TÉCNICO
LISBOA

Voice Conversion with Deep Learning

Miguel Varela Ramos

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisor(s): Prof. Isabel Trancoso
Prof. Nuno Fonseca, IPLeiria

Examination Committee

Chairperson: Prof. João Sequeira
Supervisor: Prof. Nuno Fonseca
Member of the Committee: Prof. Mário Figueiredo

October 2016

To my loved ones...

Acknowledgments

I would like to start by thanking Prof. Isabel Trancoso and Prof. Nuno Fonseca for accepting the invitation for supervising this thesis and having embraced the topic with great passion. Their technical insights and suggestions helped me a lot throughout the development of my work and certainly improved the quality of this thesis.

For the technical part, I would like to express my deepest gratitude to Ramón Fernandez Astudillo for the incredible support and guidance provided, which has been essential to the development of most of the work developed. He has played a role of an unofficial supervisor for this thesis and I owe him most of my current machine learning technical skills. Without him, my progress on this thesis would have taken much longer, so I can never thank him enough.

I would like to acknowledge as well some people who had a relevant impact on my work and helped me make some breakthroughs. Some of them are Miguel Matos and Carlos Mendes who helped me in the most early stages of my work. Zhizheng Wu, Lifa Sun and Helen Meng, who have worked with state of the art voice conversion techniques, also helped me a lot by promptly being available to reply to my never ending emails filled with questions.

The members of INESC-ID and L^{2}F also deserve to be mentioned, for providing a welcoming and friendly work environment in which has been a great pleasure to work. In particular, I would like to thank António Lopes, who became a great friend, and played an important role on the development of the ongoing deep learning framework, used for all my experiments.

Last, but not least, and on a more personal note, I would like to thank my family and friends for the never ending support.

Resumo

As técnicas de conversão de voz têm como objectivo a modificação das características vocais de um indivíduo de forma a que este soe como outra pessoa. Nesta tese, o nosso principal foco é a componente de mapeamento espectral da conversão de voz. A maior parte das técnicas que antecedem este trabalho recorrem a um alinhamento das características vocais para conseguir converter as mesmas. Neste trabalho, é proposto um modelo de mapeamento de sequências com um mecanismo de atenção para permitir mapear sequências de tamanhos diferentes. De forma a estabelecer um termo de comparação, foi implementado um sistema de conversão de voz com um modelo intitulado *deep bidirectional long short-term memory recurrent neural network*, correspondente a resultados do estado da arte e que também recorre a um alinhamento das características vocais. As experiências realizadas revelam que o nosso modelo proposto atinge resultados superiores aos do actual estado da arte quando numa situação em que este tem acesso a informação das caracterśticas do alvo, embora tenha um baixo desempenho quando deprivado dessa informação. Ambos os modelos foram avaliados usando uma métrica objectiva e o desempenho do modelo proposto é discutido no decorrer do trabalho, sendo propostas algumas alternativas de resolução do problema de previsão do modelo.

Palavras-chave: conversão de voz, *deep learning*, modelos de atenção, mapeamento de sequências, redes neuronais recurrentes, processamento da fala

Abstract

Voice conversion techniques aim to modify a subject's voice characteristics in order to sound like someone else. In this thesis, we mainly focus on the spectral mapping component of voice conversion. Most existing deep learning based voice conversion systems rely on a previous frame alignment step using, for instance, dynamic time warping before the conversion of the speaker's spectra. We propose a sequence to sequence model with attention which completely avoids this explicit alignment, mapping spectral features with different lengths. In order to establish a baseline, we implemented a deep bidirectional long short-term memory recurrent neural network based voice conversion system, which achieves some of the current state of the art results and resorts to the feature pre-alignment step. The experiments conducted revealed that our model is capable of achieving state of the art results if allowed to have a peek into the target features, but struggles with prediction if prevented from accessing this information. Both models are evaluated using an objective metric and the poor performance during prediction is discussed, as well as some possible solutions that can be explored.

Keywords: voice conversion, deep learning, attention models, sequence to sequence, recurrent neural networks, speech processing

Contents

List of Tables

List of Figures

List of Acronyms

ADAM	Adaptive Moment Estimation
ANN	Artificial Neural Network
BBAM	Bernoulli Bidirectional Associative Memory
BLFW	Bilinear Frequency Warping
BPTT	Back Propagation Through Time
CRBM	Conditional Restricted Boltzmann Machine
DAE	Deep Auto Encoder
DBLSTM	Deep Bidirectional LSTM
DBGRU	Deep Bidirectional GRU
DBRNN	Deep Bidirectional RNN
DBN	Deep Belief Networks
DL	Deep Learning
DFW	Dynamic Frequency Warping
DKPLS	Dynamic Kernel Partial Least Squares
DTW	Dynamic Time Warping
EM	Expectation Maximization
F0	Fundamental Frequency
FW	Frequency Warping
GB-RBM	Gaussian Bernoulli Restricted Boltzmann Machine
GMM	Gaussian Mixture Model
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
HMM	Hidden Markov Model
JD-GMM	Joint Density Gaussian Mixture Model
LP	Linear Prediction
LPC	Linear Predictive Coding
LSF	Line Spectral Frequency
LSTM	Long Short-Time Memory
MCD	Mel Cepstral Distortion
MCEP	Mel Cepstral Coefficient

MFCC	Mel-Frequency Cepstral Coefficient
MGC	Mel Generalized Cepstral Coefficient
ML	Maximum Likelihood
MOS	Mean Opinion Score
MSE	Mean Squared Error
MVR	Multivariate Linear Regression
NN	Neural Network
PLS	Partial Least Squares
PSOLA	Pitch Synchronous Overlap and Add
RBF	Radial Basis Function
RBM	Restricted Boltzmann Machine
RNN	Recurrent Neural Network
SD-RTRBM	Speaker Dependent Restricted Time RBM
SGD	Stochastic Gradient Descent
SPTK	Speech Processing Toolkit
TTS	Text To Speech
US	Unit Selection
VC	Voice Conversion
VQ	Vector Quantization
VTLN	Vocal Tract Length Normalization

List of Software

LaTeX^{TM}

MATLAB® 2015 - Academic Version

Python® 2

Speech Processing Toolkit (SPTK)

Chapter 1

Introduction

1.1 Motivation

Speech is an essential communication tool among individuals. As such, the development of computational systems that process speech in various ways is a very interesting and important challenge. A particular case of relevant speech manipulation concerns the modification of speech timbre and prosody characteristics, and is called voice conversion.

Voice conversion (VC) is defined as the process of, given an utterance with the source speaker S's voice, reproducing the same utterance in the target speaker T's voice. Speaker S's utterances should acquire qualities inherent to speaker T, like timbre and pitch, while saying a completely different utterance. This is possible due to the physical characteristics of each individual, associated with one's vocal tract and glottal source, which differ from person to person.

Ideally, a VC technology would convert both timbral and prosodic characteristics from a speaker. In this thesis we focus on the conversion of timbral characteristics, more specifically spectral features, while leaving some prosodic characteristics of the source unaltered. Spectral features are believed to convey more speaker individuality and are easier to extract and model [1], thus being the focus of our work. Although the success of voice conversion systems is still not at desired levels regarding speech quality and naturalness, this research area has been making relevant breakthroughs in recent years.

Applications such as identity change in a text-to-speech (TTS) system are possible with the use of a voice conversion system, without resorting to a huge number of parameters and inherent limitations of the current methods. However, this thesis was motivated by a specific use case of VC, related to an application for the movie industry. With a VC system matching the quality standards, it would be possible to perform movie voiceovers in a foreign language, while keeping the original actor's voice characteristics. Motivated by the latter, this thesis represents a toned down approach to the problem by keeping both source and target speakers in the same language. This allows utterance matching between speakers and therefore considerably simplifies the voice conversion problem.

1.2 Contributions

VC is a fairly recent area of research. However, it has been growing strong in the latter years, especially with the deep learning advances. Therefore, this thesis intends to contribute with a step towards a complete voice conversion system with deep neural networks. In order to do so, we take the current state of the art approach using recurrent neural networks and propose the use of attention mechanisms, already in use on other sorts of deep learning problems, to improve results and eliminate the need for feature alignment. The removal of the alignment process avoids unnecessary errors being modelled by the voice conversion system and is an important advance towards an end-to-end voice conversion system. In addition, we contribute with a Python deep learning framework for Theano to deal with sequence to sequence and attention models, which until then were not supported by the available libraries.

1.3 Thesis Outline

This thesis is divided into six chapters. In Chapter 2, different techniques that impacted the voice conversion state of the art are briefly presented and explained, as well as some basic concepts that are important to understand the general voice conversion system framework.

Chapter 3 describes the chosen state-of-the-art baseline approach for this work, and describes all the essential implementation decisions taken to implement such a system, as well as some minor tweaks we decided to make on the original proposed work.

Due to the need for feature alignment in the baseline approach, we introduce in chapter 4 sequence to sequence models that are currently used in some of the state-of-the-art for speech recognition and machine translation, and are capable of dealing with sequences of different sizes. Related to this type of models, in the same chapter, we introduce the concept of an attention mechanism and propose a sequence to sequence attention model for a voice conversion framework.

In Chapter 5 all the experiments made with the baseline and proposed models are described and discussed, based on the results of the objective evaluation measure used.

Finally, in Chapter 6 we draw some conclusions based on the architectures and experimental results, evaluating the pros and cons of each approach. Some possible future work directions are also discussed in this chapter.

In addition, in appendix A we provide a small guide to the deep learning framework developed as well as some practical code usage examples.

Chapter 2

Background

To be able to understand a voice conversion system, one should be aware of the theoretical background attached to the topic, as well as the several different approaches to the problem in the current state of the art research. Therefore, this chapter focuses on making a superficial literature review of the state of the art of voice conversion.

This chapter has the following structure: Section 2.1 deals with speaker identification; Section 2.2 briefly describes a typical process of voice conversion; Section 2.3 describes current state of the art spectral mapping techniques; and finally Section 2.5 summarizes the main ideas of this chapter.

2.1 Speaker Recognition and Identification

Speaker recognition is a field of speech processing that consists of both speaker identification and speaker verification. Speaker identification is related to the identification of an unknown speaker from a set of known speakers. In other words, we want to find the speaker who sounds the closest to the speech of the unknown speaker. Speaker verification deals with the verification of a subject's identity. In a voice conversion paradigm, knowing how humans and machines identify a specific speaker is of great interest.

Humans easily recognize people by their voices on a daily basis with great accuracy, especially if the degree of familiarity with the subject is high. Sometimes, even a short non-linguistic sample, is sufficient for us, humans, to be able to recognize someone. This is termed as naïve speaker recognition. The process of identifying a speaker with a full speech analysis and decision making via computer analysis is called automatic speaker recognition.

In a voice conversion paradigm, we are interested in both naïve and automatic speaker recognition, and more specifically in speaker-independent methods (the 'how it is being said'). These are of particular interest since it is relevant to know how humans perceive different speakers and what features are important to represent a certain subject.

Identifying someone has its challenges, even for a human listener. Someone's speech may differ due to a range of factors that can contribute for variability and consequently to speaker recognition mismatch. For example, we may find difficult to recognize someone's voice through the telephone or when they have a cold. Due to these challenges, the highest level possible of robustness is desired in the features extracted. Nonetheless, every individual has characteristic voice traits that are unique to them. These characteristics are mainly correlated to vocal tract physiology and learned habits of articulation. Even twin brothers have slightly different vocal traits from one another.

The vocal tract can be thought of as a filter, consisting of the pharynx and the nasal and oral cavities, which can be used to generate speech sounds. Speech sounds can be classified as voiced and unvoiced. Voiced sounds have a quasi periodic waveform, corresponding to the vibrations of the vocal folds, while unvoiced sounds are produced by the relaxed larynx, where the air flows freely, and its waveform resembles random noise without periodicities.

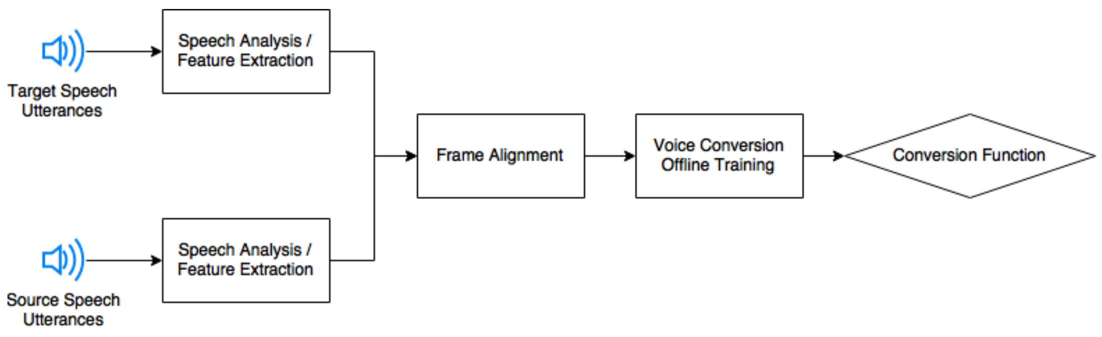
Robust features like the fundamental frequency ($F0$), spectral shape and spectral envelope, are crucial features that relate to speaker identification and consequently to voice conversion. $F0$ is correlated to pitch and corresponds to the frequency of the vocal folds vibrations, while spectral shape and spectral envelope are the features that allow humans to identify different sounds. The peaks of the spectral envelope are known as formants and the first four formants are the most meaningful in speech. By combining and manipulating some of these features, it is possible to achieve a voice conversion system that is capable of emulating a subject's vocal tract.

2.2 Voice Conversion Overview

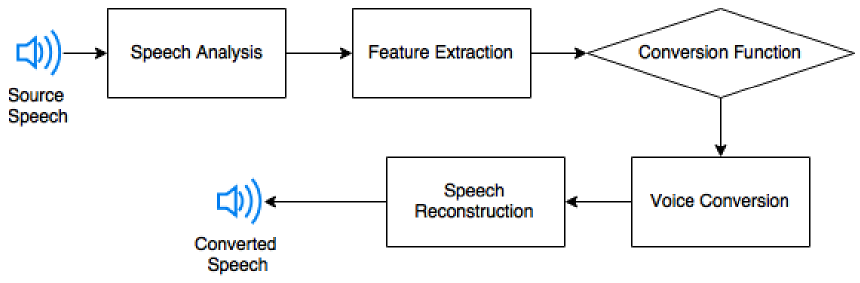
Voice Conversion consists in the modification of a speaker's speech signal so that it can be perceived as if spoken by a specified target speaker. In the more traditional sense of voice conversion (the one discussed in this thesis) this process is done without modification of the language content of the original speech signal. That is, the source and the target speakers have their utterances spoken in the same language and only the speaker-dependent information, such as spectral shape, pitch, duration, intonation and intensity, are modified to achieve the similarity. Cross lingual voice conversion is also possible to perform, where both speakers utter different languages, however this topic is outside the scope of this thesis, as is it introduces several other difficulties into the process.

A typical framework for a voice conversion system is represented in figure 2.1 and it can be broken down into two essential steps of operation: an offline training phase, and a runtime conversion phase.

The main goal of the training phase is to obtain a mapping function of speaker dependent features between source and target, which we will refer to as $F(.)$. However, before obtaining $F(.)$, there are a few processes that the speech has to go through. Input data consists of pairs of source and target utterances with similar language contents. Both signals from the input pair go through a speech production model to extract speaker dependent information, like spectrum envelope, fundamental frequency ($F0$) and an aperiodic component. Some of these components are even further processed to extract low dimensional and more efficient representations of the signal. Depending on the model used, the features obtained may require an alignment process through an algorithm such as dynamic time warping (DTW), in order to align each phoneme to its' counterpart. After this process, the conversion function $F(.)$ can be estimated


Target Speech Utterances
Speech Analysis / Feature Extraction
Source Speech Utterances
Speech Analysis / Feature Extraction
Frame Alignment
Voice Conversion Offline Training
Conversion Function

(a) Training Phase



(b) Runtime or Conversion Phase

Figure 2.1: Typical framework phases for a voice conversion system.

to be able to perform a conversion operation.

At runtime, only the source speech signal goes through the speech analysis and feature extraction modules to obtain a valid input for the conversion function $F(.)$. Once those features are obtained, the conversion function receives them as input and outputs a new set of features. In order to obtain a perceivable speech signal, the converted features go through a speech reconstruction model capable of synthesizing the converted speech.

Depending on the techniques of voice conversion being used, these modules may suffer some changes, although, in general, the purpose of each one stays unchanged.

2.2.1 Speech Analysis and Reconstruction

The speech analysis and reconstruction modules are important in a voice conversion system. As shown in figure 2.1, these two modules are strongly correlated with one another. While the analysis module should be able to represent the signal as a set of mutually independent components for independent and flexible modifications, the reconstruction module should be able to recreate the signal from the same representations. The flexibility of such features is even of greater importance in a voice conversion system, since the representations fed into the speech reconstruction module are the ones modified in the conversion process. The main requirement for these modules is that, after the feature conversion process, the reconstructed speech should be a high quality signal without audible artifacts. Nevertheless, in practice, these models are not perfect, and therefore artifacts still exist in the reconstructed speech.

Both modules are usually based on speech production models, such as the source-filter model. The most commonly used speech production models for both speech analysis and reconstruction modules are Pitch Synchronous Overlap and Add (PSOLA) [2], Linear Prediction (LP) [3], Harmonic Plus Noise

[4], and STRAIGHT [5].

For this thesis we choose STRAIGHT, which stands for ‘Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum‘ which also is based on the source-filter model. STRAIGHT works as a high quality vocoder that analyses and separates the speech into three independent components: a smooth spectrogram free from periodicity in time and frequency, a fundamental frequency ($F0$) contour, and a time-frequency periodicity map that captures the spectral shape of the noise and its temporal envelope. Our choice is based on the flexibility STRAIGHT has for feature manipulation and the high quality of speech it produces. For the same reasons STRAIGHT has been widely used on voice conversion research [6–11].

Although the other models are not discussed in this thesis, we refer the reader to [1] for a brief description of the remaining speech analysis and reconstruction models.

2.2.2 Feature Extraction

In speech processing tasks, typically the signal is not fed directly to the system, but rather represented with more compact features for better manipulation. More specifically, if we go back to the scheme in 2.1, it is easily observed that the feature extraction step is present in both training and runtime phases for an effective signal representation.

With the focus of this thesis being mainly spectral mapping and prosodic manipulation with features incoming a speech analysis module, it is implied that we should be handling spectra or spectrum related features. However, there are many representations to choose from. In order to be suitable for statistical parametric modelling, spectral features should meet the following requirements [1]:

1. be able to represent the speaker individuality well;

2. fit the spectral envelope well and be able to be converted back to spectral envelope;

3. have good interpolation properties and allow for flexible modification.

Representing a spectra data through its spectrogram implies having high dimensional features that take a high amount of storage space when dealing with large amounts of data. Therefore, resorting to a more efficient and low-dimensional representation of the spectrum that can be exploited is necessary. In the following sub-sections we present two different alternatives for representing spectrum data efficiently and why we chose one over the other.

Generalized Cepstral Coefficients

Cepstral analysis is a widely used tool in speech processing tasks. Cepstral coefficients model both spectral peaks and valleys, and provide good envelope fit, an important property for speech synthesis. The particular case of Mel-cepstral coefficients represents the spectral envelope with coefficients spaced from one another in the Mel scale, which focus on the frequencies that have more relevance for the human speech and hearing.

A unified approach to speech spectral analysis was proposed in [12], allowing the computation of different sets of features. In these features are included Linear Prediction and Mel-Cepstral Analysis, among others. By varying two parameters, α and γ, it is possible to choose between the different available features.

A cepstrum $c(m)$ of a real sequence $x(n)$ is defined as the inverse Fourier transform of the logarithmic spectrum, while the mel-generalized cepstral coefficients (MGC) $c_{\alpha,\gamma}(m)$ are defined as the inverse Fourier transform of the generalized logarithmic spectrum calculated on a warped frequency scale $\beta_\alpha(\omega)$:

$$s_\gamma(X(e^{j\omega})) = \sum_{m=-\infty}^{\infty} c_{\alpha,\gamma}(m) e^{-j\beta_\alpha(\omega)m} \tag{2.1}$$

where $s_\gamma(\omega)$ is the generalized logarithmic function and $X(e^{j\omega})$ is the Fourier transform of $x(n)$. The generalized logarithmic function is defined as

$$s_\gamma(\omega) = \begin{cases} \frac{\omega^\gamma - 1}{\gamma}, & 0 < |\gamma| \leq 1 \\ \log \omega, & \gamma = 0 \end{cases} \tag{2.2}$$

and the warped frequency scale $\beta_\alpha(\omega)$ is defined as the phase response of an all-pass system

$$\Psi_\alpha(z) = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}|_{z=e^{j\omega}} = e^{-j\beta_\alpha(\omega)}, \quad |\gamma| < 1 \tag{2.3}$$

where

$$\beta_\alpha(\omega) = \tan^{-1} \frac{(1-\alpha^2)\sin\omega}{(1+\alpha^2)\cos\omega - 2\alpha}. \tag{2.4}$$

In [12] it is admitted that the speech spectrum $H(e^{j\omega})$ can be modeled by $M+1$ mel-generalized cepstral coefficients as follows:

$$H(z) = s_\gamma^{-1}\left(\sum_{m=0}^{M} c_{\alpha,\gamma}(m)\Psi_\alpha^m(z)\right). \tag{2.5}$$

By taking $|\alpha| < 1$ and $\gamma = 0$ it is then possible to obtain the Mel-Cepstral features (MCEP) [13] as

$$\exp \sum_{m=0}^{M} c_{\alpha,\gamma}\Psi_\alpha^m(z) = K \cdot D(z), \quad \gamma = 0 \tag{2.6}$$

where

$$K = \exp(c_{\alpha,\gamma}(0)), \quad \gamma = 0 \tag{2.7}$$

$$D(z) = \exp \sum_{m=1}^{M} c_{\alpha,\gamma}(m)\Psi_\alpha^m(z), \quad \gamma = 0 \tag{2.8}$$

and $\boldsymbol{c} = [c_{\alpha,\gamma}(0), c_{\alpha,\gamma}(1), \ldots, c_{\alpha,\gamma}(M)]^T$ is the vector of mel-cepstral coefficients with the special coefficient $c_{\alpha,\gamma}(0)$, usually called the energy component, since it corresponds to the average log-power of the frame.

Line Spectral Frequency (LSF)

The Line Spectral Frequency is another set of features that have good quantization, interpolation and good representation of the formant structure [1]. These features have been used widely in the text-to-speech research works, achieving better performance than other features, including mel-cepstral coefficients. For voice conversion, however, if the conversion is not stable, guaranteeing a value of the LSF features between 0 and π in ascending order may be hard. This issue may create artifacts in the synthesis process, and therefore these features are not ideal, despite its use on some previous voice conversion works [14, 15].

2.2.3 Frame Alignment

Most voice conversion systems can't handle features with different lengths, therefore the data is required to be aligned in order to allow mapping and comparison between features from source and target speakers. Two possible scenarios must be taken into account: parallel data and non-parallel data.

Parallel Data

In a situation where the dataset is constituted by pairs of utterances from two different speakers with the same linguistic content (*i.e.* speaking the same sentences), we are said to have parallel data. Nevertheless, when both audio files from the source-target speaker pair are analysed, the number of frames do not match, since both speakers uttering the same phonemes at the exact same second for the same amount of time is extremely unlikely. Therefore, in such a situation, a mechanism to align both speakers utterances is required.

The most popular and most efficient technique for frame alignment is Dynamic Time Warping (DTW) [16], a form of dynamic programming. DTW aligns the source and the target feature sequences by minimizing the spectral distance between features and taking into account the temporal structure constraint. We refer the reader to [17] for a more detailed comparison between different DTW implementations in the VC context.

Non-parallel Data

Although this thesis only takes into consideration parallel data, achieving a voice conversion system without it, is possible. In the current state-of-the-art, there are techniques ranging from the minimization of the Euclidean distance between source and target without any context [18, 19], to class based techniques [18], HMM-based methods [20] and pseudo-parallel alignments [19]. We refer the reader to [1], where the author introduces briefly all of these techniques.

2.2.4 Prosody Conversion

A typical voice conversion system involves spectral mapping and prosody conversion. Spectral mapping seeks to change the voice timbre, while prosody conversion aims to modify prosodic features, such as fundamental frequency ($F0$), intonation and duration of the speech. In this thesis, we only convert the fundamental frequency, leaving the others untouched.

In order to convert the speakers' fundamental frequency, there are two types of techniques: the ones that operate on instantaneous $F0$ values (at frame level) and the ones that deal with intonation contours, or $F0$ contours.

The most common technique applied in voice conversion falls on the first category and is a normalization of the mean and variance of the source speaker's $(\log\text{-})F0$ distribution to the target speaker's mean and variance. This operation can be achieved by a frame-level linear transformation of $F0$ values [21]. The converted value of a single frame x' is obtained by

$$x' = \frac{\sigma_y}{\sigma_x}(x - \mu_x) + \mu_y \tag{2.9}$$

where x is the source speaker's $F0$ value, μ_x and μ_y are the means and σ_x and σ_y are the standard deviations of the the source and the target speakers', respectively. By using this method, we are keeping the global $F0$ level and dynamic range while retaining the shape of the source's $F0$ contour.

There are extensions of the previous approach that still operate at the frame-level, namely methods using higher polynomials, Gaussian Mixture Model (GMM) based mapping, and piecewise linear transformations based on hand-labelled intonational target points [1].

With methods that deal with $F0$ contours, prosodic segments (*e.g.* syllables or entire utterances) are represented either as variable-length sequences and processed with DTW [22] or by parameterizing each prosodic segment by a fixed-dimensional parameter vector [23, 24] which is computationally more feasible. These methods are stated to outperform the most frame-level methods, especially in cases where the contours change drastically [25]. However, the intonational contour depends on several lexical and paralinguistic factors and if these factors do not match on the source and target speakers pair, the converted utterance may sound unnatural.

Since the scope of this thesis is the spectral mapping, we opted for the equalization technique, for its simplicity and reliability.

2.3 Related Work

Many techniques have been developed for VC although it is a fairly recent field of study. In this section we analyze some of the core techniques in order to give a brief overview to the reader regarding all the different possibilities available to achieve a VC system. Support for the motivation and choices made throughout this thesis, will also be provided.

2.3.1 Vector Quantization based methods

Early works on voice conversion date back to 1988, when Abe *et al.* proposed using a vector quantization [26, 27] approach in combination with the so-called codebook mapping, a popular technique in speech recognition to perform a mapping between spectral parameters, power values, and pitch frequencies.

In this approach, speech was then modified through the codebook mapping technique and synthesized with an LPC vocoder.

Vector Quantization (VQ) represents a simple and straightforward way to perform spectral mapping. VQ consists of a training and a runtime phase, which resembles the typical VC system illustrated in 2.1. During the training phase the clustering of the stacked features from both source and target is made to build the mapping codebook, as depicted in figure 2.2. At runtime, each source feature is paired up with the corresponding target feature entry in the codebook, which will then be selected as the converted feature.

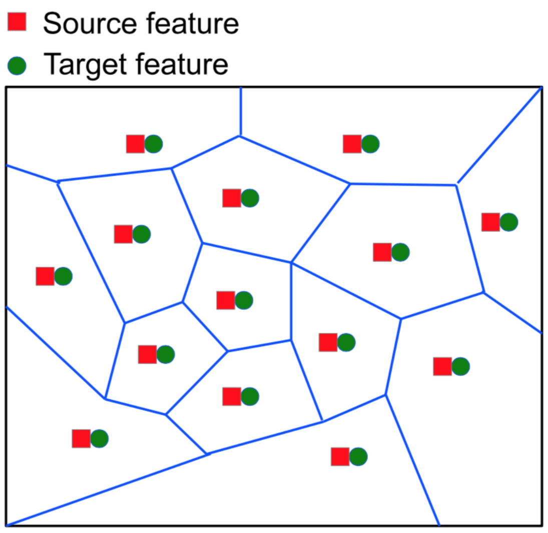


Figure 2.2: Illustration of the vector quantization feature codebook mapping process. The square points are entries for source feature codebook while circle points stand for entries for target feature codebook. The square points and circle points appear in pairs to stand for joint vectors. Extracted from [1].

The classical implementation from [27] uses dynamic time warping to make the correspondences between the features of source and target, which are then accumulated to make a histogram. This histogram will be employed as a weighting function between feature pairs to produce the mapping codebook.

The biggest problem of this technique is its hard clustering nature, which produces incoherence between frames of speech. However the target's spectral details are kept intact.

2.3.2 Gaussian Mixture Model based methods

Gaussian mixture model (GMM) based methods are the current baseline methods used in most of the state of the art research on voice conversion. The technique was proposed in [7, 15, 21] to respond to the unnaturalness and glitches in the converted audio produced with the VQ method, by using soft clustering.

From past research, linear conversion functions have been found to produce good results, however having a single global transformation is quite a limitation in performance. Therefore, GMM-based VC methods model the data with a Gaussian mixture model and find local linear transformations for each

Gaussian used.

Two main approaches are taken regarding GMM-based voice conversion systems: modeling the source with a GMM [21] or modeling the joint density (JD-GMM) between between source and target speakers [7, 15]. Depending on the criteria used to optimize the conversion function, the implementation of JD-GMM will vary. The most popular criteria are the Mean Square Error (MSE) [15] and Maximum Likelihood (ML) trajectory generation [7].

Both solutions require separate training and conversion phases. In the minimum squared error based solution, a GMM is employed to model the joint distribution of the aligned source and target features pairs. The model parameters are the mean vectors and covariance matrices of each of the Gaussian components, alongside with their prior probabilities. To estimate them, the well-known expectation-maximization (EM) algorithm is adopted to maximize the likelihood of the training data. Once estimated, the model parameters can be employed to implement the conversion function.

In the maximum likelihood parameter trajectory based solution, the joint distribution is modelled similarly, with the same parameters. However, in the runtime phase the process diverges in the way the conversion function is calculated. To generate the target feature trajectory the likelihood of the target features given the model parameters and source features is maximized instead. The EM algorithm can be employed to generate the parameter trajectory, or alternatively a sub-optimal approach can be taken in exchange for performance. Often, dynamic features are stacked with regular features for performance improvement.

Research has shown that the ML of parameter trajectory produces better speech quality and yields less spectral distortion than MSE approaches. For this reason, this approach is taken as a baseline approach in many VC related research works.

Nevertheless, GMM based methods still have unresolved issues. The mean and covariance for each Gaussian component is computed using all the training samples, which is called a statistical average. This leads to over-smoothing of the converted speech. In addition, if the correlation of source and target features is low, the covariance matrix will have very small values of cross covariance and non-diagonal elements. Such a covariance matrix has a small contribute to the converted speech, increasing the influence of the target mean parameter.

To deal with over-smoothing [28] and over-fitting problems [6, 29] of JD-GMM, several methods have been proposed. In [28], mutual information criterion has been used during parameter generation to enhance the dependency between source and converted feature vectors. In order to mitigate over-smoothing, global variance [30], bag-of-Gaussian model [31] and discriminative training were proposed. Additionally, channel model methods [10] have been proposed to improve the performance of the conversion function.

2.3.3 Frequency Warping based methods

The speech quality of VQ-based voice conversion turned out to be poor, which was attributed mainly to the LPC vocoder synthesis. This led to the proposal of a PSOLA based VC system by Valbret *et al.* [32]

in 1992, which introduced frequency warping to voice conversion.

Frequency Warping (FW) attempts to compensate for the differences between the acoustic spectra of source and target speakers through a warping function. The frequency warping function creates a new spectral cross section from another given spectral cross section. An illustration of a warping function is shown on figure 2.3.

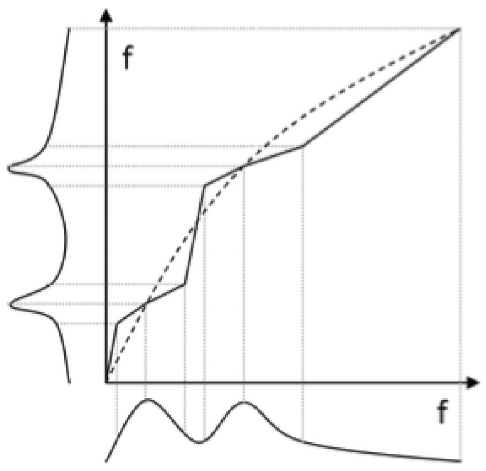


Figure 2.3: Frequency Warping. The solid line represents a traditional piecewise-linear frequency warping function, while the dotted line is a simpler version of the same function, which can be accurate enough to map the central formant frequencies. Figure extracted from [33].

Within the frequency warping based methods, there are approaches in which a single warping function is applied to the whole speech signal, whilst others apply a warping function to a group of frames. On one hand, having a single warping function provides coherence to the converted signal, but lacks flexibility regarding similarity of the converted spectra to the target. On the other hand, having multiple warping functions improves similarity results, but can lead to loss of quality due to discontinuities created at each frame group [34].

In general, these techniques produce better results with respect to sound quality of the produced speech, although less effective regarding similarity between target and converted speech [14, 35]. However, due to the good quality obtained in the converted speech, they are still a quite popular method amongst researchers.

Early approaches to VC using FW resort to popular techniques used in speech recognition like Vocal Tract Length Normalization (VTLN) [36] and formant manipulation [34]. The degree of manipulation obtained from these attempts [32, 34, 36] was low, but good quality converted speech was produced. Some of these techniques are Formant-based [34] and require a robust formant detection (and often tracking), which is difficult and thus leads to a poor performance. To tackle the low degree of spectra modification, a combination of the statistical approaches and FW were proposed in [14, 37]. Nevertheless, these approaches implied a compromise between capturing overall trends in the spectral power and maintaining spectral details.

Dynamic Frequency Warping (DFW) and Bilinear Frequency Warping (BLFW) were proposed in [35, 38] and [33, 39] respectively. These works introduced Amplitude Scaling to try to compensate for spectral inaccuracies and improve the spectral modification towards the target's spectrum, since the warping procedure does not modify the relative amplitudes of meaningful parts of the spectrum. Dynamic Frequency Warping makes use of dynamic programming to compute the lowest-distortion FW path from discretized spectra. The approach proposed in [35] works at the acoustic class level (*e.g.* phonemes) instead of depending on the traditional frame-by-frame alignment, which discards the need of parallel corpora for the training phase of the conversion. Bilinear Frequency Warping makes use of bilinear warping functions, that unlike piecewise linear FW functions are defined by a single parameter [33, 36] and therefore reduce the amount of training data needed.

In general, all the methods introduced above ignore the physical property constraint of the spectra and only consider spectral peak information. This led to the proposal of a correlation-based method of frequency warping by Tian *et al.* [40]. In this work, instead of minimizing the spectral distance between the converted and target spectra, the proposed method maximizes the correlation of spectra pairs and it is shown to improve both quality and speaker identity of the converted speech over the other FW methods.

2.3.4 Unit Selection based methods

While statistical methods use spectral features to generate converted speech and frequency warping uses the frequency axis of a source and maps it towards the target's, unit selection (US) based methods make use of the target's feature set to generate the converted speech.

Inspired by unit selection based speech synthesis, proposed in [41] to automatically select and concatenate target speech segments to generate a speech signal, unit selection based voice conversion makes use of the same principle to generate the converted speech from source to target speaker.

In [42] unit selection in conjunction with GMM VC techniques were first proposed for voice conversion, followed by [43] that proposes JD-GMM instead to achieve better results. On [42, 44], both target and concatenation cost are considered to avoid discontinuities at concatenation boundaries.

Most VC methods ignore temporal information and assume short-term frames are independent from one another. Contesting this assumption, [44] proposes an exemplar-based unit selection voice conversion system, that resorts to exemplars, time-frequency speech segments that span over consecutive frames to pair up the data.

In general, US-based voice conversion presents a high level of similarity to the target speaker, due to the way this conversion method works. However, in order to achieve good results with this technique, a large database that is phonetically rich and balanced is required to ensure phonetic coverage. Such a large dataset is expensive to collect and impractical for a voice conversion system for multiple target speakers.

2.3.5 Partial Least Squares based methods

Classic GMM approaches usually need to compute covariance matrices, a computationally heavy procedure, which often leads to the over-fitting of the training data due to the high number of degrees of freedom available for the conversion. Full covariance matrices result in a good representation of the data with a low number of Gaussian mixtures but require a large amount of data to be computed. However, a low amount of training data is desired for a practical implementation of a VC system. On one hand, having a reduced number of degrees of freedom on the covariance matrices is desired to control the over-fitting of the training data. On the other hand, computing a simplified version of the covariance matrix, like a diagonal version of it, translates into a high number of mixtures for an accurate representation.

Posed with the previous problems, the approach in [6] proposes using Partial Least Squares (PLS), specifically addressing the cross-correlation between predictor and predicted values. PLS combines the concepts from principal component analysis and multivariate linear regression (MVR), which can be viewed as a mid-term solution for the covariance matrices problem between the two previous options.

This approach assumes that the source and the target data can be projected into a latent space, in which the relationship between source and target features is established, via a linear transformation. By solving the regression model obtained from these assumptions, it is possible to map source features into new converted features, that should resemble the target. The appropriate selection of some of the involved parameters contribute to prevent over-fitting by reducing the number of degrees of freedom available to the model.

In order to obtain an efficient global conversion function and avoid the over-fitting of the data, Helander *et al.* resorts to a GMM to get the initial division of the data before extracting the mapping function with partial least squares regression. This aims to achieve multiple local linear transforms, considering the low probability of the data being well modelled by a single linear transformation. The experimental results reported confirmed the effectiveness of the PLS regression based method over the conventional JD-GMM methods.

The PLS based method for voice conversion just presented converts each frame of speech independently, not taking into account any temporal correlation between frames and relying solely on linear transformations. With these concerns, dynamic kernel partial least squares (DKPLS) voice conversion was proposed in [9]. This approach assumes that the source and the target features have a nonlinear relationship. The non-linearity is implemented by a Gaussian kernel function, which is used to project source features into a high-dimensional kernel space. In addition, to model the time dependencies in the feature sequence, kernel vectors of three consecutive frames are stacked as a supervector, and this stacked supervector is employed as the source feature which will be the input for PLS. Furthermore, this implementation fully converts spectral features, $F0$ and aperiodicity.

Although effectiveness has been shown with the DKPLS implementation against baseline JD-GMM methods, it still has some drawbacks. In the first place, it ignores temporal constraints in the target features; secondly, it lacks flexibility with respect to long-term temporal dependencies, being dependent of the source vector kernel stacking, and finally, it uses low-resolution Mel-generalized cepstral (MGC) features (24 dimensions), and results with high-resolution MGC's are still unknown.

2.3.6 Neural Networks based methods

With the success of neural networks (NN) in the field of machine learning, many researchers tried to apply some of the techniques to the voice conversion problem. These networks are exceptionally good at learning non-linear models, which until then had not been explored in the VC context (mostly linear models or variations were used), thus presenting itself as a potentially good approach.

The simplest form of neural network is referred to as Artificial Neural Network (ANN). However, more recently, a new field arose within Machine Learning: Deep Learning. The main difference with respect to the classic ANN approach is that neural networks in deep learning are much larger and usually built from multiple layers of ANNs (therefore the Deep nomenclature). In the following subsections, the approaches to VC using neural networks will be explored, starting with the early ANN approaches and concluding with the more recent Deep Learning techniques.

Early Artificial Neural Networks

One of the first attempts of a voice conversion system using artificial neural networks consisted in using a simple feed-forward neural network trained with a simple back propagation algorithm to transform the formants of the source into the target's [45] and then re-synthesize the converted speech using an LPC based vocoder alongside with a simple $F0$ conversion. Afterwards another method based on LPC spectral features was introduced [46], but this time using a three-layer radial basis function (RBF) network with Gaussian basis to transform the vocal tract characteristics.

Both experiments from [45] and [46] were conducted with carefully prepared training data, with clean recordings in a studio. This, besides being a tedious task, is inconvenient for a real-world application scenario, where having converted speech from limited data in the least time possible is desired.

The first research work that compares the mapping abilities of spectral features with ANN's against baseline methods using JD-GMM and ML trajectory was presented in [8]. This work differs from GMM based methods by directly mapping the source features onto the target acoustic space instead of capturing the joint distribution of source and target features in conjunction with maximum likelihood parameter generation to obtain smooth trajectories.

NN based voice conversion systems are also a two step process, with a training and a runtime phase. In [8], during the offline training phase, 24 dimensional Mel-cepstral coefficients (MCEPs) are extracted as spectral parameters alongside with fundamental frequency estimates ($F0$). After aligning the pairs of features from both speakers, the back-propagation algorithm is adopted to estimate the network's weight parameters, which will represent the voice conversion function. At runtime, the conversion process is as straightforward as propagating the features of the desired source speaker, from the input until the last layer of the network. This process results in a new set of features that represent the converted speech from source to target. The converted features in conjunction with some simple manipulations of the $F0$ estimates, can be transformed into a new converted speech signal via a speech synthesizer. Experimental results report that the ANN approach yields similar results to the VC systems using JD-GMM with global variance.

Later on, inspired by the JD-GMM based voice conversion techniques, the work in [47] uses restricted Boltzmann machines (RBM) as probability density models to represent the joint distributions of source and target speech features. The RBM was originally introduced as an undirected graphical model that defines the distribution of binary visible variables with binary hidden (latent) variables. After being extended to deal with real-valued data with the introduction of the Gaussian-Bernoulli RBM (GBRBM), it became a popular tool for representing complicated distributions of data, and therefore suited for representing the distributions of the speaker features.

Despite having a stable behaviour and relatively good results regarding conversion, JD-GMM techniques still lack in quality of the synthesized speech due to the over-smoothing problem. RBMs have a better ability to capture correlations between two speakers and can be used to describe the distribution of high order spectral envelopes, while also eliminating the problem of inverting correlation matrices inherent to GMM techniques.

Deep Neural Networks

Deep Learning (DL) is a new area of Machine Learning research which has seen a huge growth in popularity in the last few years. DL has been characterized as a class of machine learning algorithms that make use of a cascade of many layers of nonlinear processing units for feature extraction and transformation [48].

During many years the community believed that training multiple stacked neural networks was unfeasible, until Hinton *et al.* introduced an effective pre-training algorithm for Deep Belief Networks [49] (a network consisting of several layers of RBMs), which turned out to be a big success in the machine learning area [50].

Within voice conversion research, several techniques have been proposed in order to achieve a non-linear mapping between source and target speaker's features. After the success of the work proposed in [47] with RBMs, Nakashima *et al.* proposed the use of two DBN networks connected by a simple feedforward NN to achieve a high-order feature space that can be converted [50]. Each of these DBNs is trained independently for each source and target speaker with unsupervised learning to obtain features in a high-order eigenspace for each speaker. These representations then form input pairs for a supervised training for the simple neural network which perform a mapping function between source and target high-order features. Once the features from the source are converted through the trained NN in the high order eigenspace they are brought back to the cepstrum space using the DBNs for the inverse process. Using a similar architecture, the research in [51], experimented with sequence-error minimization instead of the traditional frame-error minimization typically used. By adopting a sequence-error optimizer, computational cost is more demanding during the training phase but it leads to improvements in comparison with its frame-error counterpart, according to the experiments.

The work described in [52–55] also resorts to RBMs but using different architectures. In [52, 54] conditional restricted Boltzmann machines (CRBM) are used to capture linear and nonlinear relationships between source and target features. A CRBM is a non-linear probabilistic model, proposed in [56], used to represent time series data consisting of: (i) an undirected model between latent variables and the current visible variables; (ii) a directed model from the previous visible variables to the current visible variables; and (iii) a directed model from the previous visible variables to the latent variables [54]. In the voice conversion context, these networks try to capture the abstractions to maximally express speaker dependent information, while ignoring phoneme related information. In addition, the network's architecture also allows to discover temporal correlations. While the research from [52] directly uses a CRBM to estimate the target speaker's features from the input, the work in [54] uses two CRBM networks for each source and target speaker to obtain higher order features that can be then converted using a feedforward NN.

The 4-layer deep neural network (DNN) described in [53] cascades two RBMs with an intermediate Bernoulli Bidirectional Associative Memory (BBAM) [57] using generative training layer by layer. The purpose of the RBMs is to model the distributions of the spectral envelopes for source and target speakers while the BBAM models the joint distribution of the hidden variables extracted from both RBMs. Both RBMs are trained using unsupervised learning. The BBAM is trained with supervised learning, similar to what was done in [50].

Experiments using a deep autoencoder (DAE), a type of unsupervised DNN that is able to learn a representation of its input, were also conducted in [58] with the same objective of achieving a high order feature representation. However, the results were not as promising as the current research with RBMs and other neural network architectures.

Later, similar to what was done with the CRBMs for voice conversion, the researchers from [59] presented a different method, which they call speaker dependent restricted time restricted Boltzmann machine (SD-RTRBM) based voice conversion [55, 60], where the previous CRBM method was improved. The architecture of the latter resembles the architecture of a recurrent neural network (RNN) and therefore inspired the work introduced in [11] which resorts to a Deep Bidirectional Long Short-Term Memory Recurrent Neural Network (DBLSTM-RNN) architecture.

The DBLSTM-RNN approach aimed at a good speaker dependent representation by using a deep architecture, resorting to several layers of bidirectional LSTMs. A single layer of BLSTM is by itself a tool suited to extract temporal dependencies from a sequence of data both in forward and backwards directions and therefore, a deep architecture of BLSTM is well suited for a voice conversion system. This approach combines a state of the art techniques with some concepts already used in other approaches. Its simplicity, promising experimental results and remaining unexplored possibilities, make this architecture one of the most interesting from a research point of view.

In general, most of the methods using deep learning architectures shown better results than the traditional JD-GMM based method with global variance and ML parameter trajectory estimation. Nevertheless, it is relevant to point out that a significant amount of training data is required to perform the training of such a system (usually more than GMM methods), which is definitely a downside to the DNN approach.

2.3.7 Exemplar-based methods

The most recent technique being used within the state of the art is exemplar-based voice conversion. Referred in a previous subsection of this chapter, exemplars have also been used to perform unit selection based voice conversion.

Exemplars are basis of spectra, extracted from the training data, that can span across multiple frames. Research in [61–64] proposes different voice conversion systems using exemplars and the results have shown to outperform some of the more traditional techniques in voice conversion.

The basic idea behind exemplar-based voice conversion is to reconstruct a spectrogram as a weighted linear combination of a limited set of basis spectra extracted from the training data. The spectrogram of each of the speakers is decomposed into two matrix components: a dictionary of exemplars and an activation matrix. During the offline training, the source and target dictionaries are constructed. At runtime conversion, the activation matrices are estimated from the source dictionary and activation matrices. Finally, the activations are applied to the target dictionary to generate the converted spectrogram.

Most of that research have been conducted by Wu *et al.* which has been developing the method for voice conversion. The latest research is presented in [64] and aims at combining the advantages of using low and high dimensional features so that the temporal constraints and spectral details can be taken into account by each type of features respectively. To overcome the main difficulty of this type of VC system, the estimation of the activation matrix, a nonnegative matrix factorization is proposed using both kinds of features.

2.4 Evaluation Metrics

The objective of a voice conversion system is to transform the source's speech to maximize the resemblance to the target speaker. A successful VC system produces natural, intelligible and identifiable speech. These qualities relate to how human-like the converted speech sounds are, easiness of how words can be identified by a listener, and how recognizable the individuality of speech is, respectively.

To distinguish between performances between different approaches to VC and to evaluate the quality of the speech produced, converted speech must be submitted to objective and subjective measures. Objective measures are exclusively based on the output produced and are widely the most adopted, since they represent the fastest and cheapest way of evaluating the performance of a VC system. Subjective measures are conducted with test groups of people and evaluate the system performance based on people's opinions.

2.4.1 Objective measures

Among the most popular objective measures for a VC system is the Mel-cepstral distortion (MCD), used in the great majority of the current state of the art research [7, 11, 64]. The MCD is calculated between the generated target frame and the target frame (ground truth) as follows:

$$MCD[dB] = \frac{10}{\ln 10}\sqrt{2\sum_{d=1}^{N}(c_d - c_d^{converted})^2} \tag{2.10}$$

where c_d and $c_d^{converted}$ denote the d-th target and converted target MCEP, MGC, or Mel frequency cepstral coefficients (MFCC) features, N represents the dimension of the features and it can include or not the energy component of such features, depending of the approach being followed.

MCD stands out for having a good correlation with the results of subjective tests and thus it is frequently used to measure the quality of the transformation. By being related to the vocal characteristics it is an important measure for evaluating voice conversion.

2.4.2 Subjective measures

As far as subjective evaluations go, there are two main measures that are very popular in voice conversion research: the ABX test and the Mean Opinion Score (MOS) test. Subjective evaluation tests have the advantage of being based on human perception, while objective tests try to mimic it through data analysis.

Mean Opinion Score (MOS)

In an MOS test, the objective is to evaluate the degree of naturalness of the converted speech in relation with the original target speech. During this test, the listeners are asked to rate the naturalness of a given converted speech using a 5-point scale (5: excellent, 4: good, 3: fair, 2: poor, 1: bad). Several utterances are evaluated by each listener and then the scores of all the listeners are averaged in order to obtain a measure for the VC system under evaluation.

ABX Test

In an ABX test, listeners are asked to choose which of the sample (A or B) sounds more similar to X, with X being the original target speaker utterance. Usually the samples from A and B are from two different VC systems and this test is best suited for distinguish which one performed best in relation to the other. To avoid biasing the listeners, the samples A and B associated with the two VC systems are shuffled before each question. When prompted with a question a listener can answer one of three options: A is more similar to X; B is more similar to X; or no preference (N/P).

2.5 Summary

In this section we introduced a general framework of a voice conversion system and briefly presented each of its operating modules. In a nutshell, these modules are speech analysis and reconstruction, feature extraction, and conversion modules. Each of these modules has its own purpose, from the analysis and representation of the speech signal, to the manipulation of such features and the reconstruction of the converted speech signal. In most VC systems, an extra module has to be added before conversion in order to align source and target features and to allow the conversion process.

Further ahead in the chapter, some state of the art voice conversion techniques were introduced. These can be divided into seven different categories: VQ based, GMM based, FW based, US based, PLS based, NN based and exemplar based. From the analysis made, it is clear that GMM based methods, and more specifically JD-GMM with ML trajectory based methods are the most popular approaches of voice conversion, and are used as a baseline reference in most research works. Nevertheless, these types of approaches struggle with the overfitting and over-smoothing problems. Some approaches, however, such as partial least squares based techniques, exemplar based techniques, and neural networks based systems have shown to outperform the current JD-GMM baseline. Due to its potential and growing popularity, the deep learning based approaches are the ones that have been explored the most and have shown some of the most interesting results and therefore, are the ones explored in this thesis.

Chapter 3

Voice Conversion Using Deep Bidirectional Recurrent Neural Networks

3.1 Introduction

Voice Conversion is a topic that still has a lot to be explored. Among all the different approaches available, choosing one to follow is hard. In this thesis, we take the deep learning approach, because it is a fast growing research area and it shows great potential for a voice conversion application. Among the different approaches that resort to deep learning for VC, the ones that seem to be the most promising are CRBMs and LSTMs, achieving similar results when compared to each other. The baseline adopted was the work of Lifa Sun *et al.*, with deep bidirectional recurrent neural networks [11]. These types of networks deal very well with large amounts of data and are ideal to learn long term temporal relations, making them a good fit to the VC problem.

This chapter has the following structure: Section 3.2 deals with some of the basic concepts of Recurrent Neural Networks (RNNs) as well as some of its variations; Section 3.3 describes the proposed framework for voice conversion using RNNs; and finally, Section 3.4 summarizes the whole chapter, highlighting the most important topics to retain. Throughout this chapter, we admit that the reader is familiar with basic machine learning concepts such as supervised learning, the multilayer perceptron, back propagation, and stochastic gradient descent.

3.2 Recurrent Neural Networks

A Recurrent Neural Network (RNN) is a type of neural network that has the capability of learning based on the content of previous data. Taking a real world example: if we consider our data being the text content of this chapter, a human being would process the meaning of each word based on the context of the previous words of the sentence. RNNs model the same behaviour with internal loops, allowing information to persist on the network.

Understanding the neural network internal loop concept can be hard, however imagining the unrolled network in time makes it easier to conceptualize how these networks work. This way, what is perceived is a sequence of multiple copies of the same neural network for each time step (figure 3.1).

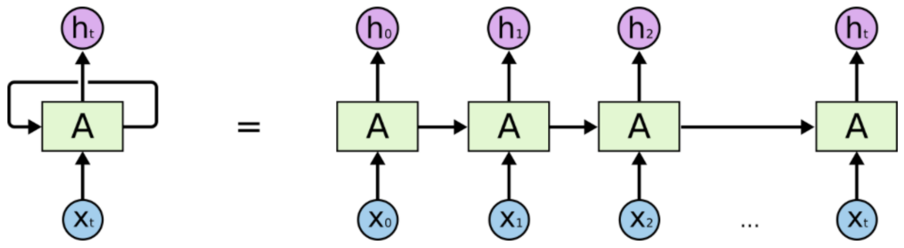


Figure 3.1: Unrolling an RNN in time. Extracted from [65].

For a traditional RNN (also known as vanilla RNN), given an input sequence $x = (x_1, \cdots, x_T)$, the hidden vector $h = (h_1, \cdots, h_T)$ and the predicted output vector $y = (y_1, \cdots, y_T)$ can be computed from $t = 1$ to T according to the following iterative equations:

$$h_t = \sigma(W_h \cdot [h_{t-1}, x_t] + b_h), \tag{3.1}$$

$$y_t = W_y \cdot [h_{t-1}, x_t] + b_y, \tag{3.2}$$

where h_t corresponds to the RNN's hidden state, x_t to the input, y_t to the output, σ to a non-linear activation function and W and b are weight matrices and biases, corresponding to the network's parameters.

In the last few years, RNNs have been used to solve a variety of tasks with plenty of success, like in speech recognition, language modeling, text-to-speech synthesis, among others [11]. However, vanilla RNNs have difficulties learning long-term dependencies if the information is spanned through longer periods of time. Training these networks with the traditional back propagation through time (BPTT) algorithm has been proved to be extremely difficult due to the exploding and vanishing gradient problems [66]. These problems led to the search of new alternative networks that could handle such information.

Motivated by the previously mentioned problems, the long short-term memory (LSTM) was proposed, and later the gated recurrent unit (GRU), a simplification of the LSTM. In the next subsections, these two network architectures are introduced and briefly explained to give the reader a better understanding of the proposed framework in section 3.3.

3.2.1 Long Short-Term Memory

The LSTM was originally proposed by Hochreiter and Schmidhuber in 1997 [67] and developed by many in following works. The current form of the LSTM is capable of learning long-term dependencies and works very well in a wide variety of problems.

While Vanilla RNNs have a very simple cell structure made of a single neural network (see figure

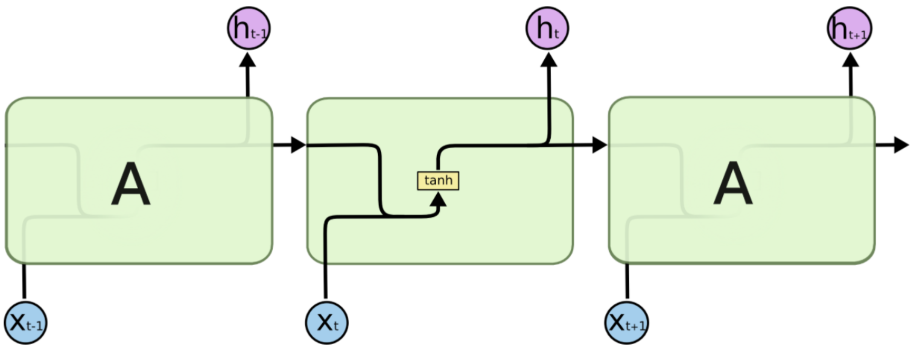


Figure 3.2: Internal RNN structure. Extracted from [65].

3.2), LSTMs have various structures called gates (four in total). These gates interact with each other to process information in order to better handle temporal relations. Each of these gates can be viewed as a separate neural network, and when combined together, they form an LSTM cell (figure 3.3).

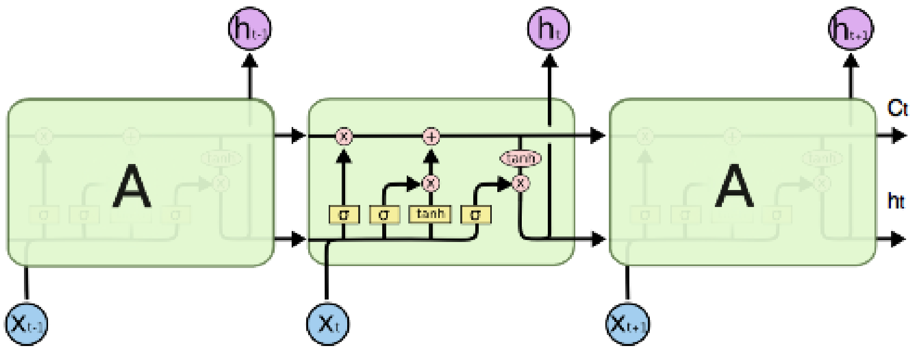


Figure 3.3: Internal LSTM structure. Extracted from [65].

To explain the LSTM operation, we will resort to the visual aid of the figures presented throughout this section, which are borrowed from [65], so understanding the notation used is important. Each line carries an entire vector, yellow squares represent a simple neural network layer and pink circles denote a pointwise operation. A divergence in the lines means a copy of the vector, while a convergence in the lines denotes a concatenation.

The LSTM equations replace equation 3.1 in the RNN operation and are described as following:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.3}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.4}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.5}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.6}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{3.7}$$

$$h_t = o_t * \tanh(C_t) \tag{3.8}$$

where σ represents an activation function (usually a sigmoid), x_t is the current input, h_{t-1} is the previous hidden state, C_{t-1} is the previous cell state, f_t is the forget gate, i_t is the input (write) gate, $\tilde{C}_t$ is called a shadow gate, C_t is the current cell state, o_t is the output (read) gate, h_t is the LSTM current hidden state, and W and b are weight matrices and bias vectors, respectively.

The LSTM cell receives information continuously from the previous states via its C_{t-1} and h_{t-1} inputs, just like a conveyor belt. With this information the cell has the capability of deciding whether to let the information through, if it is relevant, or to ignore it. This process is commanded by the gates and although this line of thought may lead the reader to think in a binary way, that is not the case. The four gates have activation functions that output a value between zero and one, and therefore they weight the information at each time step.

The first step of the LSTM decision process is to decide which information to throw away via the forget gate. The next step is to decide what to write in the cell state. The input gate decides which values to update, while the shadow gate creates new candidate values that could be added to the state. Combining the information of the forget gate with the previous cell state and the input gate with the shadow gate, an informed decision on the new value for the current cell state is made. The final step is to decide what to output. The output gate takes care of the decision on what parts of the cell are going to be outputted. Then, the current cell state is pushed through a hyperbolic tangent ($\tanh$) activation (pushing the values between -1 and 1) and multiplied by the output gate. This product will be the cell hidden state h_t. To get a more in depth view about RNNs and LSTMs, the reader is referred to [65] and [68].

3.2.2 Gated Recurrent Unit

The GRU was proposed by Cho *et al.* (2014) in [69] as a variation for the current LSTM architecture. While the LSTM has to coordinate its writes and forgets, the GRU links them explicitly into one single gate, called the update gate. In other words, instead of doing selective writes and selective forgets, the GRU takes a step further, and instead does selective overwrites by setting the forget gate to 1 minus the input (write) gate. The GRU cell internal architecture is depicted in figure 3.4.

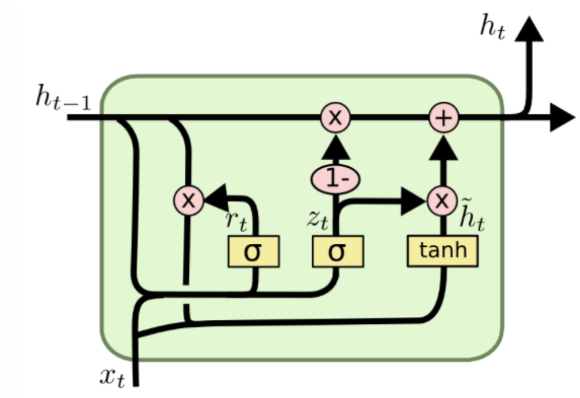


Figure 3.4: Internal GRU structure. Extracted from [65].

The fundamental equations of the GRU are the following:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{3.9}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{3.10}$$

$$\tilde{h_t} = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{3.11}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h_t} \tag{3.12}$$

where x_t is the current input, h_{t-1} is the previous hidden state, z_t is the update gate, r_t is the reset gate, $\tilde{h_t}$ is the shadow gate, h_t is the current hidden state, and W and b are the weight matrices and bias vectors respectively.

The operation of the GRU is very straightforward and similar to the LSTM with the exception of the different gates. It is relevant to point out that the original paper [69] called r_t the reset gate, although it works more as a read gate.

Different studies have compared the performance of several LSTM variants, reaching the conclusion that they are similar. However, different architectures work best on different problems [70]. One of the great advantages of the GRU is the lower number of gates, making it less computationally demanding to train.

3.3 Baseline Framework

The baseline framework adopted in this chapter is based on the architecture proposed in [11], with a few minor twists. Throughout the remainder of the chapter, we will go back to some results in that paper, comparing it with the proposed implementation.

The core concept is to train a deep bidirectional recurrent neural network to perform a mapping between source and target speakers spectral data. The concepts related with recurrent neural networks were already covered previously, but the deep and bidirectional parts deserve particular attention.

A bidirectional recurrent neural network works like two independent RNNs, a forward RNN and a backward RNN. Both these networks receive the same input and produce independent hidden state outputs h_f and h_b. However, the backward RNN inverts the data's time axis, scanning the data from the end to the start. Both these hidden states are then combined into a single hidden state, by a merge process. In this thesis, for simplicity, this process will be considered as the sum of the hidden states: $h = h_f + h_b$. Bidirectional RNNs are exceptionally good in dealing with a large number of time steps.

A deep bidirectional recurrent neural network (DBRNN) is the cascading of several layers of bidirectional RNNs to obtain a deep architecture, feeding the hidden state of the previous layer into the next one. A two layer DBRNN architecture is illustrated in figure 3.5.

The DBRNN-based voice conversion system has an architecture that is in fact very similar to the flowchart illustrated in figure 2.1. In figure 3.6 we depict the complete framework for this baseline system with combined train and conversion phases, while the next subsections describe the process to achieve

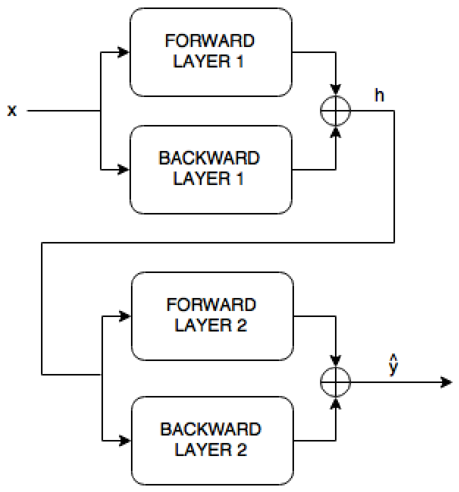


Figure 3.5: A two layer deep bidirectional recurrent neural network. The variable x represents the input, h represents the hidden state and $\hat{y}$ represents the network's prediction of the target y.

this framework, from feature extraction to conversion and speech reconstruction.

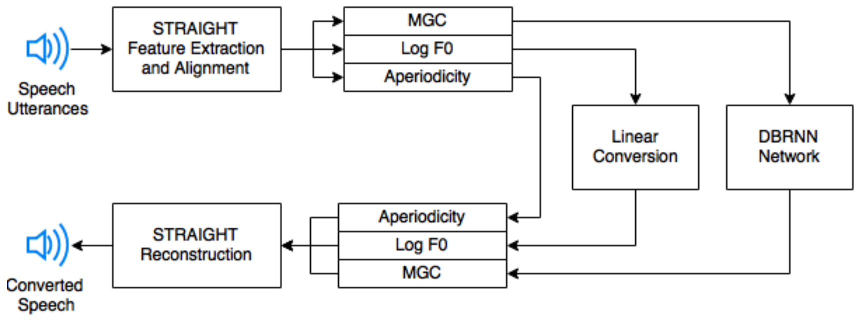


Figure 3.6: Overview of the proposed baseline voice conversion framework.

Performing spectral mapping implies a regression function. This means that the proposed network should be able to learn a conversion function that outputs a converted spectrum (or spectrum related features) from an input, based on the example pairs provided in the training set (figure 3.7). During training, the performance of the system will be periodically evaluated in a validation set, and further ahead, after training finishes the performance of the full VC system is tested on a test set.

The system described in [11] uses a deep bidirectional LSTM network to perform the spectral mapping regression. The framework in this thesis also uses a DBLSTM, and, in addition, proposes a similar architecture using a deep bidirectional GRU instead. In chapter 5, a comparison is performed between the two systems.

Note to the reader: due to the use of both LSTM and GRU recurrent neural networks in this chapter, from now on the term RNN will be used when the context is suitable for both of them. If the context of the subject being dealt with is specific to either an LSTM or a GRU, then the specific name of the architecture will be used.

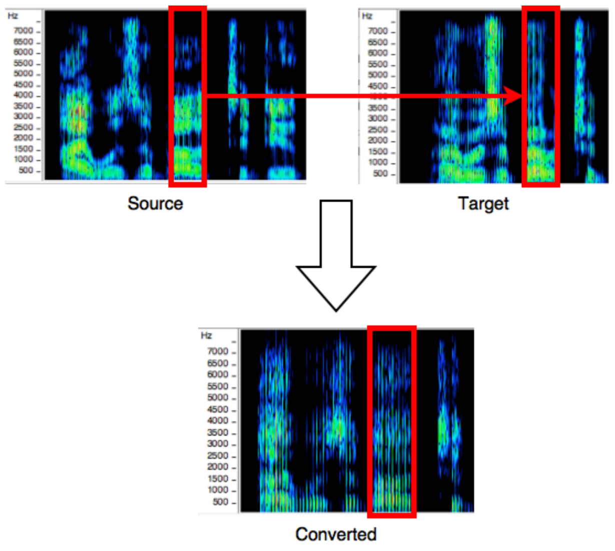


Figure 3.7: Male to male spectral mapping example. The red rectangles indicate the same phoneme for source, target and converted utterances.

3.3.1 Feature Extraction

In order to perform spectral mapping, the network must be fed with spectra from each of the speakers utterances, or some representation of it. This implies the necessity of a tool capable of extracting both source and target's signals spectra. In addition, as covered in chapter 2, this tool should be able to recreate the converted waveform, allowing the playback of the converted speech. Among the techniques introduced in chapter 2 for speech analysis and reconstruction, STRAIGHT is the one chosen in [11] and many other VC works. Despite having its limitations, STRAIGHT shows good performance and flexibility, and thus it is adopted in this thesis as well.

The STRAIGHT model analyses an audio file and outputs an estimation of its spectrogram, $F0$ trajectory and aperiodicity. Admitting that the STRAIGHT Fast Fourier Transform (FFT) window size is at the default value of 2048, the spectrogram and aperiodicity are both represented by matrices of size `(1025, n_frames)`, corresponding to the real part of the signal. While the spectrogram and fundamental frequency components are general purpose features, the aperiodic component is specific for STRAIGHT and is defined as the ratio between the lower and upper smoothed spectral envelopes in the frequency domain [11].

Using full spectrograms to train a network might by overwhelming, since each spectrogram is very large (around 10 Mb for a 2-3 second utterance). This calls for a more efficient and compact representation of the spectrogram. The most popular spectrum features in the more recent VC research are the cepstral features. Simililarly to the original paper in [11], to represent the spectrum effectively and with sufficient detail to preserve speaker dependent information, 49-dimensional Mel general cepstrum are chosen. In addition, to avoid modelling utterance-dependent information, the energy component of the MGC is removed.

To extract the MGC features from the spectrograms outputted by STRAIGHT, we resort to the signal processing toolkit (SPTK) [71], which has the required functions to do it, as well as some other useful ones required further ahead in this work. The extraction is made via the `mgcep` function and a flowchart of the process is depicted in figure 3.8.

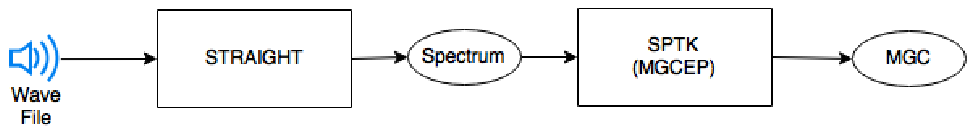


Figure 3.8: Mel Generalized Cepstra feature extraction process.

3.3.2 Frame Alignment

To train a neural network in a supervised manner, it is necessary to repeatively feed it train data pairs (x, y) in order to minimize a certain loss function. In our case, x will correspond to a matrix of source features, while y corresponds to a matrix representing the target features. To allow the computation of the loss function, the matrix representing the prediction of the target generated by our network must have the same shape as the target.

Although both source and target speakers utter the same sentence, the two speakers are not precisely synced. It may be the case that one speaker is slower than the other. This results in a different number of frames, and thus matrices of different sizes, which makes the computation of the loss function a problem, unless the model is capable of mapping different length sequences. Since the model proposed does not have this capability, it is necessary to align the features of both speakers, with respect to one of them, which will be used as reference. In this thesis, the target speaker is considered the reference for the alignment process.

Dynamic time warping is the method most suited for alignment tasks and thus being our choice, similar to what was done in [11]. To perform DTW we use another function of the SPTK toolkit: `dtw`. This function receives two separate input files and outputs a single file with the concatenation of the two aligned inputs. In the context of this framework, the inputs are two unaligned MGC features and the output is a concatenation of the two aligned MGCs (figure 3.9).

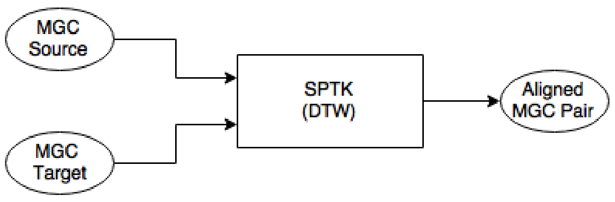


Figure 3.9: Frame alignment process.

3.3.3 Training Phase

Network Architecture

The DBRNN model proposed (either with LSTM or GRU cells) is built from several stacked bidirectional recurrent neural networks, and it is depicted in figure 3.10. The number of stacked layers varies depending on each experiment conducted, but from experimentation, in general it is around 6 layers as reported in [11]. In our implementation, the last layer of the network is called a time distributed dense layer, and it simply applies a linear transformation of a fully connected layer, with linear activation, to each time step of the hidden state incoming from the RNN stack. The purpose of this layer is to transform the hidden state of the last RNN layer onto a representation with a similar dimension to the target.

The network is trained as a whole and its objective is to project the MGC features into a progressively higher feature space that represents speaker dependent information. This idea is comparable to the work done in [50].

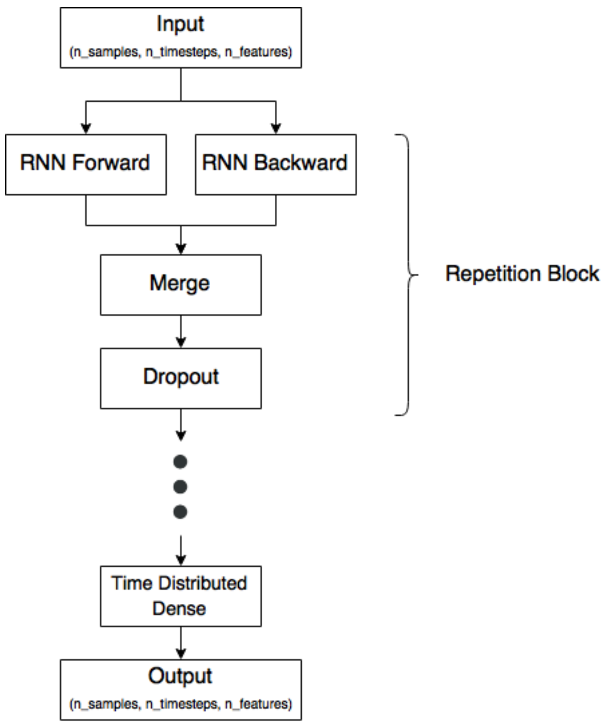


Figure 3.10: Deep Bidirectional Recurrent Neural Network architecture diagram. The repetition block will be repeated for a number of times for additional depth.

Loss Function

During training, the network processes an aggregate of samples at a time (also called a mini-batch), outputting $\hat{y}$, a prediction of y based on the current value of its weight matrices. When a prediction is made during training, a loss function is computed to evaluate how close the prediction is to the target. The gradients of the loss function with respect to each weight parameter are then computed to minimize the function.

Taking into account that the problem at hand is a regression problem, a suitable cost function is the Mean Squared Error (MSE)

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3.13}$$

where y_i and $\hat{y}_i$ are the i-th dimensions of the target and the model's output prediction, respectively.

Optimization

The basic algorithm to compute the weight matrices that lead to a minimum of the loss function is the very popular stochastic gradient descent (SGD) algorithm. However, SGD requires some fine tuning of its parameters. Since the complex nature of our problem does not allow for an exhaustive search of optimal parameters, choosing SGD is not ideal.

Some optimizers have adaptive parameter mechanisms, therefore discarding the need for fine tuning. This is the case of optimizers such as the RMSProp (unpublished) and ADAM [72], which are becoming very popular with RNNs.

Adaptive Moment estimation (ADAM) computes adaptive learning rates for each parameter. It stores an exponentially decaying average of past squared gradients v_t and past gradients m_t, which is similar to momentum:

$$v_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{3.14}$$

$$m_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{3.15}$$

m_t and v_t are estimates of the first (mean) and the second moment (uncentered variance) of the gradients, respectively.

The authors verified that, as m_t and v_t are initialized as zero vectors, they are biased towards zero. To counteract these biases, bias-corrected first and second moment estimates are used:

$$\tilde{v_t} = \frac{m_t}{1-\beta_1^t}, \tag{3.16}$$

$$\tilde{v_t} = \frac{v_t}{1-\beta_2^t}. \tag{3.17}$$

The algorithm then updates the network parameters as:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v_t}} + e}\hat{m_t}, \tag{3.18}$$

where e is a smoothing term that avoids division by zero and the θ represents the weight parameters.

In [72], the authors propose default values of 0.9 for β_1, 0.999 for β_2 and 10^{-8} for e, and proceed to compare the use of their optimizer against other adaptive learning algorithms, reaching favorable results for ADAM. For more in-depth knowledge of different machine learning optimizers, the reader is referred to [73].

Dropout Layer

After a few epochs of training any neural network, the loss function of the training set may keep diminishing, while the loss on the validation set starts to get worse. This phenomena is called overfitting, and it means that the network is learning specific information related to the training set, instead of information that can be generalized to other data.

In order to reduce overfitting, a simple regularization mechanism called dropout was introduced in

[74]. The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. During training, dropout samples from an exponential number of different “thinned” networks. At test time, it is easy to approximate the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights.

We implement this technique by generating random binary streams with a ‘do not keep‘ probability p and multiplying it by a layer output. The outputs that are not dropped out are then scaled by dividing them by the ’keep’ probability. This ensures that for any hidden unit, the expected output (under the distribution used to drop units at training time) is the same as the actual output at test time. The dropout (do not keep) probability chosen for the experiments performed in this thesis is of 0.1.

3.3.4 Conversion Phase

This thesis relies on STRAIGHT to synthesize speech and thus, it is necessary to meet its input requirements. STRAIGHT is capable of synthesizing speech from a spectrum, an $F0$ trajectory and speaker aperiodicity inputs. In this section, it is explained how to obtain and manipulate each of these features to output a converted audio file.

Spectrum

The network architecture with loaded trained weight parameters represents the conversion function $F(.)$, which allow us to process a source speaker’s MGC feature and convert it to another MGC, so that it approximates the target speaker. This operation corresponds to a forward pass of the full network.

Once the network’s forward pass outputs the converted MGC features, they are converted into a spectrum so that it can be fed into STRAIGHT.

It is important to point out that the dropout layer is only used during the training phase and not on the conversion phase, otherwise it would introduce undesired artifacts.

F0 trajectory and Aperiodicity

As previously hinted in figure 3.6 by following the data flow, the log $F0$ and aperiodic components are converted separately.

The $F0$ linear conversion is made using equation 2.9. This method maintains the source’s overall pitch trajectory but equalizes it with respect to the target’s $F0$ trajectory mean and standard deviation. Log $F0$ is used for this conversion and then is converted back to $F0$ for the reconstruction model’s input.

The aperiodic component is directly copied to synthesize the converted speech, since previous research shows that converting aperiodic component does not make statistically significant difference on the synthesized speech [75].

3.4 Summary

In this chapter, we introduce the concepts of recurrent neural networks and more specifically, its LSTM and GRU variations. These architectures play a critical part in the baseline voice conversion framework we are implementing. This framework is presented in figure 3.6 and involves STRAIGHT to analyze speech, MGC feature extraction and MGC alignment, in order to be able to perform conversion. Since we are mainly focusing in spectral mapping, the MGC features representing the spectrum are our main focus and will be converted by the network model represented in figure 3.10. This model should produce a new set of features that, together with an equalized version of the $F0$ trajectory and the source's aperiodic component, are processed again by STRAIGHT to synthesize the converted speech signal, completing the voice conversion process.

Chapter 4

Voice Conversion Using Sequence-To-Sequence Attention Models

4.1 Introduction

The success of the baseline approach in the voice conversion problem opens up new possibilities for other RNN based approaches that have not been tested up to the moment. One major error inducing factor in the baseline approach is its frame alignment step. More specifically, the dynamic time warping of the spectral features. When dealing with audio features, DTW repeats or deletes certain time frames in order to align both source and target features. This means that we are training a recurrent neural network with altered data, and therefore learning non existent temporal dependencies.

We propose bypassing the alignment process and directly learning the alignment within the neural network. Two recent techniques, growing popular in recent years, are the sequence-to-sequence and attention models. We will briefly discuss both of these approaches in this chapter, although the focus of this thesis is only on attention models.

This chapter has the following structure: Section 4.2 describes the sequence-to-sequence model concept; section 4.3 explains how soft attention models work; section 4.4 goes into detail on how we propose an attention sequence-to-sequence model to solve our VC problem; and finally section 4.6 summarizes the whole chapter, pointing out the most important concepts.

4.2 Sequence-To-Sequence Models

The encoder-decoder architecture, also known as sequence-to-sequence, was initially proposed by Cho *et al.* in the same work where the GRU cell was introduced [69]. The original model was proposed for a machine translation application, but its concept can be extended to other sorts of problems.

The sequence-to-sequence model learns to encode a variable-length sequence into a fixed-length vector representation and to decode that same fixed-length vector representation back into a variable-length sequence. It is called a sequence-to-sequence model because its input and output lengths are usually different.

If we consider the source sequence to be represented as $x = (x_1, \dots, x_{T_x})$, and the target sequence as $y = (y_1, \dots, y_{T_y})$, the encoder is a recurrent neural network that reads each time step x_t sequentially, and summarizes it into a fixed-length context vector c. This context vector should be able to summarize the whole sequence information. The decoder consists of another RNN that takes the context vector c as input, and progressively generates each time step of the output y_t, given the current hidden state h_t. However, both h_t and y_t generated by the decoder at each time step are conditioned by y_{t-1} and the context c. In order to train such a model, both encoder and decoder are jointly trained. An illustrative scheme of a simple sequence-to-sequence model architecture is shown in figure 4.1.

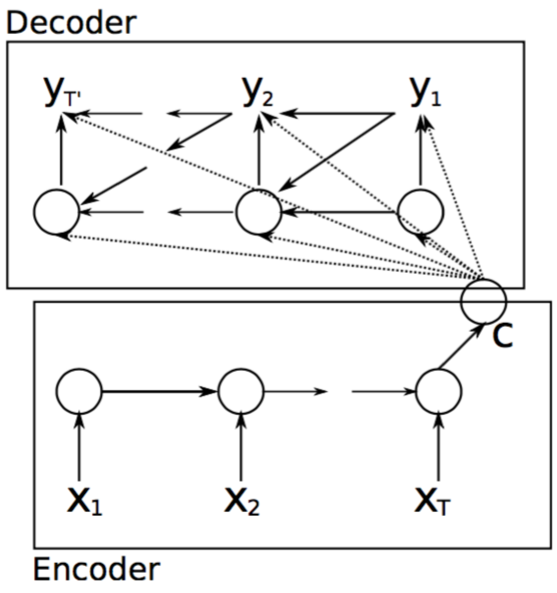


Figure 4.1: An illustration of the encoder-decoder architecture. Extracted from [69].

In the original paper concerning machine translation, the network is trained by maximizing the log-likelihood of the next word based on the information of the context vector, hidden states of the decoder and the current word. However, in the machine translation scenario, we have to deal with words that form a discrete output space. In a spectral mapping problem, we are dealing with a regression problem in a continuous input and output feature space, so using a mean squared error loss function would be more appropriate.

Another difference between scenarios is that, in [69], the authors deal with data that have a relative small number of time steps, especially when compared to the number of frames of an audio file as in our corpora. This can create difficulties for the network to condense all the information it needs to decode the sequence into a single context vector, as verified in [76]. For these reasons, although considered initially to solve our spectral mapping problem without resorting to frame alignment, the idea was abandoned as soon as we tested the network on a sinusoid prediction toy-problem. In this small one-dimensional regression toy-problem, the network revealed poor performance, and therefore made us divert our focus into attention models.

4.3 Attention-Based Models

4.3.1 Attention Mechanisms

Inspired in the human brain, the attention mechanism was introduced to allow neural networks to focus on specific parts of the input. Taking a real world example, when a subject is in a crowded room trying to have a conversation, he/she focuses on the speech coming from the person he/she is having the conversation with, ignoring the remaining noise. This person is making his/her conversational partner the focus of his/her attention. Humans constantly use their attention mechanism in a variety of situations to avoid being bombarded with overwhelming amounts of information.

When dealing with neural networks, the same concept can be applied. There are essentially two types of attention mechanisms in neural networks: visual (hard) attention and soft attention. The first one decides if a region is important or not in a binary fashion, whereas the second one is able to attribute probabilities to different input regions. In this thesis we will only deal with soft attention models.

One critical aspect of virtually all structured output tasks is that the structure of the output is intimately related to the structure of the input [77]. Therefore, aligning the input with the output is a central challenge. Soft attention mechanisms were proposed to tackle this challenge. Both soft and visual attention models have gained popularity recently in a wide variety of areas of research, such as handwriting synthesis [78], machine translation [79], image caption generation [80], visual object classification [81], and speech recognition [82].

The soft attention mechanism was originally proposed by Bahdanau *et. al.* in [79] with the purpose of improving the performance of the previous state-of-the-art encoder-decoder (sequence-to-sequence) architectures for machine translation. According to the authors, encoder-decoder architectures create a bottleneck in performance by having to encode all the information into a single context vector. Plus, the interpretability of the simple encoder-decoder is extremely low, as the context vector does not have a presupposed structure. Instead of trying to encode the whole input sequence into a context vector, the attention mechanism is introduced to allow the system to sequentially focus on different parts of the input, called the attended input. The selection of these parts is conditioned by the state of the system, which by itself is conditioned by the previous attended parts.

The purpose of attention mechanisms is twofold: to reduce computational effort of processing high dimensional inputs by focusing on smaller parts, and to allow the system to focus on distinct aspects of the input, leading to a more efficient extraction of relevant information. However, as the first may be true for some models, this does not mean an attention mechanism will always make a model run faster. In fact, in some recurrent neural network setup, it will actually worsen computation time because it makes the model analyse the whole input at each time step to compute that time step's attention.

4.3.2 Attention-based Decoder: General Framework

Attention-based decoder models are closely related to the sequence-to-sequence architecture since they also decode an input of length T_x into an output of length T_y. Usually, the input $x = (x_1, \dots, x_{T_x})$ initially goes through an encoder which outputs a sequential input representation $h = (h_1, \dots, h_{T_x})$ more suitable for the attention decoder to work with.

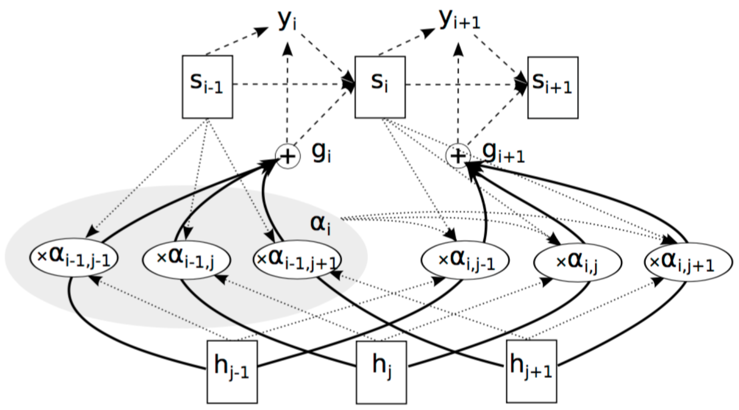


Figure 4.2: An illustration of the attention decoder architecture. Extracted from [82].

As depicted in figure 4.2, taking as input the encoded sequence h, the attention decoder first filters the information with an attention mechanism to obtain the attended input. This attention mechanism, or attention model, is a neural network that scores each input h_j with respect to the current hidden state of the decoder s_{i-1}. The scores e_{ij} are computed as

$$e_{ij} = a(s_{i-1}, h), \tag{4.1}$$

where a is an alignment model parametrized as a feed-forward neural network, jointly trained with the other components of the decoder, and the state s_{i-1} is associated with a recurrent neural network. This RNN, usually a GRU or an LSTM, unrolls for T_y time steps along with the target y. In [82], a is defined as:

$$e_{ij} = w^T \tanh(W s_{i-1} + V h_j + b), \tag{4.2}$$

where w and b are vectors, and W and V are matrices.

The a operation attributes a value to each score e_{ij} that does not have any lower or upper bound. In order to get well defined weights α_{ij}, also called the attention weights, a softmax function is computed on the scores to make them be non-negative and sum to 1 as:

$$\alpha_{i,j} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ij})}. \tag{4.3}$$

In order to filter the input, the attention weights α_{ij} are multiplied by the input and summed on the features dimension axis, obtaining what is called a glimpse g_i, defined as:

$$g_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{4.4}$$

Combining information from the previous RNN state s_{i-1}, the current glimpse g_i, and the previous target value y_{i-1}, a new state of the RNN s_i is computed as

$$s_i = Recurrency(s_{i-1}, y_{i-1}, g_i). \tag{4.5}$$

In equation 4.1 we considered a content-based attention. However, it is possible to have a location based attention if $e_{ij} = a(s_{i,j}, \alpha_{i-1})$ or even a hybrid attention version if $e_{ij} = a(s_{i,j}, h, \alpha_{i-1})$.

4.4 Proposed Framework: Deep sequence-to-sequence Attention Model

Inspired by [82], we propose a sequence-to-sequence attention model to tackle the problem of spectral mapping for spectral features of different lengths. We chose a content-based attention, both for simplicity and to allow the model to extract information from other, more distant, neighbouring frames if relevant. However, since we are dealing with long sequences that have a strong temporal correlation, a hybrid attention model would also make sense.

Our proposed architecture, which we call a deep sequence-to-sequence attention model, is depicted in figure 4.3, and makes use of the ideas proposed by the baseline work [11], along with the content-based attention mechanism already introduced. This architecture is constituted by stacked GRU encoders, similar to what is done in the baseline framework, followed by a GRU attention decoder. The GRU architecture is chosen over the LSTM, because of the lower computational requirements of the GRU architecture.

The encoders' purpose is to process the input into a high dimensional feature space, in order to facilitate the decoding process. The attention decoder receives a full context sequence from the stacked encoders, applies its attention mechanism, and decodes the sequence of length T_x into a sequence of length T_y. This layer is of critical importance in this architecture, because it decides in which parts of the input the decoder should focus, so that an alignment between source and target is possible.

Due to the high number of time steps of our spectral features, the encoders are bidirectional to better capture temporal dependencies in both ends of the sequences. However, because of the step by step prediction limitation, which we describe in 4.5, the attention decoder can not be bidirectional.

In this proposed approach, all the training conditions and regularization techniques used are similar to the ones introduced in Chapter 3. However, during the conversion stage, since now we map the source feature sequence to the length of the target sequence, instead of making use of the source's MGC energy component, now we use the target's energy component, if available. If the target is not available, as in a real application situation, it is also possible to generate a sequence with the source's

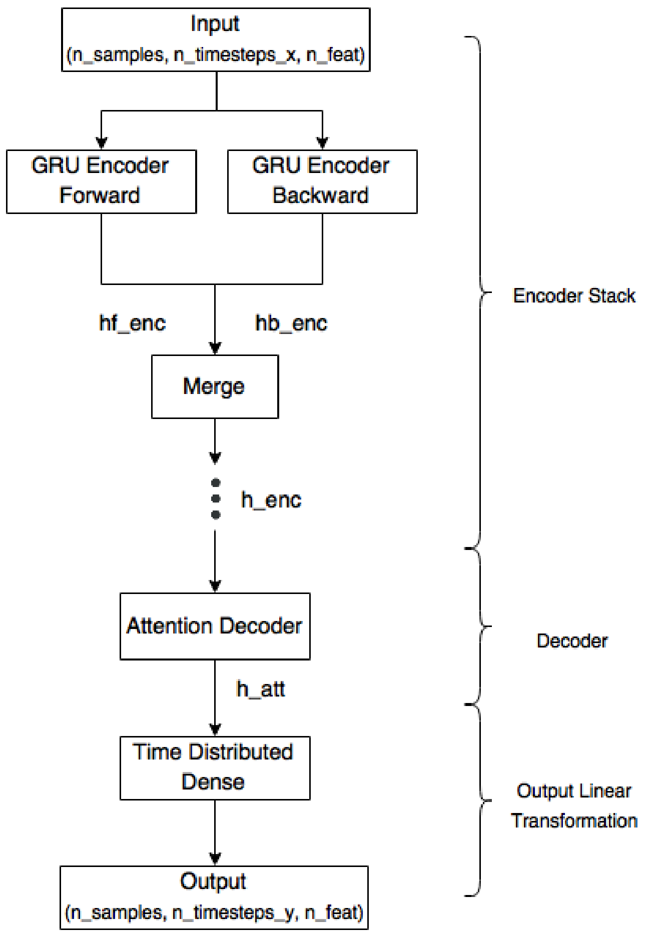


Figure 4.3: An illustration of the proposed architecture for the deep sequence to sequence attention model.

length and to use the source's MGC energy component.

4.5 Dealing With Prediction Errors

The attention decoder makes use of the ground truth, or target value, y, which is only accessible during the training process. If we desire to use the model to predict an utterance that is outside our training and validation set, this information is not available. In order to do a prediction, the model must be able to use the information of the output generated by itself, $\hat{y}$, and then use it recurrently step-by-step, until it generates the full sequence. If there were other layers after the decoder, the $\hat{y}$ prediction should be a result of a step by step forward pass through the decoder and the remaining layers after it. This method of prediction does not allow bidirectional layers for the decoder or any other layer after the decoder, since at each instant of time of prediction, both forward and backward cells would be processing a different time step.

During training we experimented with two approaches. The first one relies solely on the ground truth for training and validation, and then testing by feeding back the output predicted by the model to itself until the complete sequence is generated. The second approach consists in progressively alternating the ground truth with the model predictions during the training. The latter technique is based on the idea that, although the true value of y is always available during the process of training, it might be not desirable to always provide the ground truth. By doing this, the model may generate errors when making predictions with $\hat{y}$, and therefore propagate such errors throughout the whole sequence. For this reason, a way to make the model handle its own errors during prediction is proposed. The work of Bengio *et al.* in [83] introduces this exact same concept, which the authors call scheduled sampling, and inspired by it, we implemented it in our training. However, we are dealing with a continuous space of output features, instead of a discrete space as in the work where this technique was proposed.

Scheduled sampling consists of progressively alternating the model's y_{t-1} input between the ground truth value and the value predicted by itself $\hat{y}$. The use of either one of these values is decided through a flip of a coin, with probability ϵ_i for the ground truth value and $(1 - \epsilon_i)$ for the value predicted by the model. When $\epsilon_i = 1$, the model is trained only using true values of the target, and when $\epsilon_i = 0$, the model is trained in the same setting as inference. An illustration of the process is depicted in figure 4.4.

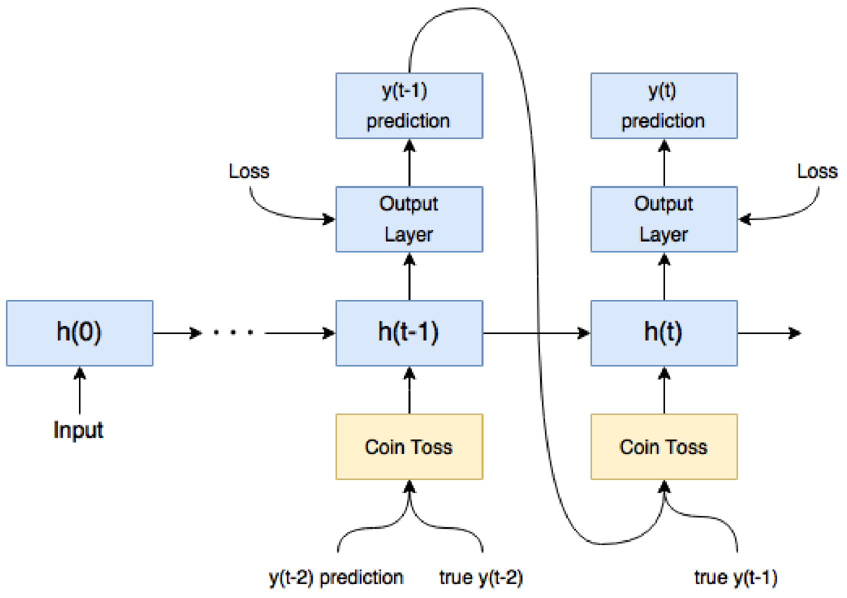


Figure 4.4: Illustration of the scheduled sampling process inspired by the work of [83].

In order to gradually introduce the changes between true and predicted values of y, the authors proposed controlling the probability ϵ_i in a decaying fashion, such that the model has a greater probability of using the true value of y_{t-1} at the beginning of training and then progressively decays towards $\hat{y}_{t-1}$ by the end of training. The different types of decay functions proposed by the authors are:

- **Linear decay**: $\epsilon_i = \max(\epsilon, k - ci)$, where $0 \leq \epsilon < 1$ is the minimum amount of truth to be given to the model, and k and c provide the offset and slope of the decay, which depend on the expected speed of convergence.

- **Inverse sigmoid decay**: $\epsilon_i = \frac{k}{k+\exp(i/k)}$ where $k \geq 1$ depends on the expected speed of convergence.

- **Exponential decay**: $\epsilon_i = k^i$ where $k < 1$ is a constant that depends on the expected speed of convergence.

In this thesis, we take ϵ_i to have an inverse sigmoid decay, as it has the smoothest transition behaviour. Examples of the different decay functions of ϵ_i with different values of k are depicted in 4.5.

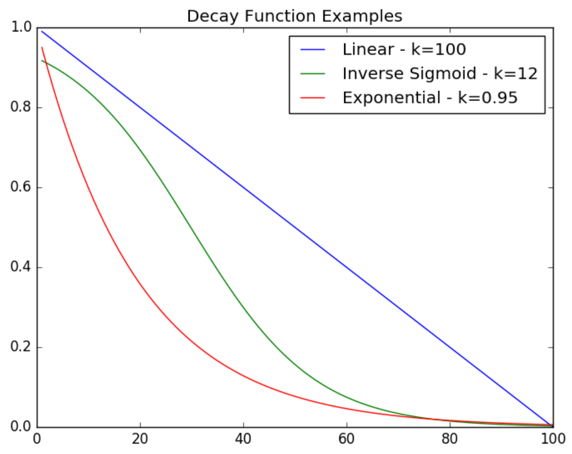


Figure 4.5: Different functions of ϵ_i for the curriculum learning approach. The vertical axis corresponds to the probability ϵ_i and the horizontal axis corresponds to the number of iterations.

4.6 Summary

In this chapter, we introduced neural networks that are able to decode a sequence of a set length into a sequence of a different length. This is the core idea of this thesis, and is intended to eliminate the explicit feature alignment step of the previously introduced baseline framework. Attention models were introduced in order to enable the decoder layer to 'focus' on different sections of the input during the generation of each step of the output sequence. The proposed neural network architecture to solve our voice conversion problem is a deep sequence-to-sequence attention model, consisting of a stack of bidirectional GRU encoders followed by a single attention enabled decoder. This architecture is trained jointly and resorts to a scheduled sampling technique, in order to mitigate the error propagation that may occur from the sequential prediction errors.

Chapter 5

Experiments

5.1 Corpora

The experiments reported in this thesis were made using the CMU ARCTIC databases [84]. These databases were constructed at the Language Technologies Institute of Carnegie Mellon University, as phonetically balanced, US English single speaker databases designed for unit selection speech synthesis research. These corpora are widely used for voice conversion research and are publicly available. For these reasons, they became the choice for this thesis. In addition, by using ARCTIC it is possible to compare the outcome of the experiments made with the state of the art research.

The databases consist of around 1130 utterances per speaker from non-copyrighted texts spoken by 7 different individuals, both male and female, in US English. Three other speakers are accented speakers. Some of them have a slightly different number of utterances, as illustrated in table 5.1.

Table 5.1: CMU ARCTIC databases details.

Speaker ID	Gender	Number of utterances	Version	Accented
bdl	Male	1132	0.95	No
slt	Female	1132	0.95	No
clb	Female	1132	0.95	No
rms	Male	1132	0.95	No
jmk	Male	1132	0.95	Yes (Canadian)
awb	Male	1137	0.90	Yes (Scottish)
ksp	Male	1132	0.95	Yes (Indian)

For each of the speakers, the databases have a vast range of speech related features and other metadata, which will not be used. For the scope of this thesis, we only require the wav files correspondent to each speaker's utterances. These utterances are 16 bit, mono waveforms, sampled at 16 kHz.

It is relevant to point out that some files of the accented speakers with the same file ID as the unaccented speakers do not match. For this reason, this thesis will only make use of the unaccented speaker databases.

For the experiments described in this thesis, we used the databases of the CMU ARCTIC related to the speakers bdl (male), rms (male), and slt (female). This way, it is possible to compare intra-gender conversions as well as inter-gender conversions. Our goal is to evaluate the capability of the voice conversion system to convert voices that have either slight or more radical differences. For each setup we perform experiments for bdl-rms (male to male) and bdl-slt (male to female) conversions.

The data set is divided into three separate subsets: a training set, a validation set, and a test set. We choose 80% of the total data set for training, 20% for validation, and 10% for testing. This corresponds to a total of 793 utterances for training, 226 utterances for validation, and 113 for testing. These proportions are common practice in machine learning. This amount of data is superior to the data used in the [11] baseline experiments, where the authors use 593 utterances for training and 119 for validations.

5.2 Deep Learning Framework

The initial experiments were ran in Python using built in functionalities of Keras [85], a deep learning library built on top of Theano [86]. Theano is a Python library that allows to define, optimize, and evaluate mathematical expressions, especially ones with multi-dimensional arrays [86]. Operations are built through symbolic expressions and chained in a computation graph, which allows to easily compute complex operations such as gradients. Due to its ‘under the hood‘ C code generation, Theano combines speeds rivaling hand-crafted C implementations with the flexibility and simplicity of Python.

The Keras library has inherent limitations due to its avoidance of direct Theano use, and more specifically, it does not support either sequence to sequence or attention models directly. Due to the architecture of Keras, it is not straightforward to make a custom layer implementation for a decoder. This was the reason for deciding to implement in Theano a full framework for RNNs with support for sequence-to-sequence and attention models.

The developed framework is a multi-purpose academic deep learning framework for recurrent neural networks, which allows the user to train his/her own model using a variety of provided layers. The main goal of this framework is to allow the user to perform more in-depth modifications of the code for a higher degree of flexibility.

In order to accelerate training, we performed all the experiments using an NVidia Titan X graphics processing unit (GPU) to enable faster matrix computations. Theano has native GPU support and therefore, our deep learning framework also makes use of this functionality.

We discuss some implementation details such as code structure, how a layer is defined and how built-in and user defined models work in appendix A.

5.3 Implementation Details

5.3.1 Feature Extraction

As described in Section 3.3.1 is done using STRAIGHT and SPTK. STRAIGHT is capable of analysing a speech signal and outputs a real spectrogram matrix, a fundamental frequency $F0$ vector and an aperiodic component matrix. To extract such information, we use STRAIGHT's MATLAB interface. An example of the code used is shown below as:

```
[x, fs] = audioread(wav_file);
f0 = MulticueF0v14(x,fs);
ap = exstraightAPind(x,fs,f0);
sp = exstraightspec(x,f0,fs);
```

The parameters in STRAIGHT were unchanged, thus using a 2048 Fast Fourier Transform (FFT) window length. However, as referred before, STRAIGHT only represents the real half of the spectrogram, outputting a 1025 dimensional spectrum. To extract the MGC features from the spectrogram, the SPTK `mgcep` command is ran with an α of 0.42, FTT length of 2048, option `-m 49` to indicate the order of the MGC features, and option `-q 3`, meaning that the input format is the L1 norm of the spectrogram, as it is the output format of STRAIGHT. The value of α depends on the sampling rate of the speech signals and, according to the SPTK manual, for a sampling rate of 16kHz, α should have a value of 0.42. An example of the command line for this function goes as following:

```
mgcep -a 0.42 -m 49 -l 2048 -q 3 spectrum_file.sp > mgc_file.mgc.
```

STRAIGHT has a MATLAB interface and SPTK runs exclusively via shell. To enable batch processing of all the utterances data, we built a Python script that iteratively calls MATLAB functions for the STRAIGHT analysis and bash commands for the SPTK MGC extraction.

5.3.2 Frame Alignment

The `dtw` function from SPTK aligns two binary files, and outputs another binary file with the concatenated aligned files. In order to use this function we have to provide the functions the argument `-l` to indicate the dimension of the input files as well as the paths for the target and source files. An example of the command line for this function goes as following:

```
dtw -l 49 target_file.mgc < source_file.mgc > aligned_file.dtw.
```

where the option `-l 49` is an indication of the features dimensions.

5.3.3 Input Pre-processing

Data normalization is a common practice in machine learning for performance improvement and it is done in the experiments described in [11], which we are following as baseline. One simple way to normalize data is by subtracting the mean and dividing by the standard deviation, as:

$$x_n = \frac{x - \mu}{\sigma}, \tag{5.1}$$

where x_n represents the normalized data, x is the raw data, μ is the data mean and σ is the data standard deviation. The objective is to get data with zero mean and unity standard deviation so that the optimizer can perform better and avoid precision errors. In our experiments, each MGC feature pair is normalized independently.

To enable mini-batch processing, another step is needed before the network receives its input: zero padding. In our deep learning framework, the network expects, at each iteration, a 3-dimensional tensor with size of `(n_samples, n_timesteps, n_features)`. The `n_samples` dimension corresponds to the batch size, `n_features` to the MGC features dimension and `n_timesteps` to the number of frames of each utterance. Since a mini-batch is a group of several utterances, it will not be possible to simply stack the matrices on top of each other due to the variation of the number of frames for each sample. Zero padding allows to easily overcome this problem. A maximum length, corresponding to the number of frames of the batch's longer utterance is chosen, and all the other samples in that batch are padded with zeros up to that size.

In order to minimize the zero padding for each batch, during the process of loading the data and creating the mini-batches, the samples are sorted with respect to the number of frames. By doing this, each mini-batch has a minimum variation in number of time steps and the network does not have to process as much silence corresponding to the padded zeros. This technique is called bucketing.

5.3.4 Masking

Zero padding the input solves one problem, but creates another one. By creating more time steps in the input, the RNNs take the zeros into consideration and try to create a model with this unnecessary information. In fact, for a network that deals with audio features, this may lead to audio artifacts. To overcome this problem, there is a technique called masking, which is commonly used within the machine learning field. A comparison between the same network being used with and without masking is made in section 5.4.

Masking consists of passing an extra input, called a mask. The mask can be thought of as a matrix of zeros with shape `(n_samples, n_timesteps)`, that for each column is filled in with an array of ones, with length corresponding to that sample's number of time steps.

When the mask is passed to the RNN, it is instructing the network to process regularly the time steps corresponding to a one in the mask and to copy its previous hidden state to the current time step when the mask corresponds to a zero. The loss function should be computed accordingly when masking is used in the network. The process of masking the cost can by summarized as computing the cost for each individual dimension of the predicted output, multiplying it by the mask and finally computing the mean over the masked cost array. By using a masked cost, the cost value won't be affected by the amount of zero padding as it would regularly be.

Throughout all the experiences in the remainder of this chapter we resort to the use of masking, except for the case when the advantages of using masking are being evaluated.

5.3.5 Conversion Process

To obtain converted speech, the model needs to be initialized and the trained weight parameters should be loaded. Once this is done, we get a function capable of transforming an input feature set from the source into a converted feature set.

Recalling that the input data of the network is normalized, the output must be adjusted. To accomplish this, the original values of mean and standard deviation of the source feature are added back in an inverse operation of equation 5.1:

$$y = y_n * \sigma + \mu \tag{5.2}$$

In addition, the network only handles MGCs without energy component, being necessary to add it back. As referred previously, the energy added to the converted MGC is the energy from the target speaker in the case of the sequence to sequence attention model approach and if available, otherwise it is simple the source's energy component. This way, the converted utterance can either maintain the intensity of the target or the source. The choice of either of them is made to maintain a matrix shape coherence.

To meet the spectrum input requirement of the reconstruction model, it is necessary to process the newly converted features. This spectrum should have the same window size used for the feature extraction (2048 FFT length). To perform the MGC to spectrum conversion, a function of the SPTK toolkit called `mgc2sp` is used. An example of the command line for this function goes as following:

```
mgc2sp -a 0.42 -m 49 -l 2048 -o 2 converted.mgc > converted.sp.
```

where the options `-a`, `-m` and `-l` should match the options of the `mgcep` extraction function.

Once all the MGC features are converted with the forward pass and the $F0$ component is equalized for both speakers, STRAIGHT synthesizes converted speech. An example of the MATLAB code to do so is shown below as:

```
sy = exstraightsynth(f0_converted,sp,ap,fs).
```

5.4 Objective Evaluation

In this section, we evaluate the different architectures and compare them using the Mel cepstral distortion as an objective evaluation metric. For the baseline models that output predictions of size different than the target, dynamic time warping between the converted and target features is used right before the evaluation. Performing this alignment is required since both matrices for the converted and target features need to have a similar number of time steps. The proposed sequence to sequence attention model, however, outputs converted utterances with the same length as the target, which eliminates the need for this step.

The Mel cepstral distortion evaluation function is performed using a command from the SPTK toolkit called `cdist`. A use case example is given as

```
cdist -m 49 converted.mgc target.mgc | dmp +f
```

where `-m` corresponds to the MGC order and the `dmp +f` pipe is used for user readability, converting a binary value into a float.

It is also relevant to point out that, since the energy component is directly extracted from either the source or the target features, the energy component from the MGC should be removed before evaluating, otherwise it will introduce undesired perturbations.

5.4.1 Baseline Approach: DBRNN

The first experiment to establish the baseline of our work is to test our implementation of the work proposed in [11]. In this paper, the authors propose the use of 6 layers of bidirectional LSTMs with hidden layer dimensions of [49, 128, 256, 256, 128, 49]. This configuration intends to progressively create a higher dimensional feature space from layer to layer, in order to try to better represent the speaker dependent information. Nevertheless, with neural networks, the choice of hidden sizes is most often an *ad-hod* process, and there is no report of tests with different configurations in the paper.

We tested the model with the originally proposed configuration to perform male to male and male to female conversions. Next, an experiment with a wider hidden size configuration of [128, 256, 256, 256, 256, 128] is performed, to evaluate if the model benefits from larger hidden sizes.

Table 5.2: Mel cepstral distortion values in dB for the baseline deep bidirectional long short-term memory recurrent neural network (DBLSTM) model for male-male and male-female conversions.

			MCD (dB)		
Model	Source	Target	Train	Validation	Test
DBLSTM	bdl (♂)	rms (♂)	5.82	6.58	6.54
DBLSTM Wide	bdl (♂)	rms (♂)	**5.72**	**6.57**	**6.51**
DBLSTM	bdl (♂)	slt (♀)	6.14	**6.41**	**6.47**

From table 5.2, it is possible to observe that we reach values of distortion relatively close to the ones presented in [11], which reported MCD of around 5.5 dB. This can be attributed to a variety of factors. The results from [11] are only relative to a male to female conversion with speakers AWB (male) and SLT (female). In this work, due to an utterance incompatibility related to the different versions of the CMU ARCTIC database, it was not possible to reproduce the exact same male to female conversion. Moreover, our models were trained for about 30 epochs, with a batch size of 10, until the early stopping condition was reached, which corresponds to about 10 hours. The baseline paper approach states 48 hours of training, which should correspond to more that 30 epochs, although the GPU the authors used has less computation power than the Titan X, and there are no reports of the batch size used.

With the results obtained from our experiments, it is possible to observe that on average, the DBLSTM model performs better on male to female conversion. This fact might be attributed to the larger gap in spectrogram differences between male and female individuals that facilitate the learning process, opposed to less pronounced changes in timbre as in the case of a male to male conversion.

Regarding the hidden sizes differences, it is observed that the model benefits from the larger hidden size configuration in the male to male conversion. Despite the differences in the MCD from the original configuration being minimal, this can translate in less artifacts and slightly better audio quality in informal listening tests. Experimentation with other wider configurations could be done with the architecture, but that has to be left for future work.

5.4.2 Baseline Approach: LSTM vs GRU

LSTMs have shown very good results on a wide variety of problems, but the recently proposed GRUs are stated to have similar results to LSTMs. In addition, due to the GRU more compressed architecture, it demands less usage of computational power and has slightly faster convergence times. In this set of experiments, we intended to compare the performances between the DBLSTM and the deep bidirectional gated recurrent unit (DBGRU) models.

Table 5.3: Mel cepstral distortion values comparison between the use of a GRU cell versus the use of an LSTM cell in the DBRNN model.

			MCD (dB)		
Model	Source	Target	Train	Validation	Test
DBLSTM	bdl (♂)	rms (♂)	**5.82**	**6.58**	**6.54**
DBGRU	bdl (♂)	rms (♂)	6.09	6.82	6.77

From table 5.3 it is possible to observe that the LSTM performed slightly better than the GRU. Nevertheless, the GRU may be ideal in a scenario with less computational power or less GPU memory, in order to obtain a similar performance to what is obtained with the usage of an LSTM, but with a lighter setup.

5.4.3 Baseline Approach: Masking vs No Masking

In the baseline architectures, the authors did not reveal the use of any masking techniques for the training of their model. Since the DBLSTM model processes batches of padded sequences, we argue that masking has a relevant role on improving the model's performance and avoiding modelling unnecessary information. We experimented with two different scenarios, one in which we mask the training and validation padded sequences, and one in which the sequences are padded but not masked. Both scenarios were tested using a DBLSTM model with the original baseline hidden size configuration.

Observing the results shown in table 5.4, it is clear that masking has a small impact in the model's performance, since it has shown lower Mel cepstral distortion values throughout all the sets it was tested on. Nevertheless, it is relevant to point out that the bucketing technique was used for both cases, and therefore sequences with similar lengths were batched together. Discarding this technique would emphasise the performance differences from the usage of masking.

Table 5.4: Mel cepstral distortion values comparison between the use of masking with the DBLSTM model.

			MCD (dB)		
Model	Source	Target	Train	Validation	Test
DBLSTM - masking	bdl (♂)	rms (♂)	**5.82**	**6.58**	**6.54**
DBLSTM - no masking	bdl (♂)	rms (♂)	5.85	6.61	6.55

5.4.4 Sequence To Sequence Attention Model

To test our proposed sequence to sequence attention model, we experimented with three different scenarios, and compare them with the best model obtained in the baseline experiments. All the scenarios perform male to male conversions for the same source and target speakers.

The first scenario (S1) allows the model prediction process to have the same information it has during training, *i.e.* we allow the model to access y_{t-1} information from the target. The purpose of this experiment is to evaluate the model's behaviour during training and evaluate how well it is learning from the data.

In the second scenario (S2) the model is not allowed to access y_{t-1} from the target, and has to output its own predictions $\hat{y}_{t-1}$ and use them iteratively to generate the whole sequence. The goal of this experiment is to evaluate the error propagation of the model throughout the whole sequence, due to the prediction errors.

Finally, in the third and last scenario (S3), the model from the first and second scenarios is retrained by progressively introducing the model's predictions instead of the target values of y_t, using a scheduled sampling technique. With this final scenario, the model's capacity of dealing with its own errors may be evaluated.

Table 5.5: Mel cepstral distortion values comparison male to male conversion using the proposed sequence to sequence attention model.

			MCD (dB)		
Model	Source	Target	Train	Validation	Test
DBLSTM Wide	bdl (♂)	rms (♂)	5.72	6.57	6.51
Seq2Seq Attention [S1]	bdl (♂)	rms (♂)	**3.70**	**3.84**	**3.92**
Seq2Seq Attention [S2]	bdl (♂)	rms (♂)	16.48	17.19	16.29
Seq2Seq Attention [S3]	bdl (♂)	rms (♂)	14.99	16.11	16.25

The results from the experiments are presented in table 5.5. The sequence to sequence attention models have 6-layer of depth and were trained with a [128, 256, 256, 256, 256, 128] hidden size configuration. Training these models took about 60 epochs and a total of 50 hours, using a batch size of 10 sentences on an NVidia Titan X GPU.

In the first scenario (S1), it is observed that the model outperforms the baseline approach, reaching MCD values that are about half the distortions from the DBLSTM model (in dB). Nonetheless, these results can only be used as an indicator of how well the model has learned the data, which has shown to be promising, although in a real prediction situation, it is not possible to access the true values of y.

In the second scenario (S2), it is clearly noticeable that the model struggles and does not handle well its own errors. The converted speech can barely be identified as speech, as it sounds muffled. This behaviour can be associated with the model relying too much on a correct input coming in from y_{t-1}, which can immediately lead to errors in the first few time steps and propagate throughout the remainder of the sequence. It is important to keep in mind that we are dealing with a number of time steps that can vary between 2000 and 8000, depending on the length of the sentences. Compared to the initial applications of attention models for machine translation that dealt with a few dozen time steps, this is a considerable gap.

In the third and final scenario (S3), the model benefits from the introduction of its own predictions during training with the scheduled sampling technique and the distortion is reduced in about 1 dB. However, the improvements are not substantial, especially if compared to the initial performance of the system in the first scenario. The final converted audio from these experiments still sound like a babbling of some sort, and the language content is still not perceptible.

5.5 Discussion

During the first half of these experiments we have successfully reproduced the baseline model and proposed a few minor modifications that can be useful in scenarios where there are hardware or time limitations, with the use of the GRU, and improved the model's performance with a different hidden size configuration. Nevertheless, it is important to point out that despite the fact of these voice conversion systems based on deep learning have been demonstrating great success regarding performance, they have a few disadvantages when compared to other, more conventional systems. To start with, we are still dealing with a one-to-one type of voice conversion, meaning that we would require to train a different model for each pair of speakers that we desire to convert. Next, there is the training time factor that is far superior when compared to some other VC methods, and finally, there is the need to weigh the considerable amount of parallel data these systems require in order to be trained.

Despite not solving most of the problems exposed, the sequence to sequence attention model proposed was motivated by the problem of the feature alignment requirement in the baseline approach, which introduced errors in the data that might lead to a deprecation in the model's performance. In addition, there is no report of use of these type of models to solve problems in a continuous space, which increased the motivation for our investigation.

With the deep sequence to sequence attention model proposed, we verified from the experiments that the model is capable of learning from the data with results that surpass the performance of the baseline approach. However, these results are only valid for a situation where the model can access some information from the target, an impossible situation in a real world application. In a situation in which the target information is not available, the model will necessarily generate the output sequence one step at a time, and use its own prediction as target information. If the first few iterations of the generation process include an error factor, the model will be led by this error factor and augment it, since each following time step will have its own error factor.

This error propagation is mitigated in a discrete problem situation by maximizing the likelihood of each candidate for the next output y_t and choosing accordingly. It is even possible to perform a beam search for the N best output candidates and allow the model to make a more informed decision for its output. However, in a continuous feature space, as the search space is infinitely large, it becomes more difficult to use this technique.

Our architecture, as currently presented, is not capable of using its own predictions to generate a real valued output prediction. This can be caused by a variety of factors from the architecture. To start with, the fact of the attention is not being conditioned by the location, in order to be smoothed and be more continuous in time, may be affecting performance. Changing from a pure content based attention to a hybrid attention mechanism can help improve results. Another factor is the robustness of the architecture that deals with the input coming from the target. Introducing a more robust neural network able to handle both values from the ground truth and the error prone predictions may also contribute to the model's improvement. An example of an utterance predicted from our model is presented in figure 5.1. It is possible to observe a repeating pattern, which is an indicator of the model's poor performance.

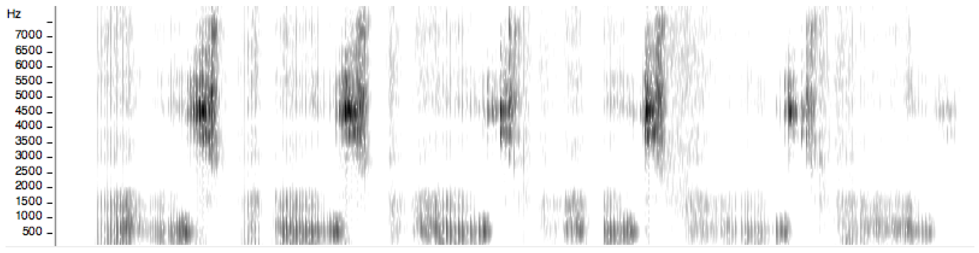


Figure 5.1: Spectrogram of a converted utterance from a sequence to sequence attention model trained with scenario S3.

There are a few more techniques that could be experimented with our model in order to solve this problem. However, due to the limited time a Master thesis imposes, this research has to be left as future work. With this in mind, our experiments do not provide enough evidence to discard this model for continuous inputs just yet, and rather provide extra motivation to explore possible future solutions to solve the problem of regression for sequences with different lengths for real valued data. We will discuss some of the possible techniques that can be applied in order to try to solve this problem in section 6.2.

Chapter 6

Conclusions

6.1 Contributions

The objective of this thesis was to contribute to the development of a voice conversion system with deep learning techniques that eliminate the requirement for alignment the source and target features.

We started by exploring the state of the art, selecting one of the most promising techniques related to deep learning, and implemented it. The baseline technique consisted of a stack of bidirectional LSTM recurrent neural networks that were intended to map the aligned input spectral features into a target.

The fact that the implementation of the baseline approach relied on the performance of the dynamic time warping algorithm provided the motivation to explore alternatives that could handle an alignment within the model itself. By taking inspiration from the current state of the art in machine translation, which resorts to the encoder decoder and attention architectures, we explored the same model concepts and applied it to our spectral mapping problem. However, there were some significant conceptual differences between the machine translation and spectral mapping problems, such as the real valued outputs and the considerably higher number of time steps.

Our architecture relies on a stack of GRU encoders to encode the sequence into a higher dimensional representation, also known as a context. The context is decoded step by step by a decoder with an attention mechanism so that the model can focus on different interest points from the input. While doing this, the model is at the same time aligning both input and output sequences.

All the experiments from both the implemented models were performed on the CMU ARCTIC corpus and the results were evaluated using a Mel Cepstral Distortion objective measure. It was observed that our implementation of the baseline approach achieved very similar results to the ones reported in the original paper. However, no subjective measure was performed for comparison.

The experiments with the sequence to sequence attention model revealed promising results during training, when allowed to peek at the target information. Nonetheless, the experiments in a full prediction environment where target information was not accessible, turned out to have a poor performance. These results were attributed to the difficulties in error handling during the prediction stage, together with the considerable number of time steps. Despite this turn of events, there is still not enough evidence to disprove the success of a sequence to sequence model with an attention mechanism to deal with regression problem in a continuous feature space, as there are more experiments that should be conducted.

While implementing the baseline and proposed models, we developed a Python framework based on Theano to support recurrent neural networks, sequence to sequence models and attention models. This framework will be available for an academic environment to allow anyone to easily apply these type of models to their own problems.

Ultimately, the two main contributions of this thesis are the study of the behaviour of attention models with real valued inputs and outputs, on which there were still no reports available, and the development of the sequence to sequence Theano framework.

6.2 Future Work

There are a few possible future work directions this thesis can take. Taking into account the further development of the model concept we proposed, we suggest a few techniques that can be tried out to help to mitigate the error propagation problem:

- **Hybrid attention**: the attention mechanism could benefit from a smoothing, or a hybrid attention mechanism, in order to block the attention weights to disperse through further neighbouring frames which can cause problems.
- **Error handling network**: to learn from the error of the model predicted and to prevent the whole model to rely on the previous target input, we suggest the introduction of a separate neural network just to deal with the prediction errors and minimize the error propagation throughout the sequence.
- **Output Conditioning**: in order to narrow down the output decisions, we suggest to add a text representation to condition the output. Work from [87] has reportedly achieved successful results decoding a real-valued sequence with output conditioning, by using LSTMs. The same concept could be applied to voice conversion.
- **Discrete Features**: another possibility would be to quantize the output features to be able to apply the same decision techniques during the prediction phase, such as beam search. Techniques such as vector quantization could be applied to achieve a discrete feature space. However, this approach might introduce other limitations, due to the way the quantization is done. In comparison with the error introduced by the DTW, the quantization may do more harm than good.
- **Alignment Information**: since the model's behaviour hints that the information provided is not enough to deal with the predictions, it would be interesting to explore the possibility of providing just enough information about the start and end of each phoneme. This way, the model could improve its learning ability in deciding whether to trust its previous prediction or not. This information wouldn't affect significantly the model's performance, since this information can easily be compressed.

Taking our focus away from the sequence to sequence models, there are a few interesting new developments that can be interesting for the future of voice conversion systems with deep learning. Recently, WaveNet [88] was proposed by Google's Deepmind, presenting a revolutionary state of the art technique for text-to-speech synthesis that outperforms every other current techniques. It would be interesting to explore WaveNet's architecture and develop a voice conversion system on top of it. Moreover, a many-to-one voice conversion framework using DBLSTMs was proposed in [89], using Phonetic PosteriorGrams (PPGs) obtained from a speaker-independent automatic speech recognition (SI-ASR) system and without the need for parallel data, which might be worth looking into as well.

The developed deep learning framework is currently under development in order to make it more robust to user errors, improve overall code quality and user experience. The framework should be publicly available via github soon.

Bibliography

[1] Z. Wu. *Spectral Mapping for Voice Conversion*. PhD thesis, Nanyang Technological University, 3 2015.

[2] E. Moulines and F. Charpentier. Neuropeech '89 pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones. *Speech Communication*, 9(5):453 – 467, 1990. ISSN 0167-6393. URL `http://www.sciencedirect.com/science/article/pii/016763939090021Z`.

[3] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, April 1975. ISSN 0018-9219.

[4] Y. Stylianou. Applying the harmonic plus noise model in concatenative speech synthesis. *IEEE Trans. Speech and Audio Processing*, 9(1):21–29, 2001. URL `http://dx.doi.org/10.1109/89.890068`.

[5] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigné. Restructuring speech representations using a pitch-adaptive time–frequency smoothing and an instantaneous-frequency-based {F0} extraction: Possible role of a repetitive structure in sounds1. *Speech Communication*, 27(3–4):187 – 207, 1999. ISSN 0167-6393. URL `http://www.sciencedirect.com/science/article/pii/S0167639398000855`.

[6] E. Helander, T. Virtanen, J. Nurminen, and M. Gabbouj. Voice conversion using partial least squares regression. *IEEE Trans. Audio, Speech & Language Processing*, 18(5):912–921, 2010. URL `http://dx.doi.org/10.1109/TASL.2010.2041699`.

[7] T. Toda, A. W. Black, and K. Tokuda. Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory. *IEEE Trans. Audio, Speech & Language Processing*, 15(8):2222–2235, 2007. URL `http://dx.doi.org/10.1109/TASL.2007.907344`.

[8] S. Desai, E. V. Raghavendra, B. Yegnanarayana, A. W. Black, and K. Prahallad. Voice conversion using artificial neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2009, 19-24 April 2009, Taipei, Taiwan*, pages 3893–3896, 2009. URL `http://dx.doi.org/10.1109/ICASSP.2009.4960478`.

[9] E. Helander, H. Silén, T. Virtanen, and M. Gabbouj. Voice conversion using dynamic kernel partial least squares regression. *IEEE Trans. Audio, Speech & Language Processing*, 20(3):806–817, 2012. URL `http://dx.doi.org/10.1109/TASL.2011.2165944`.

[10] D. Saito, S. Watanabe, A. Nakamura, and N. Minematsu. Statistical voice conversion based on noisy channel model. *IEEE Trans. Audio, Speech & Language Processing*, 20(6):1784–1794, 2012. URL `http://dx.doi.org/10.1109/TASL.2012.2188628`.

[11] L. Sun, S. Kang, K. Li, and H. M. Meng. Voice conversion using deep bidirectional long short-term memory based recurrent neural networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2015, South Brisbane, Queensland, Australia, April 19-24, 2015*, pages 4869–4873, 2015. URL `http://dx.doi.org/10.1109/ICASSP.2015.7178896`.

[12] K. Tokuda, T. Kobayashi, T. Masuko, and S. Imai. Mel-generalized cepstral analysis - a unified approach to speech spectral estimation. In *The 3rd International Conference on Spoken Language Processing, ICSLP 1994, Yokohama, Japan, September 18-22, 1994*, 1994. URL `http://www.isca-speech.org/archive/icslp_1994/i94_1043.html`.

[13] T. Fukada, K. Tokuda, T. Kobayashi, and S. Imai. An adaptive algorithm for mel-cepstral analysis of speech, 1992.

[14] D. Erro and A. Moreno. Weighted frequency warping for voice conversion. In *INTERSPEECH 2007, 8th Annual Conference of the International Speech Communication Association, Antwerp, Belgium, August 27-31, 2007*, pages 1965–1968, 2007. URL `http://www.isca-speech.org/archive/interspeech_2007/i07_1965.html`.

[15] A. Kain and M. W. Macon. Spectral voice conversion for text-to-speech synthesis. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98, Seattle, Washington, USA, May 12-15, 1998*, pages 285–288, 1998. URL `http://dx.doi.org/10.1109/ICASSP.1998.674423`.

[16] L. R. Rabiner and B. Juang. *Fundamentals of speech recognition*. Prentice Hall signal processing series. Prentice Hall, 1993. ISBN 978-0-13-015157-5.

[17] E. Helander, J. Schwarz, J. Nurminen, H. Silén, and M. Gabbouj. On the impact of alignment on voice conversion performance. In *INTERSPEECH 2008, 9th Annual Conference of the International Speech Communication Association, Brisbane, Australia, September 22-26, 2008*, pages 1453–1456, 2008. URL `http://www.isca-speech.org/archive/interspeech_2008/i08_1453.html`.

[18] H. Ney, D. Sündermann, A. Bonafonte, and H. Höge. A first step towards text-independent voice conversion. In *INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing, Jeju Island, Korea, October 4-8, 2004*, 2004. URL `http://www.isca-speech.org/archive/interspeech_2004/i04_1173.html`.

[19] D. Erro, A. Moreno, and A. Bonafonte. INCA algorithm for training voice conversion systems from nonparallel corpora. *IEEE Trans. Audio, Speech & Language Processing*, 18(5):944–953, 2010. URL http://dx.doi.org/10.1109/TASL.2009.2038669.

[20] H. Ye and S. J. Young. Voice conversion for unknown speakers. In *INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing, Jeju Island, Korea, October 4-8, 2004*, 2004. URL http://www.isca-speech.org/archive/interspeech_2004/i04_1161.html.

[21] Y. Stylianou, O. Cappé, and E. Moulines. Continuous probabilistic transform for voice conversion. *IEEE Trans. Speech and Audio Processing*, 6(2):131–142, 1998. URL http://dx.doi.org/10.1109/89.661472.

[22] D. T. Chappell and J. H. L. Hansen. Speaker-specific pitch contour modeling and modification. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98, Seattle, Washington, USA, May 12-15, 1998*, pages 885–888, 1998. URL http://dx.doi.org/10.1109/ICASSP.1998.675407.

[23] E. Helander and J. Nurminen. A novel method for prosody prediction in voice conversion. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2007, Honolulu, Hawaii, USA, April 15-20, 2007*, pages 509–512, 2007. URL http://dx.doi.org/10.1109/ICASSP.2007.366961.

[24] D. Lolive, N. Barbot, and O. Boëffard. Pitch and duration transformation with non parallel data. In *4th conference of Speech Prosody*, pages 111–114, 2008. URL https://hal.inria.fr/hal-00987810.

[25] Z. Inanoglu. Transforming pitch in a voice conversion framework, 2003.

[26] R. Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, April 1984. ISSN 0740-7467.

[27] M. Abe, S. Nakamura, K. Shikano, and H. Kuwabara. Voice conversion through vector quantization. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 655–658 vol.1, Apr 1988.

[28] H. Hwang, Y. Tsao, H. Wang, Y. Wang, and S. Chen. A study of mutual information for gmm-based spectral conversion. In *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, pages 78–81, 2012. URL http://www.isca-speech.org/archive/interspeech_2012/i12_0078.html.

[29] Y. Uto, Y. Nankaku, T. Toda, A. Lee, and K. Tokuda. Voice conversion based on mixtures of factor analyzers. In *INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17-21, 2006*, 2006. URL http://www.isca-speech.org/archive/interspeech_2006/i06_2076.html.

[30] H. Hwang, Y. Tsao, H. Wang, Y. Wang, and S. Chen. Incorporating global variance in the training phase of gmm-based voice conversion. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA 2013, Kaohsiung, Taiwan, October 29 - November 1, 2013*, pages 1–6, 2013. URL `http://dx.doi.org/10.1109/APSIPA.2013.6694179`.

[31] Y. Qiao, T. Tong, and N. Minematsu. A study on bag of gaussian model with application to voice conversion. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, pages 657–660, 2011. URL `http://www.isca-speech.org/archive/interspeech_2011/i11_0657.html`.

[32] H. Valbret, E. Moulines, and J. P. Tubach. Voice transformation using psola technique. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 145–148 vol.1, Mar 1992.

[33] D. Erro, E. Navas, and I. Hernáez. Parametric voice conversion based on bilinear frequency warping plus amplitude scaling. *IEEE Trans. Audio, Speech & Language Processing*, 21(3):556–566, 2013. URL `http://dx.doi.org/10.1109/TASL.2012.2227735`.

[34] Z. Shuang, R. Bakis, S. Shechtman, D. Chazan, and Y. Qin. Frequency warping based on mapping formant parameters. In *INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17-21, 2006*, 2006. URL `http://www.isca-speech.org/archive/interspeech_2006/i06_1768.html`.

[35] E. Godoy, O. Rosec, and T. Chonavel. Voice conversion using dynamic frequency warping with amplitude scaling, for parallel or nonparallel corpora. *IEEE Trans. Audio, Speech & Language Processing*, 20(4):1313–1323, 2012. URL `http://dx.doi.org/10.1109/TASL.2011.2177820`.

[36] D. Sundermann and H. Ney. VTLN-based voice conversion. In *Signal Processing and Information Technology, 2003. ISSPIT 2003. Proceedings of the 3rd IEEE International Symposium on*, pages 556–559, Dec 2003.

[37] D. Erro, T. Polyakova, and A. Moreno. On combining statistical methods and frequency warping for high-quality voice conversion. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2008, March 30 - April 4, 2008, Caesars Palace, Las Vegas, Nevada, USA*, pages 4665–4668, 2008. URL `http://dx.doi.org/10.1109/ICASSP.2008.4518697`.

[38] S. H. Mohammadi and A. Kain. Transmutative voice conversion. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 6920–6924, 2013. URL `http://dx.doi.org/10.1109/ICASSP.2013.6639003`.

[39] V. Popa, J. Nurminen, and M. Gabbouj. A novel technique for voice conversion based on style and content decomposition with bilinear models. In *INTERSPEECH 2009, 10th Annual Conference*

of the International Speech Communication Association, Brighton, United Kingdom, September 6-10, 2009, pages 2655–2658, 2009. URL http://www.isca-speech.org/archive/interspeech_2009/i09_2655.html.

[40] X. Tian, Z. Wu, S. W. Lee, and E. Chng. Correlation-based frequency warping for voice conversion. In *The 9th International Symposium on Chinese Spoken Language Processing, Singapore, September 12-14, 2014*, pages 211–215, 2014. URL http://dx.doi.org/10.1109/ISCSLP.2014.6936725.

[41] A. J. Hunt and A. W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings, ICASSP '96, Atlanta, Georgia, USA, May 7-10, 1996*, pages 373–376, 1996. URL http://dx.doi.org/10.1109/ICASSP.1996.541110.

[42] D. Sündermann, H. Höge, A. Bonafonte, H. Ney, A. W. Black, and S. S. Narayanan. Text-independent voice conversion based on unit selection. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing, ICASSP 2006, Toulouse, France, May 14-19, 2006*, pages 81–84, 2006. URL http://dx.doi.org/10.1109/ICASSP.2006.1659962.

[43] T. Dutoit, A. Holzapfel, M. Jottrand, A. Moinet, J. Pérez, and Y. Stylianou. Towards a voice conversion system based on frame selection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2007, Honolulu, Hawaii, USA, April 15-20, 2007*, pages 513–516, 2007. URL http://dx.doi.org/10.1109/ICASSP.2007.366962.

[44] Z. Wu, T. Virtanen, T. Kinnunen, E. Chng, and H. Li. Exemplar-based unit selection for voice conversion utilizing temporal information. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, pages 3057–3061, 2013. URL http://www.isca-speech.org/archive/interspeech_2013/i13_3057.html.

[45] M. Narendranath, H. A. Murthy, S. Rajendran, and B. Yegnanarayana. Transformation of formants for voice conversion using artificial neural networks. *Speech Communication*, 16(2):207–216, 1995. URL http://dx.doi.org/10.1016/0167-6393(94)00058-I.

[46] T. Watanabe, T. Murakami, M. Namba, T. Hoya, and Y. Ishida. Transformation of spectral envelope for voice conversion based on radial basis function networks. In *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002, Denver, Colorado, USA, September 16-20, 2002*, 2002. URL http://www.isca-speech.org/archive/icslp_2002/i02_0285.html.

[47] L. Chen, Z. Ling, Y. Song, and L. Dai. Joint spectral distribution modeling using restricted boltzmann machines for voice conversion. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, pages 3052–3056, 2013. URL http://www.isca-speech.org/archive/interspeech_2013/i13_3052.html.

[48] L. Deng and D. Yu. Deep learning: Methods and applications. *Foundations and Trends in Signal Processing*, 7(3-4):197–387, 2014. URL `http://dx.doi.org/10.1561/2000000039`.

[49] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. URL `http://dx.doi.org/10.1162/neco.2006.18.7.1527`.

[50] T. Nakashika, R. Takashima, T. Takiguchi, and Y. Ariki. Voice conversion in high-order eigen space using deep belief nets. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*, pages 369–372, 2013. URL `http://www.isca-speech.org/archive/interspeech_2013/i13_0369.html`.

[51] F. Xie, Y. Qian, Y. Fan, F. K. Soong, and H. Li. Sequence error (SE) minimization training of neural network for voice conversion. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2283–2287, 2014. URL `http://www.isca-speech.org/archive/interspeech_2014/i14_2283.html`.

[52] Z. Wu, E. Chng, and H. Li. Conditional restricted boltzmann machine for voice conversion. In *2013 IEEE China Summit and International Conference on Signal and Information Processing, ChinaSIP 2013, Beijing, China, July 6-10, 2013*, pages 104–108, 2013. URL `http://dx.doi.org/10.1109/ChinaSIP.2013.6625307`.

[53] L. Chen, Z. Ling, L. Liu, and L. Dai. Voice conversion using deep neural networks with layer-wise generative training. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 22(12):1859–1872, 2014. URL `http://dx.doi.org/10.1109/TASLP.2014.2353991`.

[54] T. Nakashika, T. Takiguchi, and Y. Ariki. Voice conversion using speaker-dependent conditional restricted boltzmann machine. *EURASIP J. Audio, Speech and Music Processing*, 2015:8, 2015. URL `http://dx.doi.org/10.1186/s13636-014-0044-3`.

[55] T. Nakashika, T. Takiguchi, and Y. Ariki. Voice conversion using RNN pre-trained by recurrent temporal restricted boltzmann machines. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 23(3):580–587, 2015. URL `http://dx.doi.org/10.1109/TASLP.2014.2379589`.

[56] G. W. Taylor, G. E. Hinton, and S. T. Roweis. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 1345–1352, 2006. URL `http://papers.nips.cc/paper/3078-modeling-human-motion-using-binary-latent-variables`.

[57] B. Kosko. Bidirectional associative memories. *IEEE Trans. Systems, Man, and Cybernetics*, 18(1):49–60, 1988. URL `http://dx.doi.org/10.1109/21.87054`.

[58] S. H. Mohammadi and A. Kain. Voice conversion using deep neural networks with speaker-independent pre-training. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South*

Lake Tahoe, NV, USA, December 7-10, 2014, pages 19–23, 2014. URL `http://dx.doi.org/10.1109/SLT.2014.7078543`.

[59] T. Nakashika, T. Takiguchi, and Y. Ariki. High-order sequence modeling using speaker-dependent recurrent temporal restricted boltzmann machines for voice conversion. In *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, pages 2278–2282, 2014. URL `http://www.isca-speech.org/archive/interspeech_2014/i14_2278.html`.

[60] I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted boltzmann machine. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1601–1608, 2008. URL `http://papers.nips.cc/paper/3567-the-recurrent-temporal-restricted-boltzmann-machine`.

[61] R. Takashima, T. Takiguchi, and Y. Ariki. Exemplar-based voice conversion in noisy environment. In *2012 IEEE Spoken Language Technology Workshop (SLT), Miami, FL, USA, December 2-5, 2012*, pages 313–317, 2012. URL `http://dx.doi.org/10.1109/SLT.2012.6424242`.

[62] Z. Wu, T. Virtanen, E. Chng, and H. Li. Exemplar-based sparse representation with residual compensation for voice conversion. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 22(10): 1506–1521, 2014. URL `http://dx.doi.org/10.1109/TASLP.2014.2333242`.

[63] Z. Wu, T. Virtanen, T. Kinnunen, E. S. Chng, and H. Li. Exemplar-based voice conversion using non-negative spectrogram deconvolution. In *The Eighth ISCA Tutorial and Research Workshop on Speech Synthesis, Barcelona, Spain, August 31-September 2, 2013*, pages 201–206, 2013. URL `http://www.isca-speech.org/archive/ssw8/ssw8_201.html`.

[64] Z. Wu, E. Chng, and H. Li. Exemplar-based voice conversion using joint nonnegative matrix factorization. *Multimedia Tools Appl.*, 74(22):9943–9958, 2015. URL `http://dx.doi.org/10.1007/s11042-014-2180-2`.

[65] C. Olah. Understanding LSTM networks, 2015. URL `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[66] Y. Bengio, P. Y. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994. URL `http://dx.doi.org/10.1109/72.279181`.

[67] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

[68] R2RT. Written memories: Understanding, deriving and extending the LSTM, 2016. URL `http://r2rt.com/written-memories-understanding-deriving-and-extending-the-lstm.html`.

[69] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734, 2014. URL `http://aclweb.org/anthology/D/D14/D14-1179.pdf`.

[70] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL `http://arxiv.org/abs/1412.3555`.

[71] SPTK Working Group. Speech Signal Processing Toolkit (SPTK). `http://sp-tk.sourceforge.net/`, 2015.

[72] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[73] S. Ruder. An overview of gradient descent optimization algorithms, 2016. URL `http://sebastianruder.com/optimizing-gradient-descent/`.

[74] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[75] Y. Ohtani, T. Toda, H. Saruwatari, and K. Shikano. Maximum likelihood voice conversion based on GMM with STRAIGHT mixed excitation. In *INTERSPEECH 2006 - ICSLP, Ninth International Conference on Spoken Language Processing, Pittsburgh, PA, USA, September 17-21, 2006*, 2006. URL `http://www.isca-speech.org/archive/interspeech_2006/i06_1681.html`.

[76] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014. URL `http://arxiv.org/abs/1409.1259`.

[77] K. Cho, A. C. Courville, and Y. Bengio. Describing multimedia content using attention-based encoder-decoder networks. *IEEE Trans. Multimedia*, 17(11):1875–1886, 2015. URL `http://dx.doi.org/10.1109/TMM.2015.2477044`.

[78] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL `http://arxiv.org/abs/1308.0850`.

[79] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

[80] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2048–2057, 2015. URL `http://jmlr.org/proceedings/papers/v37/xuc15.html`.

[81] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2204–2212, 2014. URL `http://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention`.

[82] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 577–585, 2015. URL `http://papers.nips.cc/paper/5847-attention-based-models-for-speech-recognition`.

[83] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099, 2015. URL `http://arxiv.org/abs/1506.03099`.

[84] J. Kominek and A. W. Black. The CMU arctic speech databases. In *Fifth ISCA ITRW on Speech Synthesis, Pittsburgh, PA, USA, June 14-16, 2004*, pages 223–224, 2004. URL `http://www.isca-speech.org/archive_open/ssw5/ssw5_223.html`.

[85] F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[86] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

[87] A. Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL `http://arxiv.org/abs/1308.0850`.

[88] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499v2, 2016. URL `https://arxiv.org/abs/1609.03499v2`.

[89] L. Sun, K. Li, H. Wang, S. Kang, and H. M. Meng. Phonetic posteriorgrams for many-to-one voice conversion without parallel data training. In *IEEE International Conference on Multimedia and Expo, ICME 2016, Seattle, WA, USA, July 11-15, 2016*, pages 1–6, 2016. URL `http://dx.doi.org/10.1109/ICME.2016.7552917`.

[90] B. van Merriënboer, D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, and Y. Bengio. Blocks and fuel: Frameworks for deep learning. *CoRR*, abs/1506.00619, 2015. URL `http://arxiv.org/abs/1506.00619`.

Appendix A

L2F Deep Learning: A Deep Learning Framework

A.1 Framework layout

In order to address the deep learning stage of this thesis, several frameworks were considered, for example, Keras [85] and Blocks [90], among others. However, the flexibility we encountered in these frameworks (at least at the time of this thesis realization) did not met our requirements. Therefore, we developed a small framework for personal usage, although it is currently being developed into a cleaner version for general usage.

The framework was built on top of the well known mathematical framework Theano [86], this way supporting all the sound mathematical operations required by the models and layers developed. Developing right on top of Theano allows a greater flexibility versus the other frameworks considered (some are also built on top of this mathematical framework), although this also implied that some features already on those frameworks had to be implemented. Some of these features, such as, optimizers, were borrowed and put in our project with the reference and credit from where they were developed.

The proposed framework is structured into four main modules: `layers`, `models`, `optimization`, and `utils`. The `layers` module contains the layer class template, which we call `GenericLayer`, and implemented layers that are built in into the framework and that should be used through a model. The `models` module contains the implemented models we developed for this framework, and which can be trained by the user with a few lines of code. The `optimization` module content includes everything that is related to model optimization, such as optimizer algorithms (*e.g.* ADAM, RMSProp, SGD), batch update training, and regularization techniques like dropout. Finally, the `utils` module deals with everything else that is not included in the remaining modules, such as decay functions for a curriculum learning train, data handlers templates and Theano variables initializations. In the remaining sections of this appendix, the essential components of our framework will be described in their corresponding section.

A.1.1 Layers

A layer is an object representation of a possible layer in a deep neural network. Although there is a considerable amount of possible layers, we provide a few built in ones required for RNNs, sequence to sequence and attention models. In addition, the user is able to implement its own layer. The implemented layers are:

- Attention mechanism
- Fully Connected Layer (Dense)
- Time Distributed Fully Connected Layer (Time Distributed Dense)
- GRU
- Bidirectional GRU
- LSTM
- Bidirectional LSTM
- GRU Decoder
- LSTM Decoder
- GRU Attention Decoder

In the `layers` module, a class called `GenericLayer` defines the template for every layer included in the module, as well as futures layers that can be added to the framework by the developer, or by the user. The `GenericLayer` class works as a super class for all the layers. This class only has a `params` attribute and methods for loading and saving parameters from or to a file.

An example of the code of one of the simplest layers implemented, the dense layer, is shown below.

```
import theano.tensor as T
import utils.shared as shared
from layers.model import GenericLayer

class Dense(GenericLayer):

    def __init__(self, output_size, input_size, activation='linear', bias=True, name='dense'):
        self.input_size = input_size
        self.output_size = output_size
        self.activation = activation
        self.name = name
        self.bias = bias
        self.W = shared.random((self.input_size, self.output_size), name='{}_W'.format(name))
        param = [self.W]
        if self.bias:
            self.b = shared.zeros((self.output_size,), name='{}_b'.format(name))
            param += [self.b]
        super(Dense, self).__init__(param)

    def forward(self, x):
        z = T.dot(x, self.W) + self.b
        if self.activation == 'linear':
            return z
        elif self.activation == 'tanh':
            return T.tanh(z)
        elif self.activation == 'softmax':
            return T.nnet.softmax(z)
        elif self.activation == 'relu':
            return T.nnet.relu(z)
        else:
            raise Exception("Invalid Activation Function")

    def predict(self, x):
        return self.forward(x)
```

As observed in this example, the layer has three essential and mandatory components: a constructor, a `forward` method and a `predict` method.

The constructor's goal is to set and initialize all the layer's parameters. In the example depicted, the a matrix weight W and a bias vector b are initialized according to the user's options, which are also saved in the class's arguments. A list of all the layer's parameters are then passed to the `GenericLayer` constructor, which finalizes the initialization.

The `forward` method should define layer's behaviour during training. In the this example, the simple operation $z = \sigma(Wx + b)$ is applied, with W and b being the model's parameters and σ a non-linearity function. The general layer of our framework should receive an input with 3 dimensions of shape `(n_samples, n_timesteps, n_features)` as is the case of a time series sequence. However, some layers, as the dense, don't admit a time series and expect an input of shape `(n_samples, n_features)`. In order to solve this problem, we first reshape the input to obtain the expected shape. If the user desires to apply a dense layer to each time step of a sequence, the time distributed dense layer should be used instead.

The `predict` method should be present in each layer to distinguish the model's behaviour on prediction versus its behaviour during training, as some layer may behave differently. In the example presented the behaviour is identical and therefore, the method reproduces the `forward` method behaviour.

In the current version of the framework, a layer is not designed to be able to be trained individually and the user should define a model instead. This is expected to be changed in a future update.

A.1.2 Models

A model is an abstract representation for the desired behaviour using the layers defined. This is, the model allows, for example, stacking multiple layers to achieve a deep neural network, where the hidden state h'_{t-1} is the input for the next layer. The models developed are the following:

- Stacked Dense Layers (Simple DNN)
- Deep Bidirectional LSTM
- Deep Bidirectional GRU
- Sequence to Sequence
- Attention Sequence to Sequence

The model class scope is to combine several implemented layers as the user might intend to. Each model must be a subclass of the `GenericLayer` class and implement a `forward` and `predict` methods, similarly to what was done with the layer class. Implementing a model is an extremely easy process because it only involves calling the already defined methods of the layers one wants to use. An example of the definition of a model is shown below:

```python
from layers.dense import Dense
from layers.model import GenericLayer
from optimization.dropout import dropout

class StackedDense(GenericLayer):

    def __init__(self, input_size, h_sizes, output_size):
        self.input_size = input_size
        self.h_sizes = h_sizes
        self.output_size = output_size
        self.d1 = Dense(output_size=h_sizes[0], input_size=input_size, activation='relu', name='d1')
        self.d2 = Dense(output_size=h_sizes[1], input_size=h_sizes[0], activation='relu', name='d2')
        self.d3 = Dense(output_size=output_size, input_size=h_sizes[1], activation='softmax', name='d3')
        param = self.d1.param + self.d2.param + self.d3.param
        super(StackedDense, self).__init__(param)

    def forward(self, x, train_phase=True):
        # Forward Pass
        h = self.d1.forward(x)
        if train_phase:
            h = dropout(h, 0.1)
        h2 = self.d2.forward(h)
        if train_phase:
            h2 = dropout(h2, 0.1)
        y_hat = self.d3.forward(h2)
        return y_hat

    def predict(self, x):
        return self.forward(x, train_phase=False)
```

Each model requires to be compiled, this is, to build the computational graph with Theano variables and the parameters of each model in order to be able to apply the loss function to the predicted output and the true output, retrieving the gradient of the parameters, and applying the optimizer (*e.g.* Stochastic

Gradient Descent, Adam, among others) to the parameters. Then with these steps, the train and validation functions can be built using a *Theano function*, which will allow us to attribute numeric values to the computational graph built. In this compilation method, we define how we evaluate the model and allow some useful techniques to be applied, such as masking, masked cost, number of output time steps, as well as allowing the decoders access the true output (if the model handles different length sequences). A batch update function is responsible to apply these functions to the training and validation data, in order to train the model.

Our implementations of batch update support both regular sequential models, as well as sequence to sequence models, with an option to use a scheduled sampling approach. The batch update algorithm iteratively feeds mini-batches of samples from a training set to the training function, until the end of an epoch. At the end of every epoch, the validation function is computed on a validation set to evaluate how well the model performs outside the training set.

An early stopping with patience mechanism is implemented within the batch update algorithm, based on the loss of the validation set. The developed mechanism implemented sets a maximum patience value (in epochs) at the beginning of training that decreases when the validation loss of an epoch is worse than the best model and worse than the previous epoch, otherwise it stays unaltered. As soon as another best model is found, the maximum patience value is reestablished. The training stops when the patience value reaches zero, or the maximum number of epoch is reached. As soon as a new best model is found, the model's parameters are saved into a file by serializing a list of numpy matrices that correspond to all of the network parameters. Within the framework, it is not yet possible to save the model architecture. Therefore, the user must know, with resort to proper file naming, the architecture corresponding to the saved weights file.

A.1.3 Usage example

Using a model with our framework is extremely simple and requires very little code. An example of a single GRU layer to learn a sinusoidal signals is presented below:

```
# Local
from bench import load_sinusoidal_data, prediction, plot_history, plot_predictions
from models.dbgru import DBGRU
from optimization.batch_update import batch_update

if __name__ == '__main__':

    # MODEL HYPERPERAMETERS
    hid_size = 7
    # CONTINOUS-CONTINOUS DATA
    input_size = 1
    output_size = 1
    # LOAD DATA
    data = load_sinusoidal_data()
    # TRAIN / VALIDATION / TEST
    train_data, val_data, test_data = data
    # MODEL DEFINITION
    model = DBGRU(input_size=input_size, h_sizes=[hid_size], output_size=output_size)
    # BATCH UPDATE
    print "Training"
    history = batch_update(model=model,
                           train_data=train_data,
                           validation_data=val_data,
                           n_epochs=400,
                           output_name='sin',
                           shuffle=True,
                           mask=False)
    # PLOT RESULTS
    plot_history(history)
    pred = prediction(model)
    plot_predictions(pred, data)
```

Using a sequence to sequence model with scheduled sampling is also very straightforward and requires only a few more parameters, with the use of `batch_update_info` instead of the regular `batch_update` function. An use case for mapping different size sinusoidal signals, using the Attention Sequence to Sequence model is shown below:

```
# Local
from bench import load_sinusoidal_data, prediction, plot_history, plot_predictions
from models.seq2seq import Seq2SeqAttention
from optimization.batch_update import batch_update_info
from utils.decay import *

if __name__ == '__main__':
    # MODEL HYPERPERAMETERS
    hid_size = 7
    att_size = 5
    # CONTINOUS-CONTINOUS DATA
    input_size = 1
    output_size = 1
    # LOAD DATA
    data = load_sinusoidal_data()
    # TRAIN / VALIDATION / TEST
    train_data, val_data, test_data = data
    # MODEL DEFINITION
    model = Seq2SeqAttention(input_size=input_size, h_sizes=[hid_size, hid_size],
                             att_size=att_size, output_size=output_size)
    # BATCH UPDATE (Exponential Decay, k = 0.9, save model below epsilon = 0.2)
    print "Training"
    history = batch_update_info(model=model,
                                train_data=train_data,
                                validation_data=val_data,
                                n_epochs=50,
                                output_name='sin',
                                shuffle=True,
                                        mask=False,
                                        curriculum_learning=exponential,
                                        k = 0.9, epsilon_threshold=0.2)
    # PLOT RESULTS
    plot_history(history)
    pred = prediction(model, info=True)
    plot_predictions(pred, data, info=True)
```

A.1.4 Future Development

A cleaner and more flexible version is currently being developed. This version will support even more features, such as beam search, embeddings layers, convolution layers, among others. Although there are very powerful frameworks already being developed, this is yet another one with the advantage of modelling the layers as python classes, providing intelligible code, flexible enough to add other layers easily as well as models. All of these advantages right on top of Theano.