voice conversion. *Speech Communication* (1989), pp. 147 - 158.

[30] Childers, D., Yegnanarayana, B. and Wu, K. Voice conversion: Factors responsible for quality. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (1985), pp. 748–751.

[31] Wahlster, W. *Verbmobil: foundations of speech-to-speech translation*. Springer Science & Business Media, 2013.

[32] Turk, O. and Schroder, M. Evaluation of Expressive Speech Synthesis With Voice Conversion and Copy Resynthesis Techniques. *IEEE Transactions on Audio, Speech, and Language Processing* (2010), pp. 965-973.

[33] Wu, Z. Spectral mapping for voice conversion. PhD thesis. 2015.

[34] Stylianou, Y. Applying the harmonic plus noise model in concatenative speech synthesis. *IEEE Transactions on Speech and Audio Processing* (2001), pp. 21–29.

[35] Morise, M., Yokomori, F. and Ozawa, K. WORLD: A Vocoder-Based High-Quality Speech Synthesis System for Real-Time Applications. *IEICE Transactions on Information and Systems* E99.D (2016), pp. 1877-1884.

[36] Gray, R. Vector quantization. *IEEE Assp Magazine* (1984), pp. 4–29.

[37] Abe, M., Nakamura, S., Shikano, K. and Kuwabara, H. Voice conversion through vector quantization. *Journal of the Acoustical Society of Japan (E)* (1990), pp. 71–76.

[38] Toda, T., Black, A. W. and Tokuda, K. Voice Conversion Based on Maximum-Likelihood Estimation of Spectral Parameter Trajectory. *IEEE Transactions on Audio, Speech, and Language Processing* (2007), pp. 2222-2235.

[39] Kain, A. and Macon, M. W. Spectral voice conversion for text-to-speech synthesis. *IEEE International Conference on Acoustics, Speech and Signal Processing, (ICASSP)* (1998), pp. 285-288.

[40] Stylianou, Y., Cappé, O. and Moulines, E. Continuous probabilistic transform for voice conversion. *IEEE Transactions on Speech and Audio Processing* (1998), pp. 131–142.

[41] Hunt, A. J. and Black, A. W. Unit selection in a concatenative speech synthesis system using a large speech database. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (1996), pp. 373-376.

[42] Chen, L.-H., Ling, Z.-H., Song, Y. and Dai, L.-R. Joint spectral distribution modeling using restricted boltzmann machines for voice conversion. *Interspeech* (2013).

[43] Wu, Z., Chng, E. S. and Li, H. Conditional restricted boltzmann machine for voice conversion. *2013 IEEE China Summit and International Conference on Signal and Information Processing* (2013), pp. 104–108.

[44] Nakashika, T., Takiguchi, T. and Ariki, Y. Voice conversion using speaker-dependent conditional restricted Boltzmann machine. *EURASIP Journal on Audio, Speech, and Music Processing* (2015), pp. 1–12.

[45] Nakashika, T., Takashima, R., Takiguchi, T. and Ariki, Y. Voice conversion in high-order eigen space using deep belief nets. *Interspeech* (2013), pp. 369-372.

[46] Sun, L., Kang, S., Li, K. and Meng, H. M. Voice conversion using deep Bidirectional Long Short-Term Memory based Recurrent Neural Networks. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2015), pp. 4869-4873.

[47] Inanoglu, Z. and Young, S. Data-driven Emotion Conversion In Spoken English. *Speech Communication* (2009), pp. 268-283.

[48] Inanoglu, Z. and Young, S. A system for transforming the emotion in speech: Combining data-driven conversion techniques for prosody and voice quality. *Eighth Annual Conference of the International Speech Communication Association* (2007).

[49] Picheny, M. A., Durlach, N. I. and Braida, L. D. Speaking clearly for the hard of hearing II: Acoustic characteristics of clear and conversational speech. *Journal of Speech, Language, and Hearing Research* (1986), pp. 434–446.

[50] Uchanski, R. M., Choi, S. S., Braida, L. D., Reed, C. M. and Durlach, N. I. Speaking clearly for the hard of hearing IV: Further studies of the role of speaking rate. *Journal of Speech, Language, and Hearing Research* (1996), pp. 494–509.

[51] Vydana, H. K., Kadiri, S. R. and Vuppala, A. K. Vowel-based non-uniform prosody modification for emotion conversion. *Circuits, Systems, and Signal Processing* (2016), pp. 1643–1663.

[52] Erro, D., Sainz, I., Navas, E. and Hernaez, I. Harmonics plus noise model based vocoder for statistical parametric speech synthesis. *IEEE Journal of Selected Topics in Signal Processing* (2013), pp. 184–194.

[53] Hu, Q., Stylianou, Y., Maia, R., Richmond, K., Yamagishi, J. and Latorre, J. An investigation of the application of dynamic sinusoidal models to statistical parametric speech synthesis. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2015).

[54] Haykin, S. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[55] Haykin, S. S. et al. *Neural networks and learning machines/Simon Haykin.* New York: Prentice Hall, 2009.

[56] Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* (1958), pp. 386.

[57] Glorot, X., Bordes, A. and Bengio, Y. Deep sparse rectifier neural networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 315–323.

[58] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[59] Werbos, P. J. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks* (1988), pp. 339–356.

[60] Hornik, K., Stinchcombe, M., White, H. et al. *Multilayer feedforward networks are universal approximators.* Elsevier, 1989.

[61] Cireşan, D. C., Meier, U., Gambardella, L. M. and Schmidhuber, J. Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation* (2010), pp. 3207–3220.

[62] Mohamed, A.-r., Dahl, G. E. and Hinton, G. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing* (2011), pp. 14–22.

[63] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations* (2014).

[64] Duchi, J., Hazan, E. and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* (2011), pp. 2121–2159.

[65] Zeiler, M. D. Adadelta: an adaptive learning rate method. *ArXiv* (2012).

[66] Tieleman, T. and Hinton, G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning* (2012), pp. 26–31.

[67] Nesterov, Y. E. A method for solving the convex programming problem with convergence rate O (1/kˆ 2). *Dokl. akad. nauk Sssr* (1983), pp. 543–547.

[68] Hinton, G. E., Osindero, S. and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural Computation* (2006), pp. 1527–1554.

[69] Bengio, Y. Deep learning of representations for unsupervised and transfer learning. *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*. 2012, pp. 17–36.

[70] Braverman, M. Poly-logarithmic independence fools bounded-depth boolean circuits. *Communications of the ACM* (2011), pp. 108–115.

[71] Delalleau, O. and Bengio, Y. Shallow vs. deep sum-product networks. *Advances in neural information processing systems* (2011), pp. 666–674.

[72] Bengio, Y., Delalleau, O. and Simard, C. Decision trees do not generalize to new variations. *Computational Intelligence* (2010), pp. 449–467.

[73] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* (2014), pp. 1929–1958.

[74] Wan, L., Zeiler, M., Zhang, S., Le Cun, Y. and Fergus, R. Regularization of Neural Networks Using Dropconnect. *International Conference on Machine Learning*. 2013, pp. 1058–1066.

[75] Dahl, G. E., Yu, D., Deng, L. and Acero, A. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing* (2011), pp. 30–42.

[76] Pascanu, R., Dauphin, Y. N., Ganguli, S. and Bengio, Y. On the saddle point problem for non-convex optimization. *ArXiv* (2014).

[77] Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B. and LeCun, Y. The loss surfaces of multilayer networks. *Artificial Intelligence and Statistics* (2015), pp. 192–204.

[78] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics* (1980), pp. 193–202.

[79] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* (1998), pp. 2278–2324.

[80] Le Cun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., Henderson, D., Howard, R. E. and Hubbard, W. Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine* (1989), pp. 41–46.

[81] Lecun, Y. and Bengio, Y. Convolutional networks for images, speech, and time-series. *The handbook of brain theory and neural networks*. MIT Press, 1995.

[82] Kleijn, W. B., Lim, F. S., Luebs, A., Skoglund, J., Stimberg, F., Wang, Q. and Walters, T. C. WaveNet based low rate speech coding. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 676–680.

[83] Kobayashi, K., Hayashi, T., Tamamori, A. and Toda, T. Statistical Voice Conversion with WaveNet-Based Waveform Generation. *Interspeech* (2017), pp. 1138-1142.

[84] Niwa, J., Yoshimura, T., Hashimoto, K., Oura, K., Nankaku, Y. and Tokuda, K. Statistical Voice Conversion Based on Wavenet. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 5289-5293.

[85] Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A. et al. Conditional image generation with pixelcnn decoders. *Advances in Neural Information Processing Systems* (2016), pp. 4790–4798.

[86] Boilard, J., Gournay, P. and Lefebvre, R. A Literature Review of WaveNet: Theory, Application, and Optimization. *Audio Engineering Society Convention 146*. 2019.

[87] Kominek, J. and Black, A. W. The CMU Arctic speech databases. *Fifth ISCA Workshop on Speech Synthesis*. 2004.

[88] Cooke, M., Mayo, C., Valentini-Botinhao, C. et al. Hurricane natural speech corpus. LISTA Consortium:(i) Language and Speech Laboratory, Universidad del Pais, 2013.

[89] Rothauser, E. IEEE recommended practice for speech quality measurements. *IEEE Trans. on Audio and Electroacoustics* (1969), pp. 225–246.

[90] Black, A. W. and Tokuda, K. The Blizzard Challenge-2005: Evaluating corpus-based speech synthesis on common datasets. *Ninth European Conference on Speech Communication and Technology* (2005).

[91] Rosenberg, A. and Ramabhadran, B. Bias and Statistical Significance in Evaluating Speech Synthesis with Mean Opinion Scores. *Interspeech 2017* (2017), pp. 3976–3980.