Toward universal speech synthesis : harnessing linguistic and stylistic embeddings for expertise-free and flexible systems

Antoine Perquin

DOCTORAT BRETAGNE LOIRE / MATHSTIC

INSA RENNES

THESE DE DOCTORAT DE

L'INSTITUT NATIONAL DES SCIENCES APPLIQUEES RENNES

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Informatique*

Par

« Antoine PERQUIN »

« Toward universal speech synthesis : harnessing linguistic and stylistic embeddings for expertise-free and flexible systems »

Thèse présentée et soutenue à « Lannion », le « 12 février 2021 »
Unité de recherche : UMR 6074 IRISA
Thèse N° : 21ISAR 05 / D21 - 05

Rapporteurs avant soutenance :

Georges Linarès	Professeur des universités – Université d'Avignon / LIA
Philip N. Garner	Senior researcher – Idiap Research Institute (Suisse)

Composition du Jury :

Président :	Pascale Sébillot	Professeure des universités – INSA Rennes / IRISA
Rapporteurs :	Georges Linarès	Professeur des universités – Université d'Avignon / LIA
	Philip N. Garner	Senior researcher – Idiap Research Institute (Suisse)
Examinateurs :	Géraldine Damnati	Chercheuse – Orange Labs
	Camille Guinaudeau	Maîtresse de conférences – Université Paris-Sud / LIMSI
Dir. de thèse :	Laurent Amsaleg	Directeur de recherche – CNRS / IRISA

Invités

Gwénolé Lecorvé	Maître de conférences, HDR, Université de Rennes 1 / IRISA
Damien Lolive	Maître de conférences, HDR, Université de Rennes 1 / IRISA
Junichi Yamagishi	Professor, National Institute of Informatics (Japon)
Marie Tahon	Maîtresse de conférences, Université du Mans / LIUM

Résumé en Français

La parole est la capacité d'un être à exprimer ses pensées et émotions par le biais de sons. En tant que telle, elle est souvent considérée comme un propre de l'homme. De nombreux travaux ont été conduits afin de comprendre comment l'être humain est capable de s'exprimer au travers de la parole, mais aussi afin de reproduire cette capacité artificiellement. Ces travaux ont donné naissance à la technologie qu'est la synthèse de parole.

Cette thèse se concentre sur la synthèse de parole à partir de texte. Il s'agit d'un processus qui permet de générer un signal de parole correspondant à la lecture d'un texte. Ainsi, ce procédé implique deux modalités du langage : texte et audio. Il est souvent décrit comme étant deux problèmes indépendants. Le premier est la prédiction de la prononciation d'un texte, ce qui nécessite une expertise linguistique. Le second est la génération d'un signal de parole correspondant à cette prononciation, ce qui nécessite une expertise en traitement du signal.

Cette technologie a de nombreux usages. Dans le domaine médical, elle peut être utilisée afin de permettre aux personnes atteintes d'aphasie de communiquer à l'oral. Elle peut aussi être utilisée pour guider la navigation de personnes malvoyantes sur le web, en décrivant oralement les contenus visuels et textuels. Dans le domaine du divertissement, la synthèse de parole pourrait être utilisée en tant que doubleur artificiel pour le doublage de films ou l'enregistrement de livres audios. L'interaction homme-machine est une application qui gagnera probablement en popularité dans les années à venir. En particulier, les utilisateurs d'assistants virtuels adressent leurs requêtes oralement. Celles-ci sont reconnues grâce à la reconnaissance de parole automatique. Une fois la requête traitée par l'assistant, il fournit sa réponse à l'utilisateur sous la forme d'un signal de parole synthétisé. La synthèse de parole a le potentiel d'être utilisée universellement, une fois que ses limitations auront été adressées.

En effet, les systèmes de synthèse vocaux actuels ne peuvent être décrits comme "universel". Selon la définition, une chose est universelle si elle est "applicable partout, dans tous les cas" et si elle est "utilisée ou comprise par tous". Les applications des systèmes actuels sont limitées par le manque de personnalisation possible. La majorité

des systèmes est limitée à produire de la parole correspondant à une unique voix. Afin de synthétiser de la parole avec d'autres voix, la partie d'un système qui génère le signal audio doit être reconstruite. De manière similaire, afin de synthétiser des textes écrits dans une autre langue, la partie qui prédit la prononciation d'un texte doit être reconstruite. Afin de reconstruire ces deux parties, l'utilisateur doit être un expert en synthèse de parole, ce qui va à l'encontre du deuxième sens du mot "universel". Pour qu'un système de synthèse vocal soit réellement universel, il doit être capable de synthétiser de nombreuses formes de parole (plusieurs locuteurs, langues, émotions, etc.). De plus, ces formes doivent être contrôlable par des utilisateurs non experts.

Ce manuscrit ne prétend pas construire le système de synthèse universel décrit précédemment. Cependant, il explore une méthode permettant d'éliminer l'expertise linguistique nécessaire à la construction d'un système de synthèse vocal. Il explore aussi différentes manières de contrôler les formes de parole synthétisable par un système. Plus précisément, ce manuscrit étudie comment la propriété de plongement (*embbedding* en anglais, terme que nous utiliserons dans ce résumé) des réseaux de neurones permet de diminuer la quantité d'expertise linguistique, et permet de modéliser explicitement différentes formes de paroles, nous rapprochant ainsi d'un système de synthèse de parole universel.

Chapitre 1 : État de l'art

La synthèse vocale transforme un texte en un signal de parole correspondant à la lecture de ce texte. Elle est souvent décomposée en deux sous-problèmes indépendants. Le premier est l'analyse linguistique, qui prédit entre autre la prononciation du texte. Le second est la génération d'un signal de parole à partir de descripteurs linguistiques résultant de l'analyse linguistique.

La prédiction de la prononciation est effectuée à l'aide d'un module appelé phonétiseur. La prononciation d'un mot est prédite grâce à un dictionnaire de prononciation et, si le mot n'est pas inclus dans le dictionnaire, des règles de prononciation. Cette étape nécessite une forte expertise linguistique. La génération du signal peut-être effectuée de différentes manières, notamment par sélection d'unités.

Les méthodes de synthèse par sélection d'unités reposent sur la présence d'une base de données contenant des échantillons de paroles préenregistrés à l'échelle du phonème. Le signal de parole est généré en concaténant des unités sélectionnées au sein de la base de parole. La séquence d'unités à concaténer est choisie afin de minimiser la somme de

deux coûts. Le coût de sélection représente à quel point une unité dans la base de parole correspond à une unité du texte à synthétiser. Ce coût est défini sur des descripteurs linguistiques et nécessite donc une expertise linguistique. Le coût de concaténation estime la qualité de la concaténation entre deux unités de la base de parole. Ce coût est défini sur les caractéristiques acoustiques des unités.

Un nouveau paradigme de synthèse de parole a récemment été introduit : les systèmes bout-en-bout. Leur but est de remplacer la totalité de la chaîne de traitement en synthèse de parole par un unique réseau de neurones. En réalité, la solution adoptée correspond à utiliser deux réseaux de neurones. Le premier effectue la prédiction de caractéristiques acoustiques à partir d'un texte ou de la séquence de phonèmes correspondante. Le second convertit les caractéristiques acoustiques en un signal audio. Ces systèmes permettent actuellement d'obtenir la meilleure qualité audio possible. Cependant, l'utilisation de réseaux de neurones conduit à un manque d'interprétabilité.

Chapitre 2 : Utilisation d'*embeddings* de phones pour réduire l'expertise linguistique

Nous proposons d'utiliser des *embeddings* de phones afin de diminuer l'expertise linguistique nécessaire pour la synthèse de parole par sélection d'unités. Le coût de sélection est définit entre les descriptions linguistiques d'un phone présent dans la base de données, et un phonème dans la séquence à synthétiser. Ce coût agit en tant que mesure de similarité entre deux phones. Deux phones peuvent être similaires car prononcés dans des contextes linguistiques proches, ou car leur réalisation acoustique est similaire. Le coût de sélection traditionnel ne couvre que le premier type de similarité. Nous proposons d'apprendre un réseau de neurones en tant que modèle acoustique afin d'extraire des *embeddings* de phones. Ainsi, les *embeddings* extraits à partir de ce réseau encodent la description linguistique du phone, et ces *embeddings* peuvent aussi capturer des caractéristiques acoustique puisque le modèle est entraîné en tant que modèle acoustique. Alors, le coût de sélection peut être définit comme la distance euclidienne dans l'espace d'*embeddings* de phone. Ce coût n'est plus définit manuellement sur la description linguistique d'un phone, ce qui diminue l'expertise linguistique.

Nous comparons un système de synthèse par sélection d'unités où le coût de sélection est défini de manière experte, avec un système où il est défini selon la méthode proposée. Nous réalisons un test d'écoute comparant les échantillons synthétisés par chacun des

systèmes. À la suite de ce test, nous pouvons conclure que le système proposé est de qualité équivalent ou meilleure à celle du système expert. De plus, cette observation reste vraie quand le modèle est utilisé pour synthétiser des textes d'un domaine différent de celui des données utilisées pour entraîner les *embeddings* de phones. Ainsi, l'expertise linguistique d'une méthode par sélection d'unités peut être diminuée grâce aux *embeddings* de phones, sans diminuer la qualité de la synthèse.

Nous proposons aussi d'explorer visuellement l'espace d'*embeddings* de phones. La visualisation est effectuée par une Analyse en Composantes Principales (ACP), et suggère que la distribution des *embeddings* de phones est effectuée par phonème, et que ces groupes par phonèmes sont distribués selon une similarité acoustique. Nous mesurons objectivement la qualité de l'espace d'*embeddings* de phones par deux mesures. La première correspond à la précision d'un modèle acoustique linéaire appris sur les *embeddings* de phones. La seconde mesure la précision d'une classification par plus proches voisins dans l'espace d'*embeddings*. Ces mesures objectives confirment que l'espace d'*embeddings* encode les caractéristiques phonétiques d'un phone, et capture certaines caractéristiques acoustiques.

Malgré l'utilisation d'*embeddings* de phone, une forme d'expertise linguistique reste nécessaire pour la prédiction de la séquence de phonèmes correspondant au texte, et pour la définition de la description linguistique d'un phone utilisée pour entraîner le modèle acoustique neuronal.

Chapitre 3 : Utilisation d'*embeddings* de caractères pour supprimer l'expertise linguistique

Afin de diminuer plus encore l'expertise linguistique nécessaire à la construction d'un système de synthèse vocale, nous nous intéressons aux méthodes bout-en-bout. Les modèles neuronaux suivant l'architecture Tacotron permettent de générer un mel-spectrogramme à partir d'une séquence de phonèmes correspondant à un texte. Cette méthode diminue la quantité d'expertise linguistique car elle ne nécessite pas la définition et l'extraction d'une description linguistique pour chaque phone. L'expertise linguistique peut même être supprimée totalement en entraînant le modèle Tacotron sur les caractères du texte directement. À l'aide d'un test d'écoute, nous montrons qu'un modèle Tacotron entraîné sur des caractères offre des performances similaires en termes de qualité à un modèle entraîné sur des phonèmes. À l'aide d'un test d'écoute et de mesures objectives, nous

montrons aussi que le modèle entraîné sur des caractères ne commet pas plus d'erreurs de prononciation que le modèle entraîné sur des phonèmes.

Nous proposons ensuite d'analyser l'espace d'*embedding* de caractères. Une visualisation de l'espace suggère que les *embeddings* de caractères sont regroupés en fonction du phonème qu'ils participent à former. Nous comparons ensuite un phonétiseur appris sur des caractères ou sur des *embeddings* de caractères. Il en résulte que le modèle appris sur les *embeddings* de caractères est plus précis. Cela suggère qu'un modèle Tacotron capture automatiquement des caractéristiques phonétiques sans aucune information explicite.

Chapitre 4 : Utilisation d'*embeddings* de locuteurs pour la synthèse multi-locuteurs

Un système de synthèse universel doit être capable de produire différents styles de parole : plusieurs locuteurs, émotions, etc. Nous nous concentrons dans un premier temps sur l'implantation d'un unique style de parole : la voix d'un locuteur. Nous proposons de conditionner un modèle Tacotron sur des *embeddings* de locuteurs. Cette modélisation explicite du locuteur doit permettre au système de capturer des caractéristiques qui lui sont propres, comme sa voix, afin de les reproduire artificiellement. Nous étudions plusieurs méthodes d'augmentation de données dans le but de stabiliser le module d'alignement d'un modèle Tacotron multi-locuteurs. Nous montrons que toute donnée textuelle, y compris celle ayant peu d'intérêt pour la synthèse de parole, peut aider à stabiliser l'alignement. Nous évaluons ensuite objectivement la capacité du système à reproduire la voix de locuteurs n'appartenant pas au jeu d'apprentissage. Simplement fournir l'*embedding* d'un nouveau locuteur ne suffit pas à capturer fidèlement sa voix, le modèle doit être corrigé par *fine-tuning*. Cependant, une petite quantité de données suffit à effectuer cette correction et permet de rivaliser avec un modèle entraîné pour synthétiser spécifiquement la voix du locuteur.

Chapitre 5 : Utilisation d'*embeddings* d'accents pour la synthèse multi-locuteurs multi-accents

Nous proposons ensuite d'étudier le cas d'un système de synthèse capable de contrôler deux types de parole. En particulier, nous tentons d'implanter un système de synthèse

multi-locuteur multi-accent. De manière similaire au système multi-locuteur simple, nous proposons de conditionner un modèle Tacotron sur des *embeddings* de locuteurs et des *embeddings* d'accents. Afin de contrôler ces deux aspects de la parole indépendamment, nous proposons d'apprendre les encodeurs de locuteurs et d'accents de manière adversarielle. Cependant, les mesures objectives effectuées ne permettent pas d'affirmer que nous ayons réussis à modéliser ces composantes indépendamment. Nous évaluons ensuite objectivement la capacité du modèle à reproduire la voix et l'accent d'un locuteur. Pour cela, nous mesurons la similarité cosinus entre des *embeddings* extraits de parole naturelle avec ceux extraits de parole synthétisé. Nous montrons qu'objectivement, le modèle est capable de reproduire fidèlement la voix d'un locuteur et son accent. Nous tentons ensuite se synthétiser de la parole avec la voix d'un locuteur et un accent différent de son accent d'origine. Des mesures objectives à l'aide de la similarité cosinus suggèrent que l'accent synthétisé est différent de celui censé être reproduit.

Au cours de cette thèse nous montrons que l'expertise linguistique d'un système de synthèse de parole peut être limitée, voir éliminée, grâce à l'utilisation d'*embeddings* de phones ou de caractères. De plus, cette propriété des réseaux de neurones peut aussi être utilisée pour modéliser explicitement différents styles de parole. Par exemple, un modèle Tacotron conditionné sur ces *embeddings* permet de synthétiser des échantillons reproduisant la voix et l'accent d'un locuteur choisi. Cependant, la question de la modélisation de ces styles de parole pour les contrôler indépendamment reste ouverte.

Table of Contents

Introduction

Speech is the ability to express one's thoughts and emotions by way of sounds. As such, it is often thought to be a trait peculiar to humankind. Many works have been made to understand how humans are able to express themselves through speech, but also how to replicate this ability artificially, giving birth to the technology of speech synthesis.

In this thesis, the focus is on Text-To-Speech (TTS) synthesis which is the process of generating a speech audio signal corresponding to a given text. Thus, this problem involves two modalities of language: text and audio. As such, the problem is often seen as two independent problems. The first one would be how to infer the reading of a text, which requires linguistic expertise. The second would be how to generate a speech signal corresponding to that reading, which requires audio processing expertise.

This technology can have many uses. Medically, it can be used to allow speech-impaired persons to communicate orally; for vision impaired persons, it can be used to orally describe visual and textual content found on the web. For entertainment purposes, speech synthesis could potentially be used as an artificial voice actor for movie dubbing or automatically generating audiobooks. A current use that will potentially become more prominent in the years to come is machine interaction. With the rise of virtual assistants, users address themselves to the assistant orally. The commands are recognized through the use of automatic speech recognition. Then, after the command is processed by the assistant, it provides an oral answer to the user's request orally thanks to speech synthesis. This technology has the potential to be universally used, once its issues are addressed.

Indeed, current speech systems can hardly be described as universal. According to the definition, for something to be universal, it needs to be "applicable everywhere or in all cases" and "used or understood by all". Current speech synthesis is limited in its range of application because of its lack of customization. Most systems are limited to output speech for a given voice. In order to synthesize speech for a different voice, the half of the system that generates the audio signal needs to be rebuilt. Similarly, to synthesize speech in another language, the other half of the system that deals with how to read the text must also be rebuilt. In order to rebuild those two parts, the user would need to be an expert in speech synthesis which goes against the second aspect of the definition of universality. For

a speech synthesis system to be truly universal, it needs to be able to synthesize a wide variety of speech (multiple speaker, language, emotions, etc.). Furthermore, the variety of speech must be easily controllable by non-expert users.

Aim of the Thesis

This thesis does not claim to build the universal system sketched previously. However, we first aim to lower the barrier of entry in building a TTS system by removing linguistic expertise. We also aim to explicitly model multiple components of speech in order to allow more variety in synthesized speech. For both goals, we investigate the use of *embeddings*, the representations extracted by the hidden layer of a Deep Neural Network (DNN).

The works presented in this manuscript began during the transition period where the standard for TTS went from either unit selection or acoustic modeling using DNNs to end-to-end approaches where the entire TTS pipeline is replaced by DNNs. As such, we begin by investigating how an hybrid approach mixing unit selection and acoustic modeling allows to lower linguistic expertise. We then study end-to-end systems for linguistic expertise removal. Finally, we attempt to extend the end-to-end paradigm to allow a single system to synthesize the voices of multiple speakers with different regional accents.

Outline

This manuscript is organized as follows:

— Chapter 1 begins by presenting the general concepts related to text-to-speech synthesis, as well as the different models available.

— Chapter 2 introduces concepts related to deep neural networks and how they are used in the context of speech synthesis.

— Chapter 3 investigates the use of embeddings in *unit-selection text-to-speech synthesis*. Many commercial systems follow this paradigm of synthesis where pre-recorded units of speech are concatenated to match the text to synthesize. The sequence of pre-recorded units are selected by a process involving two costs. The target cost measures the similarity between a pre-recorded unit and units in the text being synthesized. The join costs estimates the quality of the concatenation between two units. The definition of the target cost is usually done with linguistic expertise. In this work, we show that training a neural network as an acoustic model to

predict an acoustic description of the pre-recorded units can lead to extracting phone embeddings. Those embeddings can then be used to define the target cost automatically. Our experiences show that such a definition does not lower the quality of synthesized speech, while lowering the amount of linguistic expertise needed to build a speech synthesis system. However, linguistic expertise is still needed in order to deduce the pronunciation of the text and to describe the units.

— To further reduce the need for linguistic expertise, Chapter 4 investigates the end-to-end paradigm for speech synthesis. End-to-end models aim to use a single neural network to perform a mapping from text to audio. In reality, the problem is still cut in two: the prediction a mel-spectrogram from a text, then a conversion of the mel-spectrogram into audio. Different approaches advocate to train the mel-spectrogram prediction on the phonetization of the text rather the text itself, in order to avoid pronunciation errors. We show that for French, in the case of a well-curated dataset, both approaches perform equally well. Furthermore, a study of the embeddings derived from characters hints that the neural network learns an internal representation of text akin to phones. Thus, those networks can be trained directly on the raw text, removing the need for linguistic expertise to build a speech synthesis system.

— Chapter 5 investigates the property of embeddings to allow more variety in speech synthesized by end-to-end models. Neural networks can be trained to derive speaker embeddings from audio. Then, end-to-end models can be conditioned on those speaker embeddings to allow multi-speaker speech synthesis. Furthermore, those models can also copy the voice of speakers unseen during training with mixed success. The variety of speech is controlled simply, by giving audio of the desired voice as a reference to the model. We then investigate an extension of this method for multi-speaker and multi accent synthesis. In a manner similar to speaker embeddings, regional accent embeddings can be trained. Then, we investigate the possibility of transferring accents between speakers

— Finally, Chapter 6 concludes this manuscript by reviewing the solutions proposed and opening discussions on the remaining issues.

Part I

State of the Art

Chapter 1

Overview of Speech Synthesis

TTS synthesis is the process of generating a sound signal corresponding to the reading of a written text. Thus, an understanding of both the textual and audio components of speech are needed to build a TTS system. Section 1.1 presents how speech is produced and how both of its components can be represented. Section 1.2 introduces the general pipeline followed by most TTS systems. Finally, Section 1.3 presents an overview of the different methods available for speech synthesis.

1.1 Speech Description

Speech and text are essentially two very different phenomenon since they are tied to two different senses. While speech is tied to auditory perception, text is mainly tied to vision. However, speech can be transcribed as a text, and a text can be transcribed as speech. This is because speech and text are tied by linguistics. The rest of this section describes how speech is produced, how to transcribe it textually, and how to represent it.

1.1.1 Speech Production

Natural speech, the result of a human being speaking, is a sound signal produced by the interaction of many organs in the human body. The main organs responsible for the generation of speech are the "lungs, larynx, pharynx, nose and various parts of the mouth" (Holmes 2001) as shown on Figure **??**. Natural speech is the sound resulting from the air flow created by air expelled from the lungs and interacting with the various vocal organs. Different configurations of the vocal organs allow for various sounds to be generated, and thus to pronounce different phonemes, the smallest differential unit in the sound system of a language. In the larynx, vocal folds are two folds of tissue that can be moved by muscles. In particular, they can be brought together, almost touching and closing the larynx. When air forces its way out of the larynx with this configuration, the vocal folds

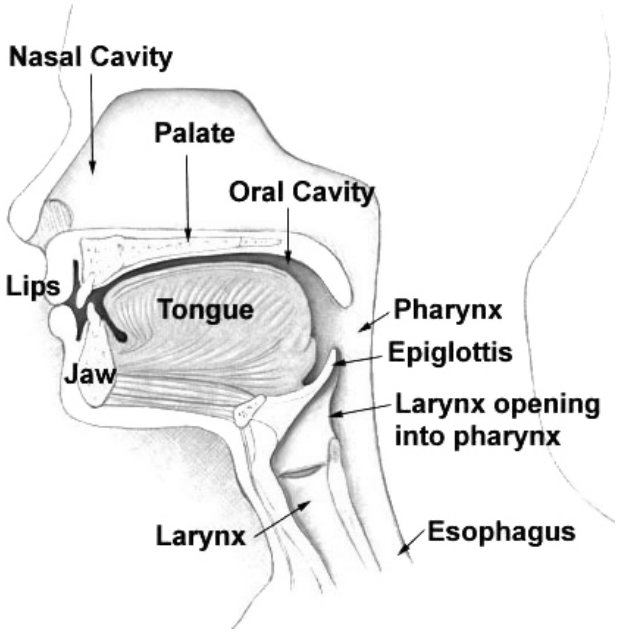


Figure 1.1 – Schema of the organs involved in speech production.

vibrate, modulating the air flow, rendering the wave quasi-periodical with a fundamental frequency F_0. This allows voiced sounds to be produced. Then, when passing through the vocal tract (cavity extending from the larynx to the pharynx, mouth and lips), a phenomenon of resonance takes place. The resulting resonant modes are called formants, and are usually noted as F_1, ..., F_i, where i is the order of the mode. Formants can be used to distinguish between phonemes. For unvoiced sounds, the vocal folds do not vibrate and thus do not possess a fundamental frequency. They are usually the consequence of the air flow being blocked then suddenly released by one or multiple vocal organs.

1.1.2 Written Speech

In linguistics, a phoneme is defined as the smallest differential unit in the sound system of a language. By combining these phonemes, a speaker can pronounce words that are combined in sentences. In comparison, the smallest differential unit in the writing system of a language is the grapheme. In the case of languages such as English or French, graphemes are letters of the alphabet. For languages like Chinese, graphemes are the ideograms. Similarly to phonemes in speech, the graphemes can be combined in words and then in sentences to record the same meaning. Thus by mapping graphemes to phonemes, one can deduce the pronunciation of a written text. However, this mapping is usually not

one-to-one. Often, two graphemes or more must be combined to form a single phoneme. A single grapheme can appear in widely different sounding phonemes. For example, in the French words "*entre*" and "*mine*" the letter "*n*" is part of the phoneme /ɑ̃/ and /n/ respectively.

The action of converting a sequence of graphemes into a sequence of phonemes is called Grapheme-to-Phoneme (G2P) conversion. In the remainder of this thesis, the conversion process will also be called phonetization while the G2P converter will also be called phonetizer. Different types of algorithms can perform this operation. For example, rule-based methods (Ainsworth 1973) rely on rules written by linguistic experts to predict the pronunciation of a word according to the graphemes it is composed of. The resulting sequence of phonemes can be written down again without any confusion using a phonetic alphabet such as the International Phonetic Alphabet (IPA).

1.1.3 Representation of Speech

As a particular type of sound, speech is a displacement of air. This physical process can be recorded by a microphone as a variation of amplitude over time called a *waveform*. An example of waveform is presented in Figure 1.2. In that case, the amplitude measured is the displacement of the diaphragm of the microphone.

As every signal, a speech signal can also be described in the frequency domain. According to the Fourier theory, every signal can be decomposed as the sum of sinusoidal signals. Each of them is described by three components : amplitude, frequency and phase. Inversely, the signal can be reconstructed from those three components by applying inverse Fourier transform. Since the human ear is not sensitive to phase variations (Taylor 2009), the phase is often discarded from speech representations. The distribution of the amplitude over frequencies is called *power spectrum*, the variation of amplitude over frequencies is called spectral envelope. An example of spectrum is presented in Figure 1.2.

The evolution of the spectrum over time can be observed using a spectrogram. The full signal is first cut into frames of speech by multiplying it with window functions. Then, the spectrum can be computed over each frame to obtain the distribution of amplitude over frequencies for each timestep. Graphically (see the second row of Figure 1.2), for each frequency in the Y axis, over time in the X axis, the color represents the amplitude : the darker the color, the higher the amplitude in a particular frequency at a certain point in time. On the picture, the lower frequencies are tight.

The frequency scale of a spectrogram does not relate well to the human perception of

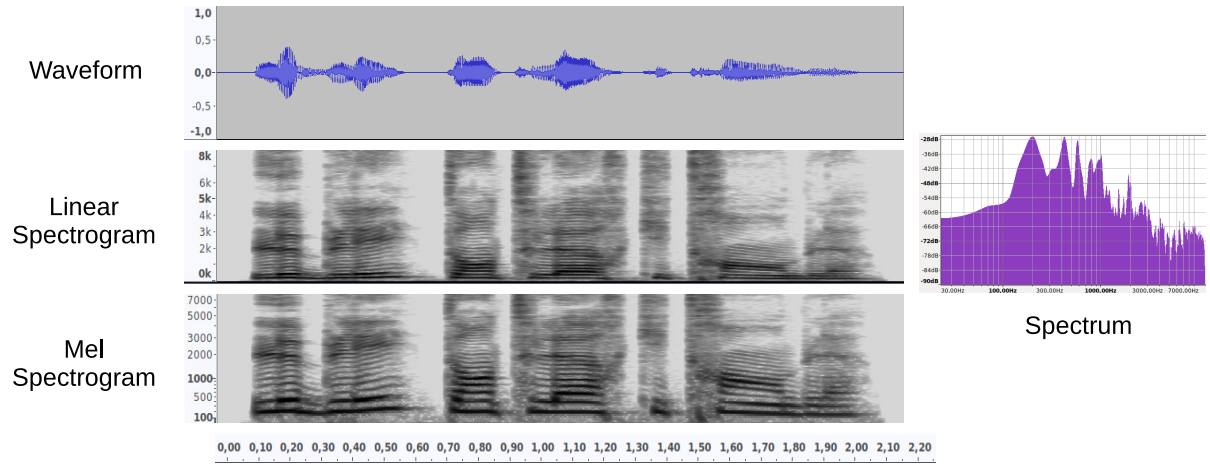


Figure 1.2 – Four different ways to represent speech.

speech. The human ear is able to distinguish between smaller difference of low-frequencies than for higher-frequencies. To take into account this aspect of speech perception, the frequency scale of spectrograms is often changed to the mel-scale. This is done by applying a non-linear transformation to the frequency scale. The result can be seen on the third row of Figure 1.2. As seen on the picture, the lower frequencies are much less packed than on the linear spectrogram.

Mel-spectrograms are often used as representations in speech synthesis. However, mel-spectrograms are not enough to reconstruct the original signal using inverse Fourier transform, since the phase information is missing. Then, the phase must be approximated using algorithms such as Griffin-Lim (Perraudin, Balazs, and Søndergaard 2013).

Finally, speech can be described by analyzing the spectral envelope of the signal through the use of Mel-Frequency Cepstral Coefficient (MFCC) or Mel-Cepstral Coefficient (MCC). MFCCs are computed after a Filter Bank Analysis (Davis and Mermelstein 1980). MCCs are computed from the cepstrum (Morise, Yokomori, and Ozawa 2016), which is defined as the Inverse Fourier Transform of the log-spectrum.

1.2 Text-to-Speech Framework

Historically, the text-to-speech framework can be divided as two sub-problems :

— How to predict the pronunciation of the text to synthesize?

— How to generate an audio signal corresponding to this reading?

Thus, text-to-speech synthesis can be done by following a two-level pipeline composed of

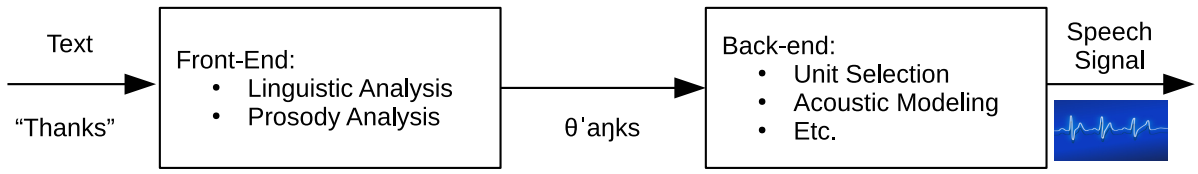


Figure 1.3 – Pipeline of the TTS process.

a front-end and a back-end, as drawn on Figure 1.3.

1.2.1 Front-end

The front-end of a TTS system takes a text as input and outputs a sequence of symbols representing the reading of that text. Usually, this sequence of symbols is a sequence of phonemes, the smallest unit in which speech can be segmented, represented using some phonetic alphabet.

This process is done using a phonetizer. One can infer the reading of a text by looking the entry of every word of that text in a pronunciation dictionary. A different approach would be to use pronunciation rules to infer the reading of the words. Actually, most phonetizers combine the two approaches by relying on pronunciation dictionaries and falling back on pronunciation rules for unknown words. Those pronunciation rules are also needed for languages where the pronunciation of words might depend on their context. It is the case of French were the ending of words might carry a "liaison" [1]. In any case, the pronunciation dictionary and rules need to be written by a linguistic expert, making the front-end language-dependent.

While the phonetization is the bare minimum output for a front-end, such systems usually extract further linguistic information from the text according to the needs of the back-end. Those information may include syllabication or statistics on different levels: position of a phone relative to the beginning of the sentence, inside the syllable, etc. The front-end might also extract prosodic information such as the potential position of pauses in the sentence, appropriate duration of words and phones, etc (Larreur, Emerard, and Marty 1989). One may also want to extract information related to speech styles. For example, whether the text corresponds to narration or dialogue, if the text should be read with a particular emotion, etc.

1. In "les enfants" the "s" in "les" is pronounced /z/, while it is muted in "les marchands".

1.2.2 Back-end

The back-end of a TTS system takes the linguistic information extracted by the front-end and generates an audio signal whose speech corresponds to the reading of a given text. This process can be done according to one of three paradigms : rule-based, concatenative, statistical parametric. For rule-based systems, the excitation parameters of a synthesizer are generated by hand-crafted rules. For concatenative systems, audio samples of prerecorded speech are arranged in a certain order and concatenated to produce speech corresponding to the text given. For Statistical Parametric Speech Synthesis (SPSS), a sequence of acoustic parameters are predicted by a statistical model. Then, those parameters can be transformed into a waveform thanks to a *vocoder*. Those paradigms will be presented in more details in Section 1.3.

1.3 Overview of Speech Synthesis Methods

1.3.1 Rule-based Systems

For rule-based systems, a synthesizer is used to produce speech from relevant parameters that vary from one type of synthesizer to another. One needs to be able to map from linguistic features to synthesizer parameters. In this approach, the mapping is done thanks to hand-crafted rules.

Formant-based synthesizers follow the Source/Filter idea : a source signal goes through a sequence of audio filters to generate a signal akin to speech. Thus, the synthesizer parameters are the fundamental frequency F_0 of the source signal and subsequent formants F_1 to F_3. Then, the problem becomes one of predicting the contour of those values from the linguistic description of a text. In this rule-based approach, each phoneme of the language is assigned a fundamental frequency and formant values. These values can be found by examining the spectrograms of recorded phonemes. The continuous contour used as synthesizer parameters are then generated by interpolating those values for following phones in the synthesizing sequence (Rabiner 1969).

Articulatory synthesizers aim to reproduce with fidelity how the human articulatory system actually works. To this end, the oldest articulatory synthesizer of von Kempelen was a mechanical machine mimicking the voice organs. Nowadays, articulatory synthesizers aim to physically model the behavior of the vocal tracts. The position of each vocal organs are measured for each phoneme thanks to Magnetic Resonance Imaging (MRI) scans of

the target speaker (Iskarous et al. 2003).

1.3.2 Concatenative and Unit Selection Systems

Concatenative speech synthesis methods were born from a simple idea to increase the naturalness of synthetic speech. By splitting pre-recorded audio samples into units of speech, storing them in a dictionary, then re-arranging and concatenating them in a different order, a new speech sample can be synthesized. The splitting of units usually occurs at the diphone level (Olive 1977) which are pairs of consecutive phones. But it can also be done at a higher level such as the word or syllable level. In any case, the dictionary contains a single recorded unit of every available type. For the diphone level, the synthesis occurs by simply picking the relevant diphones in the dictionary and concatenating them. In the case of words, rules must be written to fall back on diphones in cases where words to synthesize cannot be found in the units dictionary.

Increasing the dictionary size so that each target unit in the text can be matched to multiple recorded units with different prosody and acoustic realization helps to further improve the naturalness of concatenative synthetic speech. This actually ends up creating a *speech database*, rather than a dictionary, where units of different size can be mixed. In that case, the previous approach must be replaced with a way to select which recorded unit corresponds best to a given text unit.

For the unit selection methods (drawn on Figure 1.4), a database contains a large amount of pre-recorded speech segmented into units, usually diphones. The synthesis process is done by concatenating the sequence of database units that minimizes the sum of two costs (Hunt and A. W. Black 1996):

— The *target cost* T is computed for a given target unit u_t from the text and a given candidate unit s_t in the database. It represents whether a candidate unit is suitable to be a substitute for the target unit. This cost is usually computed as the weighted sum of sub-costs defined on the linguistic features of the units. For example, if the search for candidate units is limited to diphones with the same label as a the target unit, the target cost can be defined as the Manhattan distance between linguistic vectors containing only categorical attributes (stress, phrasing, etc.) (Taylor 2009)

— The *join cost* J is computed between two following units u_t and u_{t+1} in the selected sequence. It represents how well those units would concatenate. Usually, two diphones that follow each other in a word have a join cost of zero since their concatenation would be natural. It is usually defined a weighted sum of sub-costs

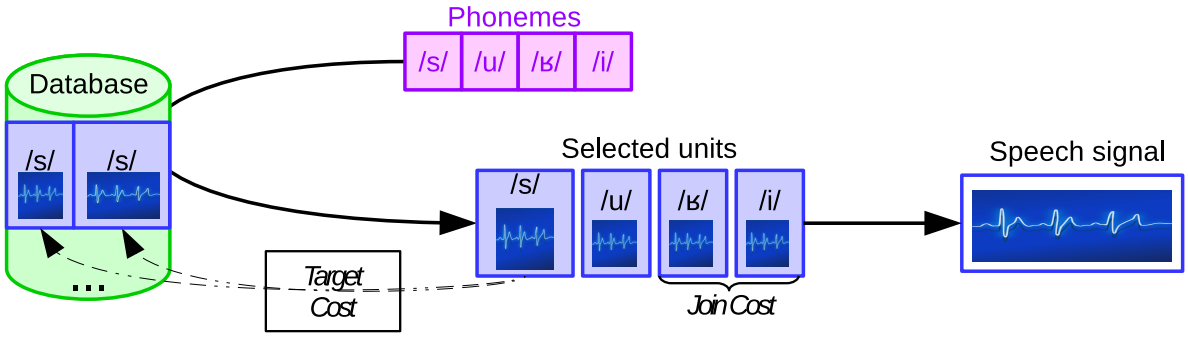


Figure 1.4 – Process to synthesize the sequence of phonemes /suʁi/ using unit selection.

defined on the acoustic characteristics of two units. For exemple, in (Hunt and A. W. Black 1996), three sub-costs are used : a "cepstral distance at the point of concatenation and the absolute differences in log power and pitch".

Then, the total cost C between a sequence of candidate units U and the sequence of target units S of length N is defined as (Taylor 2009) :

$$C(U,S) = \sum_{t=1}^{N} T(u_t, s_t) + \sum_{t=1}^{N-1} J(u_t, u_{t+1}) \tag{1.1}$$

The optimal sequence of units to select can then be found by placing all units in a lattice and exploring it by minimizing the total cost C. The resulting synthesized speech is the concatenation of those units.

1.3.3 Statistical Parametric Modeling

Similarly to formant-based synthesis, statistical parametric approaches use a vocoder to generate speech from a sequence of parameters. However, in the former approach, vocoder parameters are generated by hand-crafted rules, while in the latter approach, vocoder parameters are inferred from statistical models learned on pre-recorded speech.

The first statistical parametric systems made use of Gaussian Mixture Hidden Markov Models (HMM) (Yoshimura et al. 1999). In that case, speech is considered as acoustic observations corresponding to pronounced phonemes which can be modeled by Hidden Markov Models. At training time, the model is trained to predict the sequence of phonemes from the acoustic description of the signal. At synthesis time, the HMM is built by modeling each phoneme as a sequence of three states chosen among the pre-trained ones. Then, the acoustic description of the speech to synthesize is predicted by going over the

built HMM while maximizing the state sequence seen.

With the rise of DNN models in other fields, the HMMs started to be replaced by DNNs for acoustic modeling. In this case, a neural network is trained to predict vocoder parameters from a linguistic description of the text. The vocoder parameters are usually MCCs. They are predicted frame by frame from the linguistic features of the current phoneme and a timing information encoding the position of the frame under prediction. This means that these methods need a duration model to predict the length of each phone. The duration model may also be a neural network. The first models used to be simple feed-forward networks that achieved better-sounding synthesized speech than their HMM counter-parts (Ze, Senior, and Schuster 2013). Since speech is a temporal phenomenon, modeling time dependencies helps to further improve the performance of those models. These dependencies can be modeled by using Recurrent Neural Network (RNN)s layers such as the Long Short Term Memory (LSTM) layers. The concept of neural networks applied to acoustic modeling will be presented in further detail in Chapter 2.

1.3.4 End-to-end Systems

End-to-end systems aim to simplify the TTS pipeline by using a single neural network to directly map from the textual to the audio modality. First attempts toward end-to-end systems mimicked the traditional pipeline by replacing each part of it by a dedicated neural network (Arik et al. 2017). Current approaches simplify the pipeline further by having a single network predict vocoder parameters directly from text or its phonetization. Then, a (possibly neural) vocoder generates the speech audio from relevant predicted parameters. More precisely, the acoustic description of speech is predicted from text as a two-step process, following an encoder/decoder neural architecture. The text sentence is first encoded character per character. The process involves neural network layers such as LSTM in order to take the context of a character into account when encoding it. This allows the model to infer pronunciation. Then, vocoder parameters are predicted from the sequence of encoded characters. This involves a mechanism called *attention* that learns to align the encoded character sequence with the vocoder parameter sequence. The Char2Wav model (Sotelo et al. 2017) predicts vocoder parameters from characters, then use a SampleRNN (Mehri et al. 2016) neural vocoder to convert them into an speech signal. The Tacotron (Wang, Skerry-Ryan, et al. 2017) model predicts mel-spectrograms from characters and use Griffin-Lim to convert them into a speech signal. Both models also vary due to the architecture of their encoder and decoder.

End-to-end systems require a vocoder to convert the acoustic description predicted into a speech signal. Neural vocoders are a popular solution due to their ability to produce high quality speech. For example, WaveNet (Oord et al. 2016) is a probabilistic neural network able to generate audio. It can be conditioned on vocoder parameters to generate speech. It models waveforms as sequences of 16-bit integer values, quantized into 256 values for lower output dimensionality. The predictions are done frame by frame, conditioned on all the frames previously predicted. WaveRNN (Kalchbrenner et al. 2018) was proposed as a faster alternative.

The architecture of end-to-end systems and neural vocoders will be further presented in Chapter 2. End-to-end systems are used in Chapter 4 to remove most linguistic expertise from the TTS pipeline. They are then extended in Chapter 5 to perform multi-speaker multi-accent synthesis.

1.3.5 Hybrid systems

Hybrid systems refer to TTS systems that mix multiple paradigms of speech synthesis. In this section we focus on approaches that mix unit selection and statistical parametric modeling. More precisely, we review approaches where the target cost of a unit selection system is defined thanks to a statistical parametric model.

In (Merritt et al. 2016), a DNN is trained as an acoustic model to predict a frame of MCCs from the linguistic description $\boldsymbol{l}_{p_i}$ of a phone p_i and the timing information of a frame j inside p_i, just as in DNN-based SPSS. This work makes use of the embedding property of neural networks. Embeddings are latent representations defined by the output of one of a DNN's hidden layers. Let us note f the transformation function resulting from the application of every hidden layers up to and including the embedding extraction hidden layer. Then, in the case of an acoustic model, an embedding $\boldsymbol{e}_{p_i,j}$ is defined as:

$$\boldsymbol{e}_{p_i,j} = f(\boldsymbol{l}_{p_i}, t_j). \tag{1.2}$$

Since it is derived from t_j, this embedding is a representation of frame j of p_i rather than a representation of the phone as a whole. These embeddings are computed for every frame of a phone then pooled statistically on each fourth k of its length N by computing the mean $\boldsymbol{\mu}_k$ and standard deviation $\boldsymbol{\sigma}_k$. Then, the embeddings of each fourth are modeled

as a Gaussian mixture model $g_{p_i,k}$ following a normal distribution $\mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k^2)$:

$$\begin{aligned} \forall k \in [1,4], \boldsymbol{\mu}_{p_i,k} &= \frac{1}{N/4} \sum_{j=\frac{(k-1)\times N}{4}}^{\frac{k\times N}{4}} \boldsymbol{e}_{p_i,j}, \\ \boldsymbol{\sigma}_{p_i,k} &= \sqrt{\frac{1}{N/4} \sum_{j=\frac{(k-1)\times N}{4}}^{\frac{k\times N}{4}} (\boldsymbol{e}_{p_i,j} - \boldsymbol{\mu}_{p_i,k})^2}, \\ g_{p_i,k} &\sim \mathcal{N}(\boldsymbol{\mu}_{p_i,k}, \boldsymbol{\sigma}_{p_i,k}^2). \end{aligned} \tag{1.3}$$

Finally, the target cost between two units p_i and p_u is defined thanks to the Kullback-Leibler divergence D_{KL} (Hershey and Olsen 2007) between the random variables :

$$T(p_i, p_u) = \frac{\sum_{k=1}^{4} D_{KL}(g_{p_i,k} \| g_{p_u,k}) + D_{KL}(g_{p_u,k} \| g_{p_i,k})}{2}. \tag{1.4}$$

In (Wan et al. 2017), a DNN (drawn on Figure 1.5) is trained to predict MCC features from either linguistic features or the MCC features themselves. The linguistic and MCC features inputs are dealt with by separate encoders, before being projected into a same phone embedding space. While the linguistic features are already at the phone level, an LSTM layer has to be used to compress the MCC features sequence in a single vector representing a phone. From the embedding space, a decoder is used to predict MCC features. At training time, the phone embeddings are obtained from either linguistic or acoustic features, while at synthesis time the phone embeddings are always computed from linguistic features. Then, the Euclidean distance in the embedding space is used to define the target cost of a unit selection system. This paper works under the assumption that a good phone embedding for unit selection must reflect both the linguistic and acoustic feature spaces, which explain the use of a complex network where the phone embeddings are trained on both linguistic and acoustic features.

In Chapter 3, we build an hybrid unit selection system similar to (Wan et al. 2017) and challenge the claim that only good acoustic models lead to good phone embeddings for unit selection.

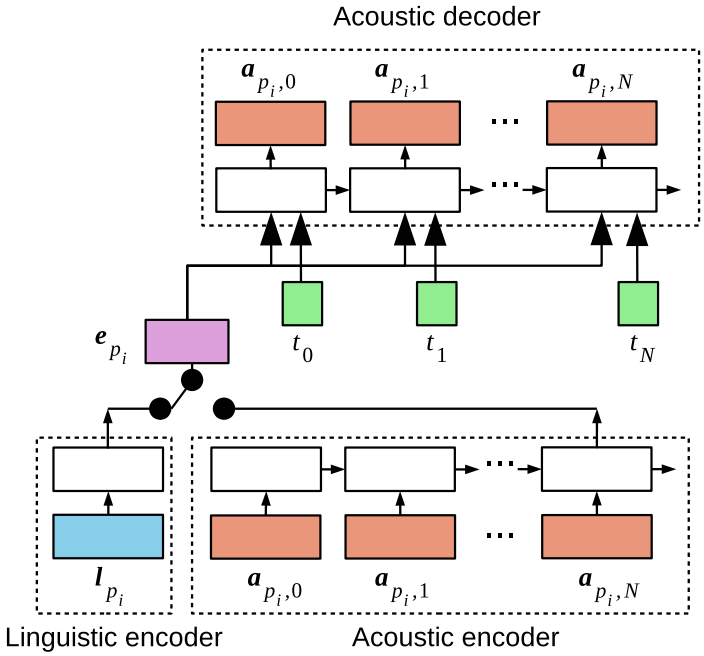


Figure 1.5 – Model of the architecture used in (Wan et al. 2017)

1.4 Evaluation for Speech Synthesis

Before presenting an overview of the different ways to evaluate synthesized speech, we need to consider what the quality of speech is. Usually, two aspects are considered. The intelligibility criterion is interested in whether the text can be understood from the synthesized samples. Naturalness represents how close to human speech the synthesized sample is. Both naturalness and intelligibility can be affected by a wide range of phenomena such as prosody and signal quality. Other criteria of evaluation include voice quality, pleasantness, etc. All of those criteria are often evaluated at once using the term "overall quality".

Since speech is perceptual in nature, the most reliable way to evaluate speech synthesis is to directly ask humans to listen to and rate synthesized samples. Such *listening tests* can be sorted among three categories according to the number of samples presented at once and the way to rate them. In order to score each system independently of each other, Mean Opinion Score (MOS) tests present a single sample from a non-specified system that must be rated between 1 and 5. To score each system relatively to each other, the Multi-Stimuli with Hidden Reference and Anchor points (MUSHRA) test presents a sample from each system in a random order on the same page. The samples must be rated between 0

and 100, relatively to each other. In MUSHRA tests, two additional systems are required: the natural samples and a low anchor system. Listeners are then instructed to rate at least one of the samples with a note of 100. While MUSHRA tests are interesting since they allow to compare systems directly, they are not practical when a lot of systems need to be compared. If one does not need a mean opinion score but only wants to rank the systems, one can use A/B tests. In that case, a sample from two systems is presented. The listeners must then choose the sample they prefer or vote for "no preference". Lastly, in the specific case of evaluating intelligibility, listeners can be asked to write a transcript of the spoken text. By comparing the transcripts to real text, one can compute Word Error Rate (WER).

However, listening tests are time-consuming to run and can be expensive if the listeners are not volunteers. In order to speed up development time and quality assessment, one can use objective measures as proxy of the quality of synthesized speech. Most often, objective measures for speech quality are extrinsic evaluation. Under the assumption that the synthesized speech quality is correlated to the underlying models quality, the evaluation of those models are used as an indication of the TTS quality.

In the case of SPSS, the prediction quality of the acoustic models can be measured thanks to the Mel-Cepstral Distortion (MCD) (Kominek, Schultz, and A. W. Black 2008). The MCD between two D-dimensional MCC sequences $\boldsymbol{x}$ and $\boldsymbol{y}$ is computed as an extension of the Euclidean norm by :

$$MCD(\boldsymbol{x}, \boldsymbol{y}) = \frac{\alpha}{T} \sum_{t=1}^{T} \sqrt{\sum_{d=1}^{D} (x_d(t) - y_d(t))^2} \text{ , with } \alpha = \frac{10\sqrt{2}}{ln(10)} \tag{1.5}$$

where T is the length of the shorter sequence if they are of different size, $x_d(t)$ and $y_d(t)$ are the d-th dimension at timestep t of vector $\boldsymbol{x}$ and $\boldsymbol{y}$ respectively. While the correlation between MCD and the overall quality of a TTS system is far from being perfect, large MCD improvement are usually perceptually audible.

1.5 Conclusion

Text-to-speech synthesis is a complex task that involves both textual and audio aspects of speech. Over the years, many methods resulting in varying quality of speech have been proposed by the community. In particular, the unit selection method and its hybrid variant using phone embeddings extracted from neural networks are of interest for Chapter 3. End-

to-end models, especially the Tacotron architecture, are of interests for Chapter 4 and Chapter 5. Chapter 4 will use them to attempt to remove all linguistic expertise from the TTS pipeline thanks to character embeddings. Then, in Chapter 5, they are conditioned on speaker and accent embeddings to model those two speech factors explicitly. Since neural networks are of importance in this thesis, the next chapter introduces concepts related to neural networks architecture as well as develops the architecture of the Tacotron model.

Chapter 2

Neural Networks for Speech Synthesis

While neural networks were introduced in the 1950s, they only picked up steam in the 2000s with the rise of computation power and of large datasets. Neural networks have re-emerged as a tool for machine learning with success in fields such as Computer Vision (Krizhevsky, Sutskever, and G. E. Hinton 2012). Thus, it is no surprise that neural networks have been used with success in the field of Text-to-Speech Synthesis. This chapter introduces the concepts and architectures used in this thesis. Section 2.1 presents the feed-forward architecture used by acoustic models, as well as extensions using more complex layers. The sequence-to-sequence paradigm is introduced in Section 2.2, before presenting the Tacotron architecture following this paradigm in Section 2.3. An example of neural vocoder, WaveRNN, is introduced in Section 2.4. Finally, Section 2.5 is an overview of the methods used to model different components of speech using neural networks.

2.1 Acoustic Models and Neural Networks

Neural networks are based on the perceptron model (Rosenblatt 1958). For a vector $\boldsymbol{x} = (x_1, ..., x_N) \in \mathbb{R}^N$, a single value $y \in \mathbb{R}$ is computed as :

$$y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b) = f(\sum_{i=1}^{N} w_i x_i + b) \tag{2.1}$$

where $\boldsymbol{w} = (w_1, ..., w_N) \in \mathbb{R}^N$ and $b \in \mathbb{R}$ are respectively the weights and bias of the neuron, f is the activation function and $\cdot$ is the dot product. The appropriate values for weights and bias can be learned by minimizing the error between prediction and reference values using optimizer algorithms such as the Stochastic Gradient Descent (SGD).

In order to predict $\boldsymbol{y}$, a M-dimensional vector rather than a single value, the model

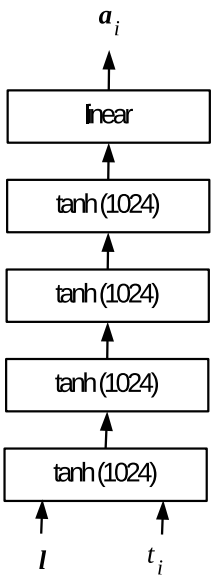


Figure 2.1 – Architecture of a feed-forward acoustic model.

can be extended with $\boldsymbol{W} \in \mathbb{R}^{N \times M}$ and $\boldsymbol{b} \in \mathbb{R}^N$:

$$\boldsymbol{y} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{2.2}$$

This process is deemed "layer" in a neural network. Indeed, since the activation function f is usually chosen to be non-linear, the process can be repeated with a new set of weights, bias and activation function and combined with the previous result. The process of combining these processes is similar to that of stacking layers. Such a network is called "feed-forward".

Feed-forward neural networks have been shown to give good results in statistic parametric speech synthesis. A sentence is cut into phonemes described using linguistic description vectors $\boldsymbol{l}$. The signal is cut into phones described using acoustic description feature vectors $\boldsymbol{a}$. Then, the neural networks predicts a frame by frame, from $\boldsymbol{l}$ and t_i the timing of the frame being predicted. The most common architecture stacks 4 or 5 feed-forward layers of size 1024 with *tanh* activation functions (Ze, Senior, and Schuster 2013) and is represented in Figure 2.1.

An interesting property of neural networks is their ability to learn latent representations. The output of each hidden layer can be seen as an alternative representation of the input data. Those alternative representations, called *embedding*, transforms a given vector into one of lower dimension containing only numerical features. Embeddings are

particularly useful when the input data is categorical. In that case, they first allow to considerably lower the dimensionality of the input. For example, in Natural Language Processing (NLP) words are usually one-hot encoded, resulting in each word being described with a vector the dimension of the vocabulary size. With embeddings, each word can be described with a smaller vector. Second, embeddings project any type of data into numerical feature spaces. This allows to compare data with intuitive measures such as the euclidean distance or the cosine similarity. Furthermore, those embeddings spaces have been shown to have interesting properties according to the training task, and may be used for secondary tasks. For example, in NLP, word embeddings spaces have been shown to keep semantic properties of the language (Mikolov et al. 2013). For example, the link between a capital and its corresponding country is captured by the embedding space. Indeed, the sum of the embedding of the words "Germany" and "capital" is close to the embedding of "Berlin". For speech, embeddings have been used to capture speaker characteristics (Snyder et al. 2018). Embeddings have also been successfully used for information retrieval (Gu et al. 2018). Overall, embeddings are compact representations useful for a wide variety of task such as comparing two elements. In Chapter 3, we will use this embedding property to define the target cost of a unit selection speech synthesis system. This is motivated by the fact that the target cost can be seen as a distance between textual units. As such, it can instead be defined as a distance between the embeddings of those units.

Since speech is a temporal phenomenon, modeling time dependencies can improve the accuracy of an acoustic model. Recurrent Neural Networks (RNN) attempt to address such a modeling. Recurrent layers work on sequence of vectors rather than a single one. The particularity of RNN is that computation for timestep t is done according to computations done at timestep $t-1$. This recurrence allows to model time-dependencies and to simulate memory by allowing information from previous timestep to be used for the computation of later timesteps. Example of RNN layers include the Long-Short Term Memory (LSTM) layer (Hochreiter and Schmidhuber 1997) and the Gated Recurrent Unit (GRU) layer (Cho et al. 2014). Pairs of RNN layers can also be combined to model time-dependencies both forward and backward in time. The first layer computes on the input sequence forward, the second computes backward and the result of both layers is combined by addition or concatenation. This process is called bi-directional RNN. Overall, the use of RNN to model the time-dependencies of speech has been shown to improve the precision of acoustic models for speech synthesis (Z. Wu, Watts, and King 2016).

Convolutional layers have originally been created to capture the spatial dependencies in images and have been applied successfully for image classification (Krizhevsky, Sutskever, and G. E. Hinton 2012). However, these layers can be adapted to work on sequential data such as time-series or text. In that case, a window slides over a given sequence of vectors to extract features. The size of the sliding window is called kernel width. The amount of timesteps the window slides over is its stride. The dimensionality of the representation computed is called the number of filters. The resulting features is computed by calculating the mathematical convolution of each filters with the windowed inputs. For a sequence $\boldsymbol{x}$ with N timesteps, a convolution layer with M filters $\boldsymbol{f_i} \in \mathbb{R}^S$ of width S , with a stride of 1, results in :

$$\boldsymbol{y_i}(t) = (\boldsymbol{f_i} * \boldsymbol{x})(t) = \sum_{j=t}^{t+S} f_i(j)x(j), \forall i \in [1, M] \tag{2.3}$$

where i is the index for the current filter, t is the timestep of the sliding window. 1-dimensional convolutional layers have been successfully used in a variety of task in NLP, such as sentiment analysis (Dos Santos and Gatti 2014). In speech synthesis, they are often used in end-to-end systems such as (Wang, Skerry-Ryan, et al. 2017). Convolutional layers allow to learn a contextualized representation, and by stacking those layers the receptive field of each layer grows with its depth, allowing to capture longer context. As such, convolutional layers can be used as an alternative to recurrent layers to work on sequential data.

2.2 Sequence-to-sequence Acoustic Modeling

The acoustic models presented in the previous section need to be trained with aligned linguistic and acoustic features. The alignment process can be a long and costly process if done manually. In this section, we present the architecture of neural networks able to predict sequences from unaligned data.

(Sutskever, Vinyals, and Le 2014) introduced the sequence-to-sequence architecture for neural translation. The aim is to translate a sentence from a source language to a target language. More generally, the model predicts a sequence $y = (\boldsymbol{y}_1, .., \boldsymbol{y}_M)$ from a sequence $x = (\boldsymbol{x}_1, .., \boldsymbol{x}_N)$. The model can be described as two sub-modules. First, the input sequence is encoded as a single summary vector : $\boldsymbol{e} = \mathit{encode}(x)$. Secondly, the decoder predicts the output sequence one timestep at a time. The prediction at timestep j, is done using the previous output $\boldsymbol{y}_{j-1}$ and a latent variable $\boldsymbol{h}_{j-1}$: $\boldsymbol{y}_j = \mathit{decode}(\boldsymbol{y}_{j-1}, \boldsymbol{h}_{j-1})$.

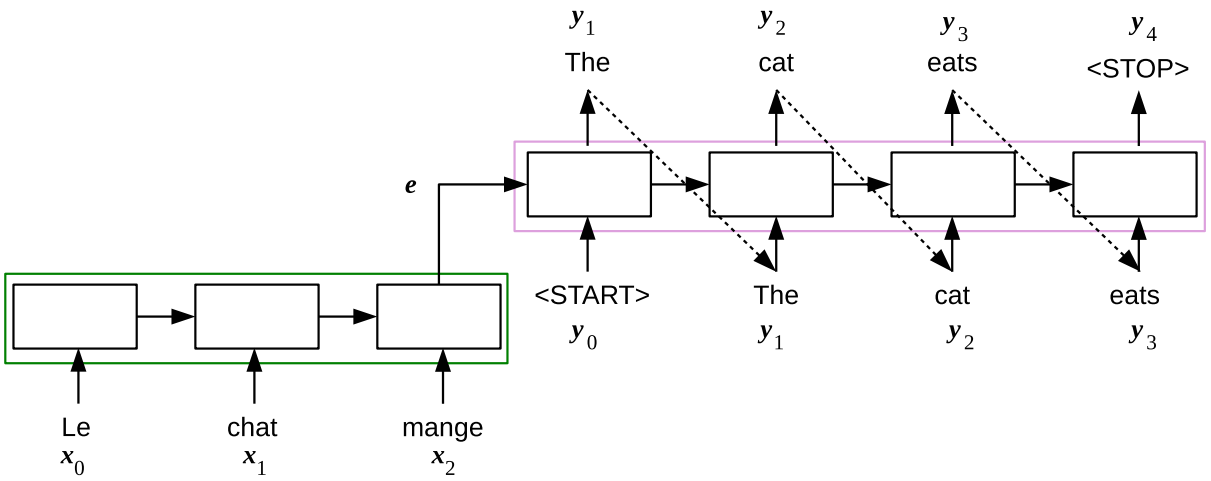


Figure 2.2 – Representation of the sequence-to-sequence architecture.

In the original architecture (drawn on Figure 2.2), the encoder is composed of a single LSTM layer. The outputs $\boldsymbol{e}_i$ of the encoder LSTM are discarded for each words but the last one. LSTM layers model time-dependencies so the output $\boldsymbol{e}_N$ for the last word can be seen as a summary of the whole sentence. Thus, $\boldsymbol{e} = \boldsymbol{e}_N$. The decoder is also made of a single LSTM layer. Its hidden state is initialized with the summary vector $\boldsymbol{e}$ allowing the decoder to be aware of the sentence to translate. Thus, $\boldsymbol{h}_0 = \boldsymbol{e}$. Then the prediction of the target sentence is done word by word. The first input to the decoder LSTM is a special token indicating the start of a new translation : $\boldsymbol{y}_0 =$ <START>. Then, for each following timestep, the input of the LSTM layer is the word predicted during the previous timestep. Words are predicted until the decoder predicts a special token indicating the end of a translation : $\boldsymbol{y}_M =$ <STOP>.

For speech synthesis, the sequence-to-sequence architecture is extended with an attention model to perform the summary. Attention is a mechanism where vectors of an input sequence $x = (\boldsymbol{x}_1, ..., \boldsymbol{x}_N)$ are aggregated relative to a timestep i of a sequence $y = (\boldsymbol{y}_1, ..., \boldsymbol{y}_M)$. This mechanism allows to determine the importance of each timestep of x relative to $\boldsymbol{y}_i$. More precisely, a summary vector of x is computed as a weighted sum :

$$\boldsymbol{c_i} = \sum_{j=1}^{N} \alpha_{ij} \boldsymbol{x_j} \tag{2.4}$$

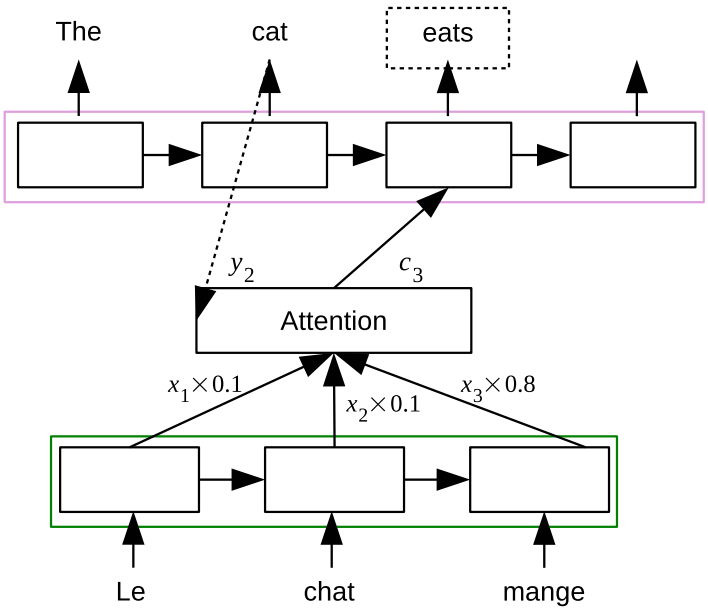


Figure 2.3 – Representation of the attention mechanism. Currently attending over the sequence (Le, chat, mange), relative to (The, cat). According to the attention, the most useful word to predict "eats" is "mange".

where $\boldsymbol{c_i}$ is the summary of x relative to y_i, and α_{ij} is the attention weight for the vector $\boldsymbol{x_j}$ relative to $\boldsymbol{y_i}$. The computation of those weights depends on the choice of attention. For example, in the additive attention mechanism (Bahdanau, Cho, and Y. Bengio 2015), they are computed as :

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{N} exp(e_{ik})} \tag{2.5}$$

$$e_{ij} = \boldsymbol{v}^T tanh(\boldsymbol{W}\boldsymbol{y}_{i-1} + \boldsymbol{V}\boldsymbol{x}_j + \boldsymbol{b}) \tag{2.6}$$

where $\boldsymbol{W}$ and $\boldsymbol{V}$ are matrices and $\boldsymbol{v}$ and $\boldsymbol{b}$ are vectors to be learned during the training at the same time as the others parameters of the network. A representation of the attention mechanism is drawn on Figure 2.3.

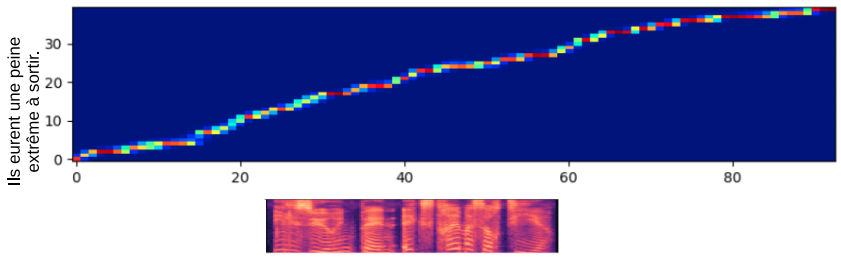


Figure 2.4 – Heatmap of the transposed alignment matrix computed between a sentence and the mel-spectrogram of a speaker reading that sentence.

Once every attention weights have been computed, they can be gathered in a matrix:

$$A_{x,y} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,N} \\ & \ddots & \\ \vdots & \alpha_{i,j} & \vdots \\ & & \ddots \\ \alpha_{M,1} & \cdots & \alpha_{M,N} \end{pmatrix} \tag{2.7}$$

It can be represented graphically as a heatmap where darker colors are associated to higher weights. This allows to interpret what the attention model learned. For example, Figure 2.4 represents the result of the attention between a text and the spectrogram corresponding to the reading of that text. In that case, the result of the attention model can be interpreted as an alignment between the text and mel-spectrogram. It matches the characters used to predict a given frame of the mel-spectrogram. As seen on the picture, the most salient weights follow a monotonous distribution function. Thus, the model learned that reading a text is done by looking at characters sequentially from left to right.

Since attention models are learned automatically, the derived alignments are prone to failure. The alignment can fail globally if no salient weights can be found for any timestep. This type of failure can also happen locally. For speech synthesis, additional alignment errors may arise:

— Repetition happens when the attention model focuses on characters already read which leads the synthesis system to repeat itself.

— Skip happens when the attention models focus on characters while skipping unread ones, leading the synthesis system to skip sounds or entire words.

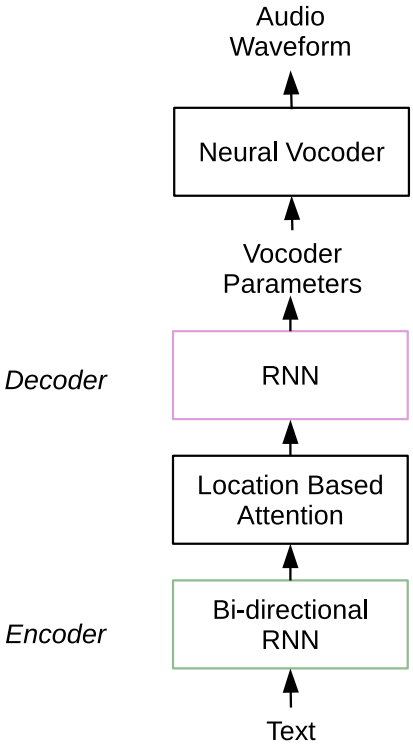


Figure 2.5 – Architecture of the Char2Wav TTS system.

Those errors are usually exacerbated when aligning long sequences (Battenberg et al. 2020).

Sequence-to-sequence models extended with attention have been successfully applied to speech synthesis. For example, Char2Wav (Sotelo et al. 2017) predicts vocoder parameters from text using this architecture, which is drawn on Figure 2.5. The encoder is a bi-directional RNN producing a sequence of character embeddings encoding each character contextually. The decoder is a single RNN predicting a frame of vocoder parameter from the previous frame predicted, and the output of an alignment model. Between the encoder and the decoder, location-based attention allows to align the sequence of character embeddings with the sequence of vocoder parameters. Finally, a neural vocoder converts the vocoder parameters into an audio waveform. Other models such as Tacotron (Wang, Skerry-Ryan, et al. 2017) or Deep Voice 3 (Ping et al. 2018) also adapt the sequence-to-sequence model with attention for speech synthesis.

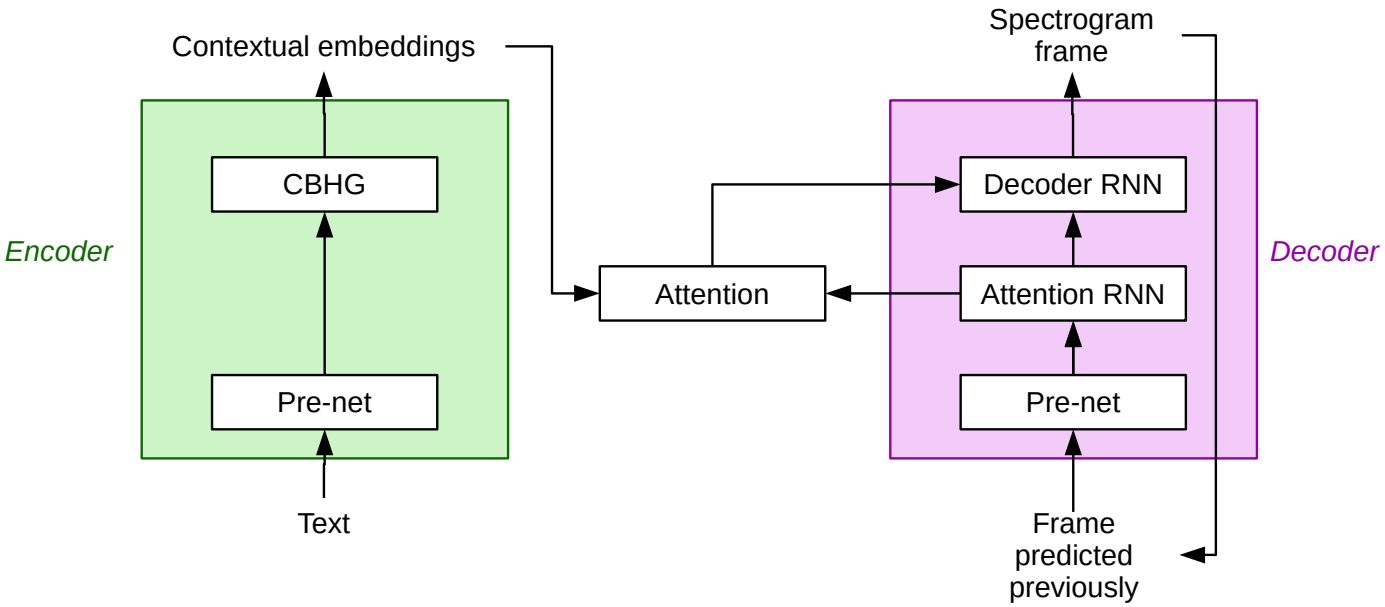


Figure 2.6 – The architecture of the first Tacotron model, introduced by (Wang, Skerry-Ryan, et al. 2017)

2.3 End-to-end Architecture: Tacotron

The family of architectures associated to Tacotron (Wang, Skerry-Ryan, et al. 2017) follows the sequence-to-sequence paradigm. As it is the basis of models used in Chapters 4 and 5, it is fully described here. As seen on Figure 2.6, textual data is encoded at the character level before being fed to an encoder to obtain a representation of each character in its context. Since Tacotron is an auto-regressive model, the prediction of the mel-spectrogram is done frame-by-frame, using previous predictions. The role of the attention model is to weight the character embeddings by importance relative to the frame being predicted. The decoder performs the prediction of the current frame of the mel-spectrogram from the result of the attention module and the previous frame being predicted. The following paragraphs describe each of those modules in further details.

Encoder

The encoder of the Tacotron model takes a sequence of characters as input, and outputs a sequence of contextual character embeddings. This high-level feature sequence encodes the identity of the character as well as the context in which it was written. The aim is that those high-level features are related to the pronunciation of the text.

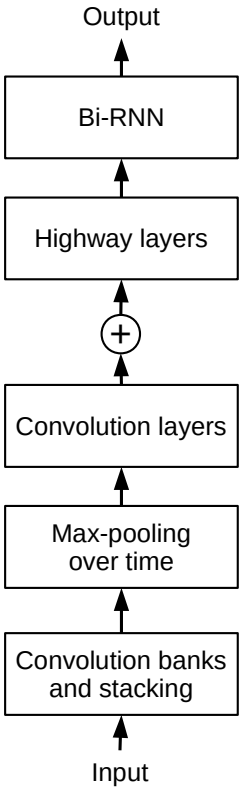


Figure 2.7 – The architecture of the CBHG module, reproduced from (Wang, Skerry-Ryan, et al. 2017)

To do so, the characters are usually encoded as one-hot vectors before being projected into a continuous space to obtain character embeddings (in orange on Figure 2.6). Then, the character embeddings are independently projected by a stack of non-linear projections dubbed "Prenet". This allows to obtain a new sequence of character embeddings with high-level features. Finally, to take into account the context in which each of the character were written, a Convolution Bank Highway Gated (CBHG) module is applied to them to obtain contextual character embeddings.

The CBHG module is the sequential application of a 1-dimensional convolution bank, a highway network and a bi-directional Gated Recurrent Unit (GRU) as seen on Figure 2.7.

The convolution bank is a set of N filters of varying width between 1 and N. The results of the convolution of those filters with the character embeddings are stacked. This gives a sequence of vectors akin to n-grams with $n \in [1, N]$. The convolution bank allows to take into account the context in which each character was written at different levels.

The sequence of n-gram-like vectors is then cut into windows of fixed width and the features over each window are max-pooled over time to summarize the n-grams of the windows. In order to keep a sequence of same length than the input sequence, the window slides with a stride of 1, and the sequence of n-grams is padded.

The summarized n-grams are then fed to multiple 1-dimensional convolutional layers and added to the original character embeddings via a residual connection before being fed to a highway network. Highway networks (Srivastava, Greff, and Schmidhuber 2015) are deep residual networks where both the residual connection and the layer outputs are gated by a learned function. This allows to learn networks with hundreds of layers. In the case of the CBHG, the highway layers are used to extract high-level features of character-level embeddings.

Finally, the outputs of the highway networks are fed to a bi-directional network, in this case a GRU. The use of a bi-directional layer allows to model the temporal dependencies both forward and backward in time. The output of this bi-RNN also constitutes the output of the encoder. This ouput sequence has the same length as the input character sequence. However, it is more than a character embedding since it encompasses information about the context in which it was written. As such, we will refer to those embeddings as contextual character embeddings.

Attention

The link between the encoder and decoder of the Tacotron model is done by an attention mechanism. The model uses the additive attention presented in section 2.2. The attention model attends over the contextual character embedding sequence. The attention is relative to the ouput of an RNN layer in the decoder.

Decoder

The decoder of the Tacotron model is recurrent. As such, the prediction at timestep $t-1$ is used during the computation at timestep t. Since the prediction of the decoder is high dimensional, a Prenet is used to lower its dimension. The output of the Prenet is then fed to the first RNN layer of the decoder. Since its output is used for the attention mechanism, it is dubbed "Attention RNN". This also means, that the attention mechanism attends over contextual character embeddings relative to an alternate representation of the mel-spectrogram predicted up to that point. The output of the attention mechanism and of the Attention RNN are concatenated. This concatenated vector represents both

the characters important for the prediction of the current frame of the mel-spectrogram as well as previously predicted frames. This concatenated vector is then fed to the second RNN of the decoder. Its output is the predicted mel-spectrogram frame. As such, it is dubbed "Decoder RNN".

Audio generation

After the decoder has run its course, the mel-spectrogram is completely predicted. In the original Tacotron system, the audio generation is done in two steps. First, the mel-spectrogram is converted back into a linear scale by using a trained CBHG module. Then, the audio signal can be reconstructed by using the Griffin-Lim algorithm on the linear spectrogram.

2.4 Neural Vocoder : WaveRNN

In order for TTS systems to use neural network solutions at every step of the pipeline, vocoders can also be modeled using neural networks. WaveRNN (Kalchbrenner et al. 2018) is a neural network generating audio designed has an alternative to WaveNet for faster generation, without any quality loss. The network predicts frames of a waveform auto-regressively: each prediction uses the previous one as an input. The neural network can be further conditioned on vocoder parameter or on spectrograms to be used as a vocoder.

The model introduces a new RNN layer designed as an extension of the GRU. The computation of a new hidden state h_t is made using previous hidden state h_{t-1} and input x_t using three separate gates. The computation of the layer (drawn on Figure 2.8 (a)) is as follows :

$$\begin{aligned} u_t &= \sigma(R_u h_{t-1} + I_u x_t) \\ r_t &= \sigma(R_r h_{t-1} + I_r x_t) \\ e_t &= \tau(r_t \times (R_e h_{t-1}) + I_e x_t) \\ h_t &= u_t \times h_t - 1 + (1 - u_t) \times e_t \end{aligned} \tag{2.8}$$

where $\times$ is the point-wise product, σ and τ are the *sigmoid* and *tanh* functions, and R_u, R_r, R_e, I_u, I_r, I_e are matrices learned during training.

The WaveRNN model (drawn on Figure 2.8 (b)) uses this RNN layer followed by

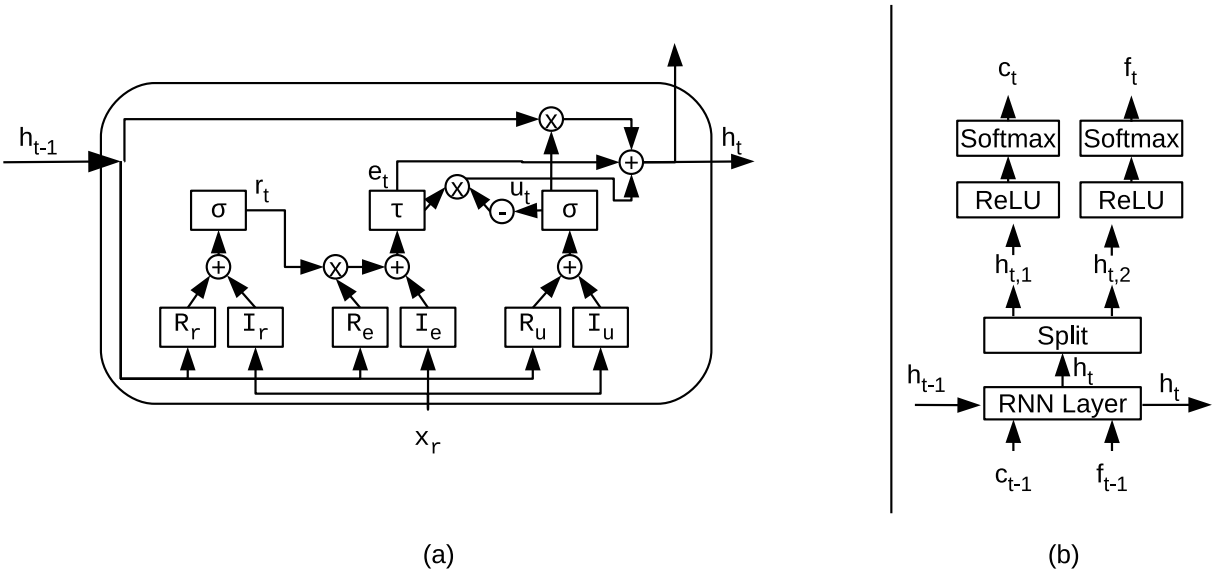


Figure 2.8 – (a) RNN cell used by WaveRNN. (b) Architecture of the WaveRNN vocoder.

a classifier to predict a waveform frame by frame auto-regressively. The waveform is considered as a sequence of 16 bit integers for each frame. Those bits are separated into 8 coarse bits c_t and 8 fine bits f_t. Thus, the input of the RNN cell is $x_t = [c_{t-1}, f_{t-1}]$ and the output of the network is $y_t = [c_t, f_t]$. The hidden state h_t of the RNN is separated into two halves $h_{t,1}$ and $h_{t,2}$, each responsible for the prediction of either c_t or f_t. The prediction is done using a classifier made of two stacks of two feed-foward layers first with a Rectified Linear Unit (ReLU), then a softmax activation function. The first stack predicts c_t from $h_{t,1}$, the second predicts f_t from $h_{t,2}$.

2.5 Modeling Components of Speech for Speech Synthesis

Usually, a TTS system is only able to generate speech with a single speaker voice and style. In order for the system to synthesize speech with different voices, two main approaches can be used: voice conversion and voice adaptation.

Conversion methods alter a speech sample so that is seems to have been spoken by a different speaker. This is usually done by transforming the acoustic representation of a speech signal. For example, in (Desai et al. 2009) (see (a) on Figure 2.9), a neural network is trained to perform a mapping between sequences of mel-cesptral coefficients extracted from utterances spoken by two different speakers. This method relies on a parallel corpus

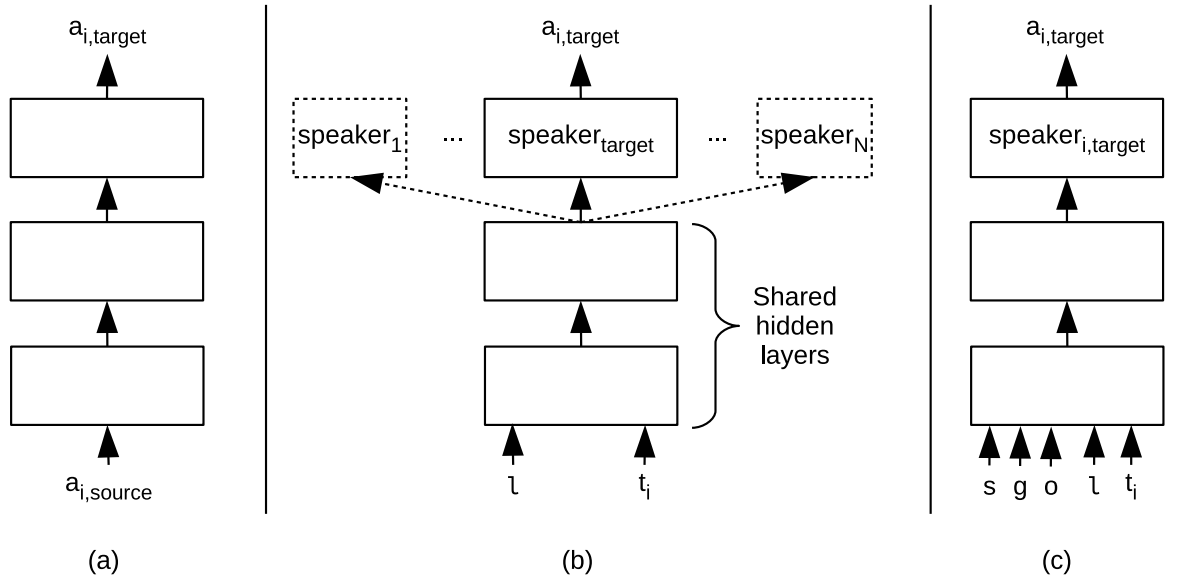


Figure 2.9 – (a) Neural network performing voice conversion by mapping each frame of acoustic features a_i from a source to a target speaker (b) Neural network acoustic model trained on a multi-speaker dataset with a stack of shared hidden layers then a speaker-specific layer per speaker. The acoustic features are predicted frame by frame from a linguistic description of the text l and the timing t_i of the frame being predicted. (c) Neural network acoustic model conditioned on speaker ID s, gender g and age o.

where sentences were read by pairs of speakers.

For voice adaptation, an average voice model is trained on a multi-speaker corpus. Then, the voice-independent model is altered or fine-tuned to match a target speaker's voice. This method has been successfully applied to both HMM (Yamagishi and Kobayashi 2007) and DNN based SPSS (Fan et al. 2015). In the second work, a feed-forward neural network (see (b) on Figure 2.9) is built to have shared layers followed by parallel speaker-specific layers, one for each of the training speakers, like in multi-task learning (Caruana 1997). This allows to train the model on multiple speakers, obtaining shared layers extracting features which are independent of each speaker's voice. Finally, the model can be adapted to new speakers by adding a new speaker-specific layer being trained on data from the relevant speaker while the weights of the shared layers are frozen.

Another method of voice adaptation consists in conditioning the statistical model on a one-hot vector of the speaker identity. In (Luong et al. 2017), a feed-forward acoustic neural network (see (c) on Figure 2.9) predicts frame of acoustic features $\boldsymbol{a_i}$ from linguistic features $\boldsymbol{l}$ describing the text, timing of the frame t_i, as well as information on each speaker

in the training set. The ID of the speaker in the database is encoded as $\boldsymbol{s}$ using one-hot encoding, the gender g and age o of each speaker is also fed to the neural network. Then, the adaptation to new speakers can be done by learning the best identity-vector $\widehat{\boldsymbol{s}}$ corresponding to that speaker. The weights of the neural network are frozen, it can be considered as a transformation function f. Then $\widehat{\boldsymbol{s}}$ is found by solving the following optimization problem:

$$\widehat{\boldsymbol{s}} = \underset{\boldsymbol{s}}{\operatorname{argmin}}(MSE(f(\boldsymbol{s}, g, o, \boldsymbol{l}), \boldsymbol{a})). \tag{2.9}$$

$\boldsymbol{s}$ is initialized randomly and updated through back-propagation in order to minimize the Mean Square Error (MSE) between the predicted and reference acoustic features.

The conditioning of an acoustic model for multi-speaker speech synthesis can be done with other types of speaker representations. For example, i-vectors (Dehak et al. 2010) have been successfully used to condition a DNN acoustic model for speech synthesis (Z. Wu, Swietojanski, et al. 2015). I-vectors are compact representations of a speaker's characteristics derived from a GMM model. Other types of speaker vectors could be used for speech synthesis, such as x-vectors (Snyder et al. 2018) or Learnable Dictionary Encoding (LDE) embeddings (Cai, Jinkun Chen, and M. Li 2018). Both are neural speaker identification vectors. They will be presented in further details in section 5.1.1. Such vectors can also be used to condition end-to-end systems such as Tacotron. For example, (Cooper, Lai, Yasuda, Fang, et al. 2020) concludes that using LDE embeddings instead of x-vectors with Tacotron leads to improved speaker similarity.

Conditioning a model on specialized vectors allows to tackle other types of speech components than speaker voice. For example, (Wang, Stanton, et al. 2018) introduces the notion of style token, representation of style components learned automatically in an unsupervised manner. The mechanism is drawn on the top part of Figure 2.10. A Tacotron model is conditioned on style embeddings extracted from reference samples. A sample is first encoded in order to get a single vector summarizing the whole utterance. Then, weights are computed using an attention mechanism to represent the contribution of each style token to the overall style expressed in the reference sample. Finally, the style embedding is computed as the weighted sum of the style tokens. The attention mechanism and the value of the style tokens are learned at the same time as the Tacotron model. Those tokens may capture speech styles such as gender, speaker identity or diverse prosody. While this method allows to model different speech components without supervision, a single

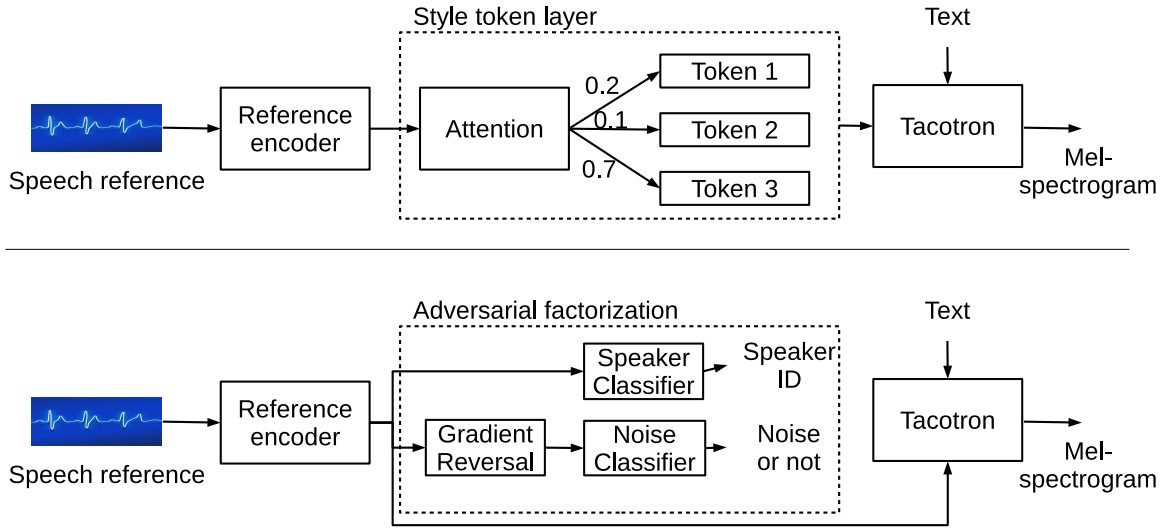


Figure 2.10 – Top : Mechanism behind style token embeddings. Bottom : Mechanism behind disentangling voice from noise.

style token might encode multiple speech components. Thus, it is not possible to control each speech component independently.

If labels exist, then the different speech components can be modeled in a supervised way. In (Hsu et al. 2019), the authors tackle two components at once : speaker identity and signal noise. The architecture is drawn on the bottom of the Figure 2.10. The Tacotron model is conditioned on a reference sample that is encoded by a speaker encoder. To disentangle speaker and noise information from the reference signal, two classifiers are added to the speaker encoder output. The first predicts speaker identity from the speaker embedding. The second predicts whether the reference sample contains noise. The loss associated to noise prediction is added in such a way as to penalize the speaker encoder if the presence of noise can be predicted from the speaker embeddings. This allows the speaker encoder to focus on encoding speaker characteristics and disregard noise characteristics.

2.6 Conclusion

Neural networks allow to design varied and complex solutions to a wide range of problems. For example, in speech synthesis, they can be used to design acoustic models for SPSS, to design end-to-end models, or to model different factor of speech variety. In

this manuscript, feed-forward neural networks will be used in Chapter 3 to train acoustic models. The sequence-to-sequence Tacotron and WaveRNN architecture will be used in Chapter 4 and 5. Finally, the embedding property of neural networks will be studied in every chapter of this manuscript. In particular, Chapter 3 and Chapter 4 study the properties of phone and character embedding respectively, while Chapter 5 applies this property to speech factors modeling.

Part II

Contributions

The works of this thesis began during the transition period where the standard for TTS went from approaches using either unit selection or statistical parametric acoustic modeling to fully neural end-to-end approaches. As such, the first work of this thesis studies an hybrid approach based on unit selection with elements borrowed from SPSS. It aims to reduce the amount of linguistic expertise in unit selection system by using phone embeddings. It also studies the notion of quality for phone embeddings.

The second work of this thesis applies end-to-end synthesis to the French language. To further the goal of linguistic expertise removal, we compare end-to-end models trained on characters to models trained on phonemes. Furthermore, we study the character embedding space defined by the end-to-end model and show that it discovers the phoneme structure of language and French phonetic information without any linguistic expertise supervision.

The third and final work of this thesis attempts to add more variety to end-to-end systems. By explicitly modeling speaker characteristics, we show that end-to-end systems can synthesize speech with the voice of different French speakers. We then attempt to model a second factor of speech variety, accent, independently of speaker characteristics.

Chapter 3

Lowering the need for linguistic expertise by using phone embeddings for Unit Selection

As presented in Section 1.3.2, unit-selection-based TTS concatenates prerecorded units of speech in order to generate a sample corresponding to the text to synthesize. In this work, we focus on the case where units are *phones*, i.e. the realization of a phoneme. As explained in Chapter 1, the unit selection process relies on a target cost which measures the similarity between an expected phone to be synthesized based on the input text and the candidate ones from a database of units. This target cost is usually defined as a distance between linguistic descriptions of these units. This definition raises two problems. First, linguistic expertise is needed to define the target cost. However, such an expertise can be expensive and might not be available for some languages. Second, such a target cost would only evaluate the similarity between units based on the linguistic context in which they were written. Since speech is an auditory phenomenon, a target cost should also measure how much two units might sound if they were read aloud. The aim of this chapter is to define a target cost with two properties.

1. It must be defined between linguistic descriptions of phones without linguistic expertise.
2. It should compare linguistic and acoustic characteristics of the two phones.

A potential solution to this problem is the use of embeddings from neural networks. Embeddings are the outputs of a hidden layer in a DNN. As such, they encode input data with respect to the task for which the network was trained. In particular, embeddings extracted from the neural acoustic models used in SPSS encode the linguistic description of phones with respect to their acoustic realization. Furthermore, embeddings encode categorical data in a numerical feature space where similar data are encoded in similar

locations. Thus, distances in the embedding space reflects similarity between data.

Because of those two properties, the proposed solution is to build a neural acoustic model to extract phone embeddings and to define the target cost of a unit selection system as the Euclidean distance between two phone embeddings. The closest work to ours is (Wan et al. 2017), which was presented in Section 1.3.5. In comparison, we use a simpler neural architecture, compare this hybrid approach to the usual expertly defined target cost, and investigate how the quality of the neural acoustic model impacts the quality of the speech synthesized by unit selection. With the hope to design a better architecture and training process, we also attempt to define the quality of a phone embedding space.

In this chapter, Section 3.1 introduces expert unit selection and how to extend it into a hybrid approach guided by neural networks. Section 3.2 presents different neural acoustic models, and how they can be adapted to extract phone embeddings. Section 3.3 studies the use of some of these models to performn hybrid unit selection, and compares this to the traditional expert approach. Finally, Section 3.4 explores the notion of quality for phone embeddings.

3.1 Expert and Hybrid Unit Selection TTS

In expert unit selection TTS systems such as the one drawn on the top part of Figure 3.1, the synthesis begins by analyzing the target sentence with the front-end. This allows to predict the pronunciation of words and other relevant information. More precisely, a sentence s is cut into a sequence of phones p_i. Each phone is represented using a linguistic description vector $\boldsymbol{l}_{p_i}$, containing the phoneme label of p_i, the phoneme labels of neighboring phones, the position of p_i in the sentence s, etc. More information about the content of $\boldsymbol{l}_{p_i}$ and how it is extracted can be found in Section 3.2.2. In the rest of this thesis, the linguistic description vector $\boldsymbol{l}_{p_i}$ will also be referred to as *linguistic features vector* or *linguistic descriptor*.

For unit selection, a speech database contains prerecorded units of speech at the phone level. The speech database is obtained using a speech corpus containing text and their readings. Each sentence in the corpus is aligned at the phone level with the audio before being analyzed by the front-end. As a result, each phone p_{DB_j} in the corpus is described with a linguistic description $\boldsymbol{l}_{DB_j}$ and a waveform w_{DB_j}. Additionally, a sequence of acoustic features $\boldsymbol{a}_{DB_j}$ can be derived from the waveform w_{DB_j}. Then, the speech database is composed of a tuple ($\boldsymbol{l}_{DB_j}$, w_{DB_j}, $\boldsymbol{a}_{DB_j}$) for each phone in the corpus.

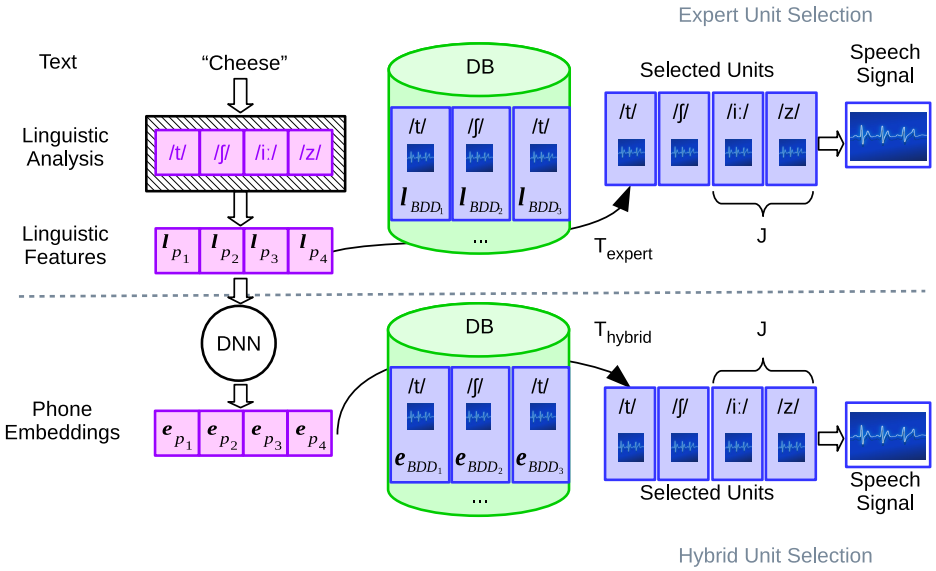


Figure 3.1 – Representation of expert (top of the figure) and hybrid (bottom of the figure) unit selection. $\boldsymbol{l}_{p,i}$ and $\boldsymbol{l}_{BD,i}$ refers to the linguistic feature vector of the i-th unit of a sentence or in the speech database. $\boldsymbol{e}_{p,i}$ and $\boldsymbol{l}_{BD,i}$ refers to the corresponding phone embeddings. T_{expert} is the expert target cost, T_{hybrid} is the hybrid target cost, and J is the join cost.

Then, speech is synthesized by concatenating phones in the speech database. The sequence of prerecorded phones is selected in a way to minimize the sum of two costs. The target cost T_{expert} measures whether phones in the speech database are good candidates for the phones in the text to synthesize. For an expert system, the target cost is defined with expertise on linguistic descriptions:

$$T_{expert}(p_i, p_{DB_j}) = expert(\boldsymbol{l}_{p_i}, \boldsymbol{l}_{DB_j}). \tag{3.1}$$

This cost is usually defined as a the sum of P sub-costs T_p between each of the linguistic description features (Taylor 2009), or on a subset of those features (Alain, Chevelu, et al. 2016):

$$expert(\boldsymbol{l}_{p_i}, \boldsymbol{l}_{DB_j}) = \sum_{p=1}^{P} T_p(\boldsymbol{l}_{p_i}, \boldsymbol{l}_{DB_j}) \tag{3.2}$$

In this work, each sub-cost is the comparison of boolean linguistic attributes between p_i and DB_j. We consider four types of attributes: textual, syllabic, positional, and phonological. Textual features describe the context the phoneme was written in, whether it was a

dialog or narration. Syllabic attributes describe the syllable the phoneme belongs: whether it has an onset or a conda. Positional features describe the position of the phone in the text in which it was written: is the phoneme the first of a word, the last of a sentence, etc. Phonological features describe the phone in articulatory terms (whether the phone is palattal, glottal, etc.), if is is nasal, etc. The list of all attributes used can be found in Table 3.1.

The join cost J is defined between two consequent speech units in a selected sequence. It estimates the quality of the concatenation between two phones. It is usually defined as the sum of Q sub-costs J_q between different acoustic characteristics of the phones. In this work,

$$\begin{aligned} J(p_{DB_j}, p_{DB_{j+1}}) &= \sum_{q=1}^{Q} J_q(\boldsymbol{a}_{DB_j}, \boldsymbol{a}_{DB_{j+1}}) \\ &= J_{MFCC}(\boldsymbol{a}_{DB_j}, \boldsymbol{a}_{DB_{j+1}}) + J_{F_0}(\boldsymbol{a}_{DB_j}, \boldsymbol{a}_{DB_{j+1}}) + J_{amp}(\boldsymbol{a}_{DB_j}, \boldsymbol{a}_{DB_{j+1}}) \end{aligned} \tag{3.3}$$

where J_{MFCC}, J_{F_0} and J_{amp} are sub-costs comparing MFCC vectors, fundamental frequency and amplitude of phones p_{DB_j} and $p_{DB_{j+1}}$. Each of those sub-costs is defined as the Euclidean distance on the respective acoustic characteristics.

To synthesize speech, units of the speech database are put in a lattice. Then, the sequence $\widehat{s}$ of phones selected to be concatenated is the one minimizing the sum of the target and join costs. To speed up the synthesis process, the amount of units put in the lattice can be limited to only a subset of the phones in the speech database using pre-selection filters (Alain, Chevelu, et al. 2016). In this work, we apply a single filter that selects only phones corresponding to the same phoneme as the one to synthesize. Then, the number of candidates put in the lattice is limited to best 25 candidate units for each phone to synthesize, according to the target cost.

The synthesis process is similar for hybrid unit selection, as drawn on the bottom part of Figure 3.1. The sentence to synthesize is linguistically analyzed to obtain a sequence of phones p_i and their respective linguistic features $\boldsymbol{l}_{p_i}$. Then, a DNN is used to extract embeddings $\boldsymbol{e}_{p_i}$ for each of those phone from linguistic features $\boldsymbol{l}_{p_i}$. Each phone in the database is described similarly using embeddings $\boldsymbol{e}_{DB_j}$ derived from the same DNN, instead of linguistic features. Since the size of the embeddings will have an impact on the storage size of the database, the embeddings are usually trained to be low-dimensional. Then, the target cost T_{hybrid} is computed as a measure d between the embeddings of

Textual features	
IS_TEXT_DIALOG	IS_TEXT_NARRATIVE
Syllabic features	
HAS_ONSET	HAS_CONDA
Positional features	
IS_IN_ONSET	IS_IN_CONDA
IS_LAST_PHONEME_OF_BG	IS_LAST_PHONEME_OF_WORD
IS_LAST_PHONEME_OF_SENTENCE	IS_FIRST_PHONEME_OF_WORD
IS_LAST_SYL_OF_BG	IS_IN_WORD_BEGIN
IS_IN_WORD_END	IS_SYLLABLE_BEGIN
IS_SYLLABLE_END	IS_LAST_SYL_OF_SENTENCE
Phonological features	
IS_PHONEME_LONG	IS_PHONEME_NASAL
IS_PHONEME_LOW_STRESSED	S_PHONEME_HIGH_STRESSED
IS_VOWEL	IS_PHONEME_LIQUID
IS_PHONEME_PULMONIC	IS_PHONEME_PLOSIVE
IS_PHONEME_FRICATIVE	IS_PHONEME_APPROXIMANT
IS_PHONEME_TRILL	IS_PHONEME_LATERAL
IS_PHONEME_FLAP	IS_PHONEME_DENTAL
IS_PHONEME_ALVEOLAR	IS_PHONEME_VELAR
IS_PHONEME_GLOTTAL	IS_PHONEME_FRONT
IS_PHONEME_BACK	IS_PHONEME_PALATOALVEOLAR
IS_PHONEME_RETROFLEX	IS_PHONEME_PALATAL
IS_PHONEME_UVULAR	IS_PHONEME_PHARYNGEAL
IS_PHONEME_EPIGLOTTAL	IS_PHONEME_NEARFRONT
IS_PHONEME_CENTRAL	IS_PHONEME_NEARBACK
IS_PHONEME_CLOSE	IS_PHONEME_NEARCLOSE
IS_PHONEME_NEAROPEN	IS_PHONEME_MID
IS_PHONEME_CLOSEMID	IS_PHONEME_OPENMID
IS_PHONEME_CLICK	IS_PHONEME_VOICEDIMPLOSIVE
IS_PHONEME_EJECTIVE	IS_PHONEME_BILABIAL
IS_PHONEME_LABIODENTAL	IS_PHONEME_ROUNDED
IS_PHONEME_DOUBLE	IS_PHONEME_AFFRICATE
IS_PHONEME_VOICED	

Table 3.1 – Categorical attributes used to define the target cost.

phones to synthesize and those of phones in the speech database. In this work, we use the Euclidean distance:

$$T_{hybrid}(p_i, p_{DB_j}) = d(\boldsymbol{e}_{p_i}, \boldsymbol{e}_{DB_j}) = \|\boldsymbol{e}_{p_i}, \boldsymbol{e}_{DB_j}\|_2 \tag{3.4}$$

The rest of the process is done the same way as for expert unit selection. A sequence of phones to be concatenated is selected in order to minimize to sum of the target and join cost.

In order to perform hybrid unit selection, a DNN needs to be trained to extract phone embeddings from linguistic descriptions. The next section introduces the architecture of the DNNs used in this work.

3.2 Acoustic Models

In this section, we introduce the four DNNs trained as acoustic models. While not all of them can be used to extract phone embeddings, they will allow us to investigate the consequences of modifying the usual acoustic model architecture to allow phone embedding extraction. After introducing the chosen architectures and training process, we will evaluate those neural networks as acoustic models.

3.2.1 Presentation of the DNN Models

At synthesis time, the only available information on the phone p_i to synthesize is its linguistic features vector $\boldsymbol{l_i}$. Thus, in order to perform unit-selection guided by phone embeddings, we need to train a neural network taking linguistic descriptors as input. Also, in order for the embedding space to reflect the acoustic aspect of speech, the neural network must predict an acoustic description of p_i. The acoustic models used for statistic parametric speech synthesis seem to be good candidates. However, adjustments to their architecture need to be made to allow the extraction of phone-level low-dimensional embeddings. The remainder of this section presents the architecture of four models applying those adjustments. A visual summary of the different neural networks is presented on Figure 3.2.

DNN-S The first model under consideration is a standard feed-forward acoustic model. It will be referred to as DNN-S, with "S" standing for "Standard". The predictions of

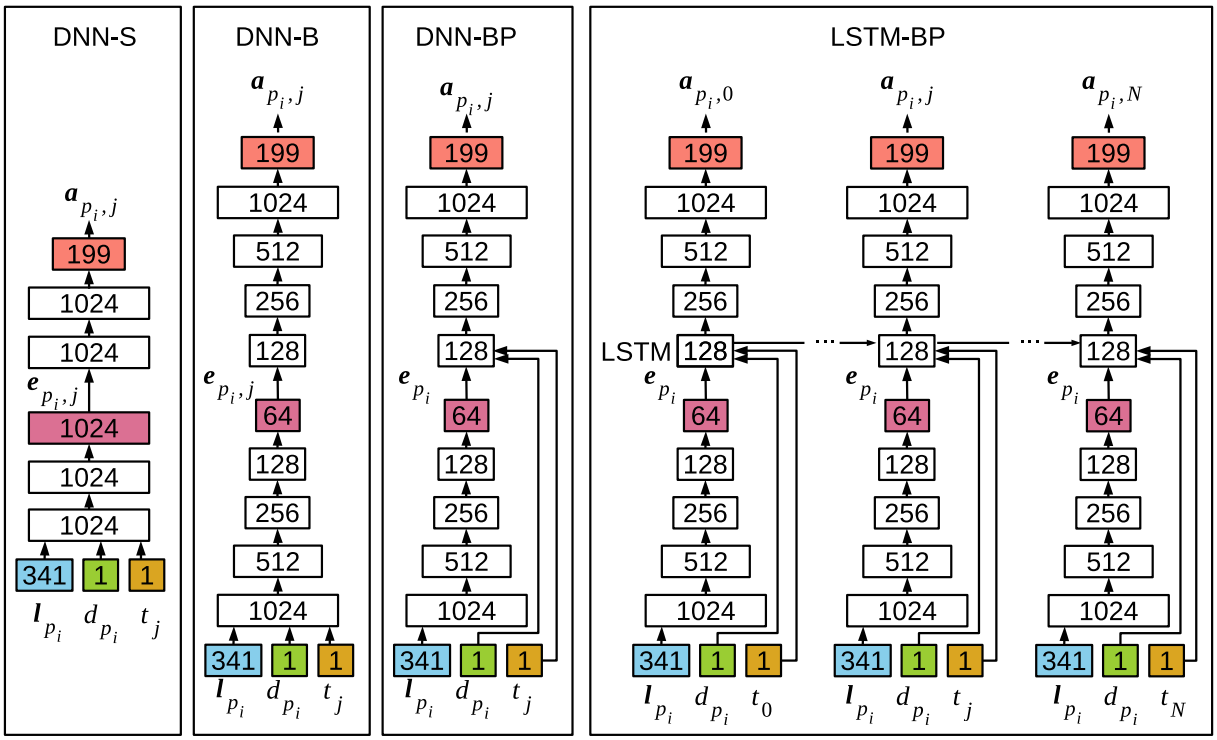


Figure 3.2 – Architecture of the 4 acoustic models.

an acoustic description of the speech signal is done frame by frame, independently. This means that for the j-th frame of phone p_i the acoustic representation $\boldsymbol{a}_{p_i,j}$ of that frame is predicted from the linguistic description $\boldsymbol{l}_{p_i}$ as well as a timing information encoding the position of the frame j in the phone. For all of the models presented in this section, the timing information is encoded as the phone duration d and the relative position of the frame in the phone : $t_j = \frac{j}{d}$. Similarly to (Z. Wu and King 2016), DNN-S has 5 hidden feed-forward layers of dimension 1024 with tangent hyperbolic activation function, while the prediction layer is linear. If we look at the output of the middle hidden layer, we get $\boldsymbol{e}_{p_i,j}$ a high-dimensional embedding of the j-th frame of phone p_i. However, for the purpose of hybrid unit selection, the embedding must be at the phone-level, computed only from $\boldsymbol{l}_{p_i}$. Furthermore, a low-dimensional embedding is preferred for low computation time and low storage footprint of the speech database.

DNN-B The second acoustic model answers the need for a low-dimensional embedding by using a bottleneck layer. It will be referred to as DNN-B, with "B" standing for "Bottleneck". As with DNN-S, the network is completely feed-forward. The prediction

is also done frame by frame using the linguistic description of a phone and the timing information of the frame as input. However, for DNN-B, the hidden layers assume a non-sequential "encoder/decoder" architecture. The inputs are encoded by going through 5 layers of dimension decreasing from 1024 to 64. The encoded information is then fed to the decoder composed of 4 layers of dimension increasing from 128 to 1024. Finally, a linear layer predicts the acoustic vector $\boldsymbol{a}_{p_i,j}$. Again, all hidden layers are feed-forward layers with tangent hyperbolic function. If we look at the output of the encoder, $e_{p_i,j}$, we get a low-dimensional frame embedding rather than a low-dimensional phone embedding as needed.

DNN-BP The third model answers the need for a low-dimensional phone embedding. A first solution to obtain a phone embedding would be to aggregate frame-level embeddings by pooling statistics like (Merritt et al. 2016) (see Section 1.3.5). Another solution could be using LSTM layers in the encoder and considering that the last frame embedding is also a phone embedding since it would be a representation taking all previous frames into account. However, since the phone embedding needed in our case must be obtainable solely from linguistic description, we adopt an approach similar to (Wan et al. 2017). The encoder only gets the linguistic description as input while the timing information is postponed and used in conjunction with the output of the encoder as input for the decoder. That way we do obtain a phone level embeddings e_i, obtained purely from linguistic information, and from which acoustic predictions are performed. Other than the timing information being postponed, the rest of the architecture is identical to the one of DNN-B. This model will be referenced to as DNN-BP, "BP" standing for "Bottleneck-Postponed".

LSTM-BP Finally, we consider a fourth model where the first layer of the decoder has been replaced by an LSTM layer in model DNN-BP. This model will be referred to as LSTM-BP. This model stems from the assumption that the quality of an acoustic model and the quality of the underlying phone embeddings are positively correlated. As LSTM layers have been shown to improve the quality of acoustic models by modeling the time-dependencies of speech (Z. Wu, Watts, and King 2016), the phone embeddings of LSTM-BP could outperform those of DNN-BP.

3.2.2 Dataset and Experimental Setup

The dataset used in this chapter is a private speech corpus. It corresponds to a French book, "Albertine disparue" written by Marcel Proust, read by a French male professional voice actor. Interestingly, due to the writing style of the author, the sentences are quite long and written in a formal register. Furthermore, since the text contains both narration and acted dialogues, the recorded speech is fairly expressive. In total, 3300 utterances are recorded for a total of 11 hours of speech. This amounts to around 390 000 phonemes. Due to the length of the corpus and the nature of the text, all phonemes of French are covered at least once. The rest of this section will present the pre-processing done on the data to obtain both the linguistic and acoustic features used for training the DNNs.

Linguistic Features The first step to obtain the linguistic descriptions is to apply the front-end of the TTS to all sentences in the dataset. This begins by converting the original text strings into sequences of phones. In order to do so, the phonetizer of the espeak TTS [1] is used. The phone transcription of words is predicted by first looking into a pronunciation dictionary then falling back on pronunciation rules for words not found in the dictionary. The pronunciation rules also allow to deal with the French "liaison", a phenomenon where a consonant ending a word might be silent or not depending on the following word.

The resulting annotation was stored along the original text using the ROOTS toolkit (Chevelu, Lecorvé, and Lolive 2014). This toolkit allows to store aligned textual sequences. In this work, it is used to keep the alignment between phones and word sequences. It allows to quickly obtain the word in which a given phone is used and thus to compute statistics such as the position of the phone inside the corresponding word. The final linguistic features used contains both categorical and numerical attributes. For each phone, categorical attributes are the quinphone identity (identity of the current, two previous and two following phones), identity of the syllable where the phone is used, articulatory features (description of how the human articulation apparatus should be set to pronounce the phone) and part of speech tag for the word where the phone is used, as well as the one of the previous and following words. All of the categorical attributes are one-hot encoded.

The numerical attributes represent information such as the position of the phone inside the word it was used, its position overall in the sentence it was used in, etc. A complete list of the attributes used for the linguistic description can be found in Table 3.2.2. In total the linguistic description is a vector of dimension 341. Those features are related to

1. http://espeak.sourceforge.net/

Feature	Dimensions
Phoneme label of the phone, two previous and two following	5*37 = 185
Description of the related syllable (label of the nucleus, position and size of the syllable, etc.)	30
Position of the phone relative to the syllable, word, sentence, etc.	17
Articulatory description of the phoneme (voiced, velar, alveolar, etc.)	38
Description of the related word (position, Part of Speech tagging, etc.)	60
Description of the phrase (number of syllables, words, etc.)	8
Description of the utterance (number of syllables, words, and phrases)	3

Table 3.2 – List of features used for linguistic description

that of the HTS toolkit (Zen et al. 2007).

Acoustic features For the acoustic features, we use the vocoder parameters of the WORLD vocoder (Morise, Yokomori, and Ozawa 2016). Considering frames with a size of 5 ms, 60 dimension MCC coefficients, a 5 dimension Band-APeriodicity (BAP) vector and the fundamental frequency F_0 are extracted. Since there are voiced and unvoiced phones, the F_0 values are discontinuous with a value of 0 on unvoiced phones and a strictly positive values on voiced phones. To deal with this discontinuity, the F_0 values are linearly interpolated on unvoiced phones and we use a boolean to remember if the phone was actually voiced or unvoiced. Furthermore, the logarithm is applied to the interpolated F_0. Finally to capture the dynamics of speech even though predicting frame by frame, we compute the delta (first order derivative) and delta-delta (second order derivative) of the MCC and BAP vectors as well as for $log(F_0)$. Delta and delta-delta computation for value for the frame t of value v is done as follows (Chapaneri 2012):

$$\Delta_v(t) = \frac{\sum_{i=1}^{D} i(v(t+i) - v(t-i))}{\sum_{i=1}^{D} i^2} \tag{3.5}$$

$$\Delta\Delta_v(t) = \Delta_{\Delta_v}(t). \tag{3.6}$$

In this work, we used $D = 2$. In total, the acoustic description is a vector of dimension 199.

The implementation was done using Keras with TensorFlow. Training was done on a GTX 1080 Ti, over 100 epochs using RMSPROP as an optimizer with the mean square

error as a loss function. The model weights with the best performance on the validation set were saved. Those models were trained using the true duration values.

3.2.3 Objective Evaluation

Before attempting to extract phone embeddings from the trained models to perform unit selection guided by phone embeddings, the quality of the underlying embedding space needs to be evaluated. However, no relevant measures currently exist to evaluate the quality of a phone embedding. Under the assumption that the quality of an embedding space is positively correlated to that of the underlying model, the quality of the phone embedding spaces can be approximated by evaluating the underlying DNNs as acoustic models directly. This evaluation will be done both objectively and subjectively.

To evaluate the four neural networks objectively as acoustic models, we measure the accuracy of their prediction. However, rather than simply measuring the loss function used during training, each of the parts of the acoustic description vector can be evaluated independently with a dedicated objective measure.

The acoustic description vector was composed of four components:

— MCC
— BAP
— voicing boolean
— fundamental frequency F_0.

The precision of the prediction of MCC and BAP coefficient can be evaluated through acoustically motivated measures introduced in Section 1.4: MCD and BAP distortion. The evaluation of the predictions for the voicing boolean can be done through an accuracy measure. Finally, the F_0 prediction can be evaluated by measuring the Root Mean Square Error (RMSE). These objective measures are computed between the predicted and reference acoustic features of the test set and reported on Table 3.3. For all objective measures, the lower the better.

As a sanity check, objectives measures from (Z. Wu, Watts, and King 2016) are also reported on Table 3.3. The architecture of that neural network is the same as DNN-S and the acoustic features are also extracted using the WORLD vocoder. However, the dataset used in (Z. Wu, Watts, and King 2016) and in this chapter are different, in particular the dataset of the reported paper is English while the one used in this chapter is French. Comparing DNN-S with the results of the reported paper, all measures are higher for our model, indicating a less accurate acoustic model. To some extent, the difference in

	MCD (dB)	BAP (dB)	V/UV (%)	RMSE(F_0) (Hz)
DNN-S	5.22	0.48	17.2	18.3
DNN-B	5.06	0.35	12.6	17.9
DNN-BP	5.09	0.36	13.7	18.2
LSTM-BP	5.80	0.49	19.7	19.5

DNN (reported from (Z. Wu, Watts, and King 2016))	4.54	0.36	11.38	9.57

Table 3.3 – Objective measures for all four acoustic models. The lower the better.

measures can probably be explained by the use of a different dataset, especially since the one used in this chapter is highly expressive. Since the magnitude of the values are similar, DNN-S pass the sanity check.

The impact of having an encoder/decoder architecture to obtain a bottleneck embedding layer can be evaluated by comparing the models DNN-S and DNN-B. Surprisingly, the model with a bottleneck has lower measures than the one without. However, the difference can not only be attributed to the presence or absence of a bottleneck layer. Since DNN-S has overall a higher number of parameters, it might be more difficult for the optimizer to reach an optimal solution in the number of epochs allotted for training. Inversely, even though DNN-B has a lower number of parameters, it has more layers than DNN-S allowing for more non-linearity. Despite not being able to conclude about the benefit of having a bottleneck layer, having one does not seem to hurt the prediction precision.

The impact of postponing the timing information can be evaluated by comparing the models DNN-B and DNN-BP. For all objective measures, DNN-BP has slightly higher values than DNN-B. This seems to indicate that postponing the timing information decreases slightly the quality of an acoustic model.

Finally, by comparing DNN-BP and LSTM-BP, we can evaluate the impact of replacing the first layer of the decoder in DNN-BP by an LSTM layer. Even though modeling time dependencies with the use of an LSTM layer should improve the quality of the acoustic predictions, LSTM-BP has much higher values than DNN-BP for all measures. This is due to the fact that LSTM-BP was wrongly trained. The three models under the label DNN-* work following the one-to-one paradigm: a single output vector is predicted from a single input vector in a frame-by-frame manner. However, LSTM-BP works under the one-to-many paradigm: predicting a sequence of output vectors from a single input vector. The

correct way to train such a network would be the following. For a phone p_i of duration d_{p_i}, all sequences of acoustic features $[\boldsymbol{a}_{p_i,0}, .., \boldsymbol{a}_{p_i,j}]$ with j in $[0, N]$ are independent training samples where the output vectors $[\boldsymbol{a}_{p_i,0}, .., \boldsymbol{a}_{p_i,j-1}]$ are discarded for the loss computation. Instead, only the sequence $[\boldsymbol{a}_{p_i,0}, .., \boldsymbol{a}_{p_i,N}]$ appeared in the training set with no outputs discarded for the loss computation. The model was not trained again using the correct method. Indeed, despite the poor performance of LSTM-BP as an acoustic model, it is interesting to keep it as a candidate for unit-selection guided by phone embeddings to evaluate the impact of the quality of an acoustic model on the resulting embeddings at synthesis time.

3.2.4 Perceptual Evaluation

In addition to the objective evaluation, a perceptual evaluation through listening tests is mandatory to assert the quality of our four DNN as acoustic models. To do so, synthesized speech for each of the models is obtained by predicting the acoustic features of phones in the test set and converting those features into an audio sample using the WORLD vocoder. In addition to the linguistic features, the timing information is also needed for the prediction of the acoustic features. Rather than use the real values in an oracle manner, a duration model is trained to predict the duration d_{p_i} of a phone p_i from the linguistic description $\boldsymbol{l}_{p_i}$. This duration model is a simple feed forward neural network with 6 hidden layers, each with an hyperbolic tangent activation function. The model is trained for 100 epochs and the weights of the better performing model are saved. At the end of the training, the mean absolute error of the duration model is 3.7 frames.

A MUSHRA-like listening test is conducted with 21 French native speakers. Each listener is asked to rate the overall quality of synthesized samples for 10 group of sentences on a range between 0 and 10 instead of the usual [0-100] range. The synthesized samples are the 105 sentences from the test set. Each sample is rated by two different listeners. In addition to the four models trained, natural speech is added as an hidden anchor, as well as samples synthesized in an analysis/synthesis manner. Those two systems are referred to as "NAT" and "VOC" respectively. Since the samples in the dataset are rather long, sentences longer than 5 seconds are cut at a point between breath groups so as to get a sentence of at most 5 seconds. The results are presented the box-and-whisker plot in Figure 3.3. Such a plot represents the distribution of a measure graphically. Each column correspond to a different system. Then, for a given column, the minimum and maximum value measured for that system are represented as the lower and higher part of each

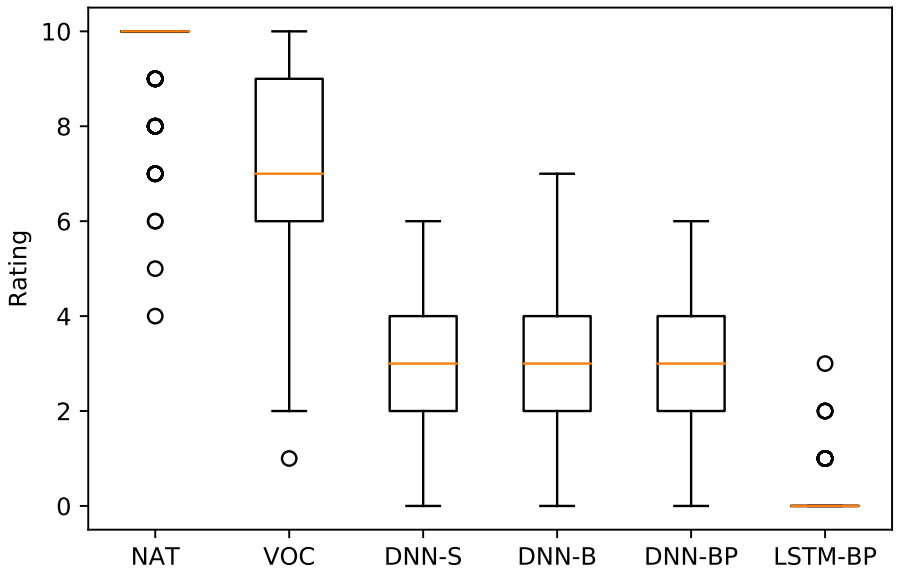


Figure 3.3 – Results of the listening test for the models in statistical parametric mode for in-domain utterances.

whisker. The first and third quartile is represented by the lowest and highest part of the box. The median is represented by a yellow horizontal bar inside the box. Outliners are represented as individual circles.

The natural samples were rated as perfect by most listeners, except for a couple samples rated below 8. Some of the shortened samples were deemed to have unnatural prosody, and thus rated lower than other samples from the natural system. Since the analysis-synthesis system was rated with a mean score of 7.2, the WORLD vocoder was deemed of "good" quality by the listeners according to the MUSHRA scale. Systems DNN-S, DNN-B and DNN-BP were rated with a mean score of approximately 3 (2.8, 2.8 and 3.0 respectively). Thus judging our acoustic models of "poor" quality and none can be distinguished from the other. This is coherent with the objective evaluation where the 3 models had slight difference in MCD values but significantly higher values than the state-of-the-art model reported. At the very least, displacing the timing information does not seem to perceptually lower the quality of synthesized speech. Finally, system LSTM-BP is rated with a mean score close to 0. The synthesized speech was judged mostly incomprehensible. This is also coherent with the objective measures, since LSTM-BP had

much higher MCD values than other systems.

Overall, through both an objective and subjective evaluation, it seems that three out of the four models trained perform as poor acoustic models, the last one being plainly bad according to the MUSHRA scale. However, the aim is to use the underlying embeddings rather the models themselves. It will be interesting to see how the quality of the acoustic models impact the quality of speech synthesized using unit selection guided by phone embeddings.

3.3 Hybrid Unit Selection Speech Synthesis Using Phone Embeddings

In this section, we build three unit selection systems. The target cost of the first one is defined with expert knowledge (see Section 3.1). The target cost of the two others is based on a comparison of phone embeddings, as presented in Section 3.2.1. Then, the three systems are compared in a listening test to determine the impact of automatically defining the target cost.

3.3.1 Presentation of the TTS Engines

Out of the four models trained, only DNN-BP and LSTM-BP can be used to extract phone-level embeddings. For unit selection synthesis systems, a speech database contains pre-recorded audio samples segmented into phones. For the usual unit selection method, the database is indexed according to the linguistic feature description vector of phones. In the case of unit selection guided by phone embeddings, those phones must instead be indexed by their phone embeddings.

The speech database is built in a similar manner to the pre-processing done in section 3.2.2, as seen on Figure 3.4. Every sentence s in the training set is analyzed by the front-end of the TTS system. This allows to obtain a sequence of phone p_i and corresponding linguistic vector $\boldsymbol{l}_{p_i}$ from sentence s. The phone sequence is then aligned with the audio sample w of the sentence to obtain (p_i, w_{p_i}) pairs. Acoustic features $\boldsymbol{a}_{p_i}$ are also extracted from each waveform w_{p_i}. The acoustic features considered are MFCC, amplitude, and fundamental frequency F_0. Then, the speech database is composed of each phone p_i in the dataset, described using a triplet $(\boldsymbol{l}_{p_i}, w_{p_i}, \boldsymbol{a}_{p_i})$. For hybrid unit selection, the

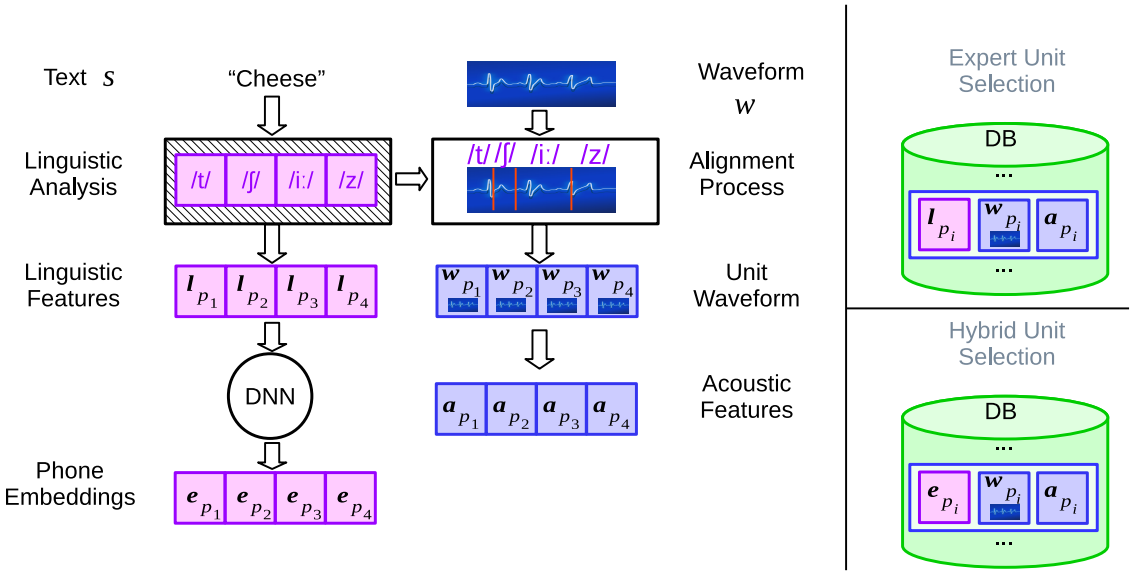


Figure 3.4 – Process to create the speech database for an expert and hybrid unit selection system.

linguistic features $\boldsymbol{l}_{p_i}$ are fed to either the DNN-BP or LSTM-BP neural network to extract the corresponding embedding $\boldsymbol{e}_{p_i}$. The embeddings are extracted from the middle layer, represented in pink on Figure 3.2. Then, the speech database is composed of triplets $(\boldsymbol{e}_{p_i}, w_{p_i}, \boldsymbol{a}_{p_i})$.

Similarly, at synthesis time, the target sentence is analyzed by the front-end to extract a sequence of linguistic description vectors. For hybrid unit selection, the linguistic vectors are additionally fed to DNN-BP or LSTM-BP to extract the corresponding phone embeddings. For each target phone, the 25 best candidates in the speech database according to the target cost are put in a lattice. The best sequence of candidate units is selected by optimising the sum of the target and join cost.

Depending on the choices made to define the target cost, we create 3 different unit selection TTS systems. For the two systems guided by phone embeddings, the target cost is defined as the Euclidean distance between the phone embeddings $\boldsymbol{e}_{p_t}$ and $\boldsymbol{e}_{p_i}$ of the target phoneme p_t and a candidate unit p_i extracted either from DNN-BP and LSTM-BP. For the expert system, the target cost is defined directly on the linguistic feature vectors $\boldsymbol{l}_{p_i}$ and $\boldsymbol{l}_{p_t}$ (see Section 3.1). The join cost is defined the same way for all system (see Section 3.1).

In total, we create three unit selection TTS systems :

— EXP: a classic unit selection system where the target cost is defined using linguistic

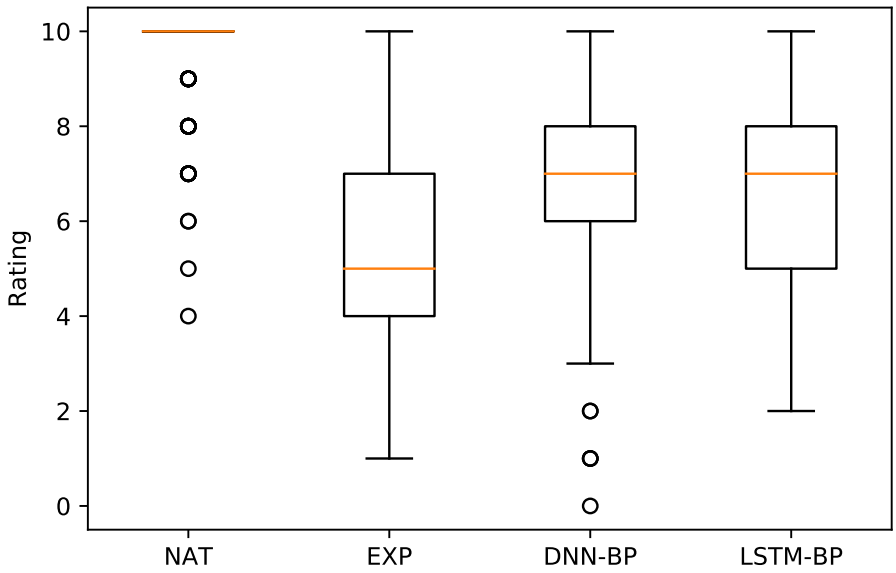


Figure 3.5 – Results of the listening comparing expert and hybrid unit selection

expertise

— DNN-BP: a unit selection system guided by phone embeddings extracted from the DNN-BP neural network.

— LSTM-BP: a unit selection system guided by phone embeddings extracted from the LSTM-BP neural network.

3.3.2 Comparison Between Automatic and Expert Cost

Similarly to the evaluation of the DNNs as acoustic models in Section 3.2.4, we perform a MUSHRA-like test to evaluate the quality of the three unit selection systems. The 105 sentences from the test set are synthesized by all three systems. The natural samples are added as a hidden reference. The maximum length of the samples is limited to 5 seconds by cutting them on breaths. The same 21 French native speakers are asked to rate the overall quality of the speech samples. Figure 3.5 presents the results.

As for the evaluation of the DNNs as acoustic models, a low number of natural samples were rated lower than 8 as listeners found the prosody of those samples unnatural because of the cut. The expert system was rated with a mean opinion score of 5.4, which corresponds to a fair score according to the MUSHRA scale. Both of the hybrid synthesis

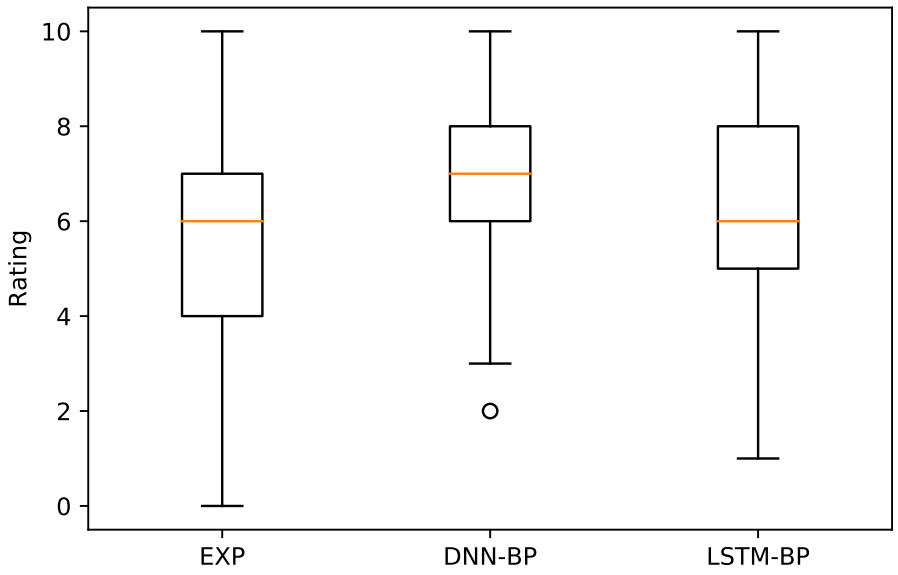


Figure 3.6 – Results of the listening test for the models working in unit selection mode for out-of domain utterances.

systems were rated with a mean opinion score of around 7 (6.8 for DNN-BP and 6.6 for LSTM-BP), which corresponds to a good score. The differences between the expert and hybrid systems is statistically significant, which suggests that defining the target cost using phone embeddings is an improvement over the manually defined cost through linguistic expertise. Interestingly, despite LSTM-BP performing poorly as an acoustic model, the difference between the system using embeddings from DNN-BP and LSTM-BP is not statistically significant. This means that even a poor acoustic model can be used to define the target cost of a unit selection system and still achieve good quality speech synthesis.

Since the definition of the target cost for the hybrid systems is the conclusion of a data-driven process, it seems important to assess whether the hybrid systems can still perform on sentences from a different domain. To do so, we perform synthesis on the Combescure dataset. This is a text dataset containing 100 sentences that globally share the same phoneme distribution as the French language. The sentences are shorter than in the audiobook used for the training, and the register is less formal. We perform a listening test in the same condition as the previous one, except that no natural samples exist for this dataset. The results of the listening test are reported on Figure 3.6.

The system with the expert system is rated with a mean opinion score of around 5.9, while the hybrid systems using embeddings from DNN-BP and LSTM-BP are rated with 6.9 and 6.3 respectively. The difference between the EXP (respectively LSTM-BP) system and DNN-BP is statistically significant. However, the difference between the EXP and LSTM-BP systems is not. This suggests that, outside of the training domain, the embeddings from an acoustic model can also be used to define the target cost of a TTS system that performs as well as the expertly defined one, or even better depending on the quality of the acoustic model.

Overall, through listening tests, we have shown that acoustic models can be used to define the target cost of a unit selection TTS system. Systems guided by phone embeddings perform as well or better than systems where the cost is defined with linguistic expertise. Interestingly, we have shown that an acoustic model producing incomprehensible speech in SPSS can still be used to perform good unit selection speech synthesis. To improve our understanding of the notion of quality of an embedding space for unit selection, the next section presents an analysis of the phone embedding spaces defined by different models.

3.4 Embedding Analysis

In the previous section, we ran a listening test in order to measure the quality of the trained embeddings for hybrid unit selection. However, listening test campaigns are long and potentially expensive to run. We need a quick way to assess whether a phone embedding space can be used for hybrid unit selection speech synthesis without having to run the full listening test. Usually, objective measures are used to estimate the perceptual quality of a TTS system. For example, the MCD of a neural acoustic model estimates the perceptual quality of a SPSS system. However, to the best of our knowledge, no objective measures have yet been defined for the quality of a phone embedding space.

The aim of this section is to define such objective measures. In doing so, we address the general question of how to assess whether an embedding is good or not for a given task without having to run that task. Section 3.4.1 presents the properties of embedding spaces trained in other domains as well as how they were evaluated. Section 3.4.2 presents the methodology followed to propose meaningful objective measures. Then Section 3.4.3 and following sections apply that methodology step-by-step.

3.4.1 Embedding Properties and their Evaluation

The quality of an embedding space E containing embedding vectors e_i is defined for a given property q. That quality can be measured in one of two ways: intrinsically or extrinsically. When measured in an intrinsic manner, the property q is evaluated using spatial criterion such as the position of points e_i within E. For example, in the case of word embeddings, a desirable property of the embedding space may be the ability to enable algebra properties that reflect semantic relationship between words, such as:

$$"king" - "male" + "female" = "queen" \tag{3.7}$$

The quality of the embedding space can be measured using a set of such relationships $(\boldsymbol{e}_{alg}, \boldsymbol{e}_{res})$ where $\boldsymbol{e}_{alg}$ is the word embedding computed using algebra and $\boldsymbol{e}_{res}$ is the embedding of the word expected as a result of the sum. The relationship is found true if the distance between $\boldsymbol{e}_{alg}$ and $\boldsymbol{e}_{res}$ is small enough. Then the quality of the word embedding space can be defined as the rate of relationships found true.

When measured in an extrinsic manner, the quality for property q of an embedding space E is investigated by measuring its potential use as input features to a machine learning model learned for a task related to q. The quality of the embedding space is then assimilated to the accuracy of the model. In the case of word embeddings, the quality property q being measured can be whether the embedding space encodes syntactic property. Then, a model predicting the part-of-speech from embeddings can be trained. In that case, the quality of the embedding space is estimated using the accuracy of the model trained as a proxy measure.

In the case of phone embeddings for unit selection speech synthesis, the quality property q being evaluated is the perceptual quality of speech synthesized by the system. With that definition of quality, the intuition is that, in a good embedding space, similar phones are close while two dissimilar phones are far away from each other. This similarity can be motivated either linguistically, i.e. two similar phones were pronounced in similar context, or acoustically motivated, i.e. two similar phones sound the same.

An intuitive intrinsic measure associated to the linguistic component would assert whether there is a small distance between embeddings $\boldsymbol{e}_{p_i}$ and $\boldsymbol{e}_{p_j}$ of phones p_i and p_j pronounced in similar contexts. This would amount to measure whether linguistic similarity is conserved in the embedding space. However, the definition of linguistic similarity is not trivial. Similarly, for the acoustic component, the definition on acoustic

similarity is difficult, especially since the waveform of each phone do not have the same length.

Intuitive extrinsic measures would evaluate the accuracy of a model trained to predict either the linguistic features $\boldsymbol{l}_{p_i}$ or acoustic features $\boldsymbol{a}_{p_i}$ of a phone p_i from the corresponding embedding $\boldsymbol{e}_{p_i}$.

The next section introduces the methodology followed to define meaningful measures of a phone embedding space's quality.

3.4.2 Methodology

To find realistic intrinsic criteria of the perceptual quality q of phone embeddings spaces, we tried to design objective measures that are able to make the distinction between embeddings that were shown to work for unit selection and embeddings that were purposefully trained to under-perform. More precisely, the remainder of the section follows this methodology:

1. Train at least two good embedding spaces E_{g_1} and E_{g_2}, i.e. ones that work well for unit selection speech synthesis.

2. Purposefully train bad embeddings spaces E_{b_j}, i.e. ones that do not work well for unit selection speech synthesis.

3. Perform a visualization of the embedding spaces to try to find distinctive criteria between one good embedding space, say E_{g_1}, and all the bad ones E_{b_j}.

4. Design objective measures m_k that are able to distinguish between this same good embedding space E_{g_1} and any bad one: $\forall k, \forall j, m_k(E_{g_1}) > m_k(E_{b_j})$

5. Validate the objective measures on other "good" embedding space, here E_{g_2}: $\forall k, \forall j, m_k(E_{g_2}) > m_k(E_{b_j})$

In that case, the objective measures m_k allow to distinguish a good embedding space from bad ones. Then for a new embedding space of uncertain quality E_i, its usefulness for hybrid unit selection can be estimated using m_k. Ideally, those measures should also allow to compare good embeddings E_{g_i} and E_{g_j} in such a way that:
$m_k(E_{g_i}) > m_k(E_{g_j}) \Rightarrow q(E_{g_i}) > q(E_{g_j})$.

3.4.3 Considered Embedding Spaces

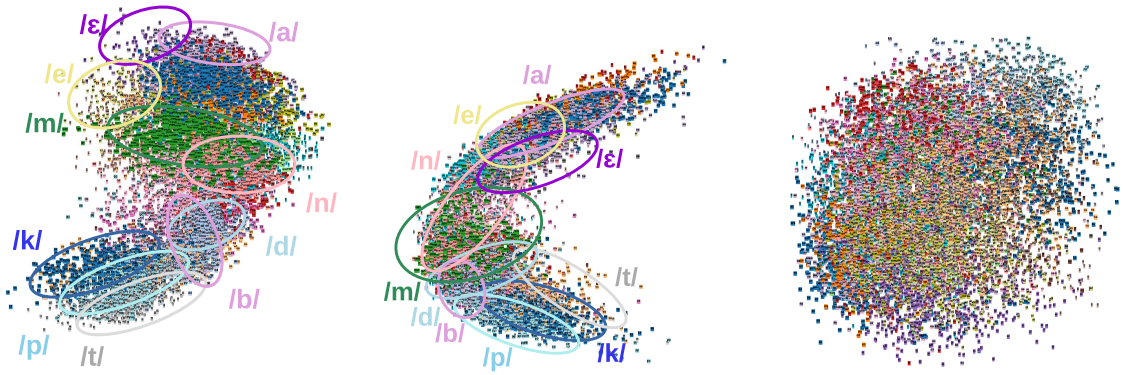
To follow our methodology, we need at least two embedding spaces E_{g_1} and E_{g_2} for which speech synthesis is of fairly good quality. In the previous section, we trained DNN-BP and LSTM-BP that both satisfy this criterion, by extracting 64-dimensional phone embeddings. DNN-BP will be used as the first good embedding E_{g_1} to design objective measures, while LSTM-BP will be used as the left-out good embedding E_{g_2} to validate the usefulness of the objective measures.

Regarding bad embedding spaces, we train two models with the same architecture as DNN-BP but in under-fitting or over-fitting conditions on a small amount of data. To do so, 256 phones are selected randomly from the training set, and only one random frame of each of those phones is used for the training. The underfitting model *DNN-BP-under* is trained for only one epoch on those 256 frames. The over-fitting one, *DNN-BP-over*, is trained for 50 epochs on the 256 frames. For both models, the weights obtained at the end of the last epoch are saved.

3.4.4 Embedding Space Visualization

For each of the phones in the test set, the phone embeddings corresponding to each model is extracted. For visualization purposes, the phones are then projected in a two-dimensional space using Principal Component Analysis (PCA). The results are presented on Figure 3.4.4. Each point on the figures is a different phone from the test set, and its color represents its corresponding phoneme. Since the number of colors available for the visualization is limited, some phonemes are represented with the same color

On Figure 3.7(a), while no clear clusters can be distinguished, phones corresponding to the same phoneme seem to be only found in a single part of the embedding space. For example, all the /t/ phones can be found in the bottom right part of the picture, while all /m/ phones are in the center. Interestingly, the structure of this phone embedding space can be described through linguistic similarity. For example, all unvoiced consonants are in the bottom of the figure, such as /t/, /k/ and /p/ phones. Similarly, all voiced consonants such /m/, /n/, /b/ and /d/ phones are in the center of the picture. Finally, vowels such as /a/ and /e/ are on the top of the picture. Furthermore, similar sounding vowels are close together such as the /E/ and /e/ phones are both on the left part of the picture. This leads us to believe the structure of the embedding space might also be respect some acoustic similarities.



(a) (b) (c)

Figure 3.7 – PCA visualization of embeddings from DNN-BP (a), DNN-BP-under (b), DNN-BP-over (c)

On Figure 3.7(b), again, no clear clusters can be distinguished but phones corresponding to the same phoneme can only be found in one part of the picture. For example, all /p/ phones are found in the bottom left part of the image. However, the groups of phones are harder to distinguish from one another. For example, half of the /n/ phones are mixed with /m/ phones, and the /b/ phones are mixed with /p/ and /d/ phones. The DNN-BP-under phone embedding space seems to follow the same linguistic structure as the DNN-BP embedding space. Unvoiced consonants such as /p/ and /k/ are found on the bottom of the picture. Voiced consonants such as /m/ and /n/ can be found in the center of the picture. Finally, vowels such as /a/ and /e/ are found in the upper part of the picture.

On Figure 3.7(c), it is hard to distinguish any structure. In particular, phones corresponding to the same phonemes only seem lightly grouped together, there does not seem to be any structure corresponding to vowel and consonant and the structure does not seem to follow any acoustic similarity.

This embedding space visualization gives insight into what makes a phone embedding useful for hybrid unit selection speech synthesis. A good embedding space groups phones according to the phoneme to which it corresponds, and it seems that similarly sounding phones are close in the embedding space. However, these are only insight and hypothesis. We still need to design objective measures corresponding to those criteria in order to try to distinguish embeddings spaces according to their quality.

3.4.5 Design of Objective Measures

The first distinctive criteria found during the visual observation of the phone embedding space is a structure where phones corresponding to the same phoneme are close together despite not forming clear clusters. This aspect can be objectively measured with intrinsic measures based on a nearest neighbor search. As drawn on Figure 3.8, each phone p_i is described using a linguistic description $\boldsymbol{l}_i$, acoustic vector $\boldsymbol{a}_i$ and embedding $\boldsymbol{e}_i$. For an integer K, the result of a K-Nearest Neighbors (KNN) search for phone p_i in the embedding space is defined as :

$$\text{KNN}(p_i) = [x_{n(1)}, ..., x_{n(K)}] \text{ with } x_{n(r)} = (\boldsymbol{l}_j, \boldsymbol{a}_j, \boldsymbol{e}_j) \tag{3.8}$$

where n is the ranking function. Once the phones p_j have been ranked according to a distance d to the embedding e_i, n permutes a rank r with the index j of the corresponding phone in the training set :

$$n : \mathbb{N} \to \mathbb{N} \text{ and } \begin{cases} n(0) = i \\ \forall r \in \mathbb{N}^*, d(e_i, e_{n(r-1)}) < d(e_i, e_{n(r)}) < d(e_i, e_{n(r+1)}) \end{cases} \tag{3.9}$$

To assess the first aspect of quality, for each phone p_i in the test set, we perform a 100-nearest neighbor search among phones in the training set. Then, it can be measured one of two ways :

1. By comparing the dominant phoneme label, i.e. the most frequent one, in KNN(p_i) to the one of p_i. This method correspond to measuring the accuracy of a KNN classification.
2. By counting the number of phones in KNN(p_i) whose phoneme label is the same as the one of p_i. This corresponds to measuring the purity of the nearest neighbors.

The second distinctive criteria found during the visual observation of the phone embedding space is a structure where similar sounding phones (in the case of vowels) are close in the embedding space while dissimilar sounding phones (vowels vs consonant) are separate in the embedding space. However, measuring an acoustic similarity between two phones p_i and p_j using an objective measures such as the MCD between p_i and p_j is not trivial. Indeed, since the two might not have the same length, they would need to be aligned first, using a method such as Dynamic Time Warping (DTW). This process might not be accurate and thus could impact the objective measures. Instead, we propose to

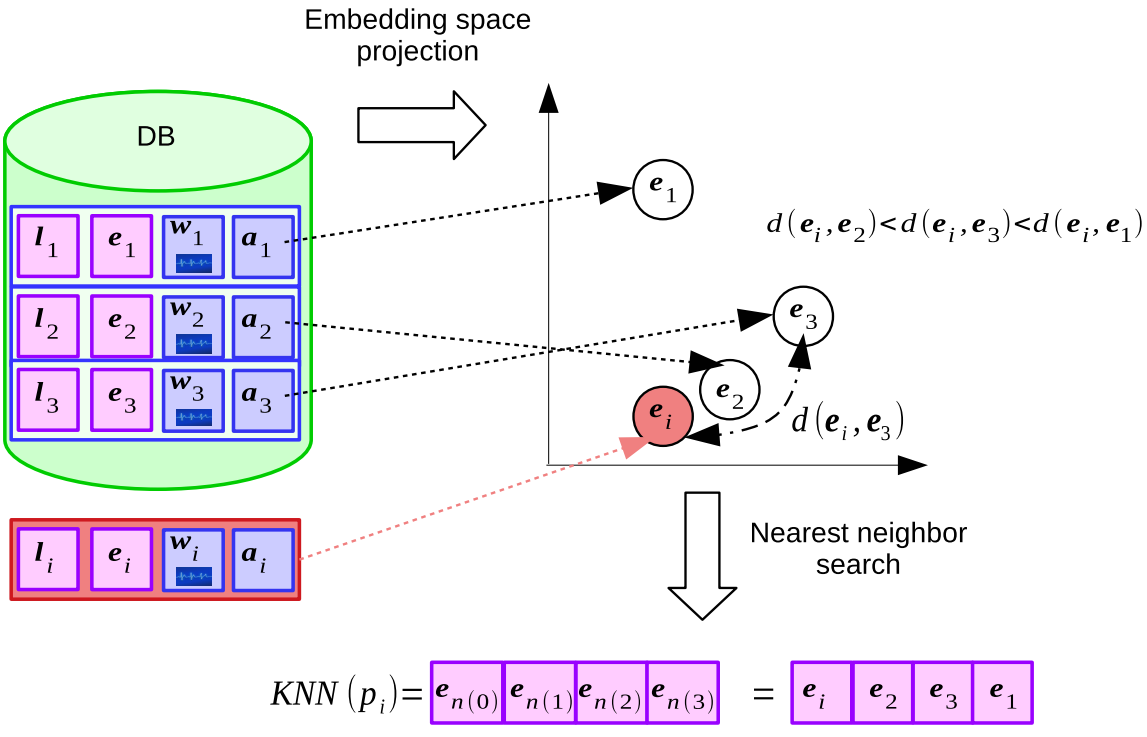


Figure 3.8 – Mechanism of a nearest neighbor search

evaluate the acoustic aspect extrinsically. We train a model predicting acoustic features $\boldsymbol{a_{i,j}}$ of frame j in phone p_i from the embedding e_i and timing information of the frame t_j. The model is trained on the training set and its accuracy measured on the test set is used to evaluate extrinsically the quality of a phone embedding space. When designing this model, it is important to note that its complexity may be considered as a bias in the interpretation of the results. Indeed, good results may come from the adequacy of its architecture for the target task, rather the from the quality of the embedding space. To avoid this ambiguity, we aim to keep the model as simple as possible, and use a simple linear regression model. This model maps $(\boldsymbol{l}_i, t_j)$ to $\boldsymbol{a}_{i,j}$. It is trained to minimize the residual sum of squares. This is similar to training a neural network containing a single linear layer to minimize the MSE.

3.4.6 Application of the Objective Measures

The objective measures are first applied on embeddings extracted from the models DNN-BP, DNN-under and DNN-over to see if they are able to distinguish between good and bad embedding spaces. Those measures are also applied to the embedding space

	DNN-BP	DNN-under	DNN-over	LSTM-BP
KNN-classification (%)	95.2	89.3	88.2	93.0
Neighborhood purity (%)	92.2	84.3	81.5	89.6
MCD (dB)	5.84	6.66	6.70	6.20

Table 3.4 – Objective measures of the quality of embedding spaces

defined by the model LSTM-BP to validate that those measures also allow to compare two embeddings of different quality for hybrid unit selection. The results of the objective measures are reported on table 3.4.

The nearest neighbour classification in the DNN-BP embedding space has a precision of more than 95%. This is slightly lower than 90% for both DNN-BP-under and DNN-BP-over. It shows that the KNN classification measure allows to distinguish between "good" and "bad" embeddings. Interestingly, the precision for all DNN-BP-* models is quite high, i.e. even our bad models. This suggests that a low amount of data is sufficient to learn an embedding that encodes somewhat faithfully the linguistic feature space. Similarly, the DNN-BP embedding space has a neighborhood purity of 92% while under/over-fitting models are under 85%. This gives rise to the same conclusion as for KNN classification. This is not surprising since both measures are closely related to each other. The linear acoustic model learned on DNN-BP embeddings has an MCD of 5.8 dB, while the MCD is higher than 6.5 dB for both DNN-BP-* models. First, this suggests that training a linear acoustic model on embedding space can also help to distinguish between "good" and "bad" embeddings. Second, the MCD of the linear model is much higher than that of original DNN-BP neural network (5.06 dB, see Table 3.3). This is not surprising since the multi-layer non-linear decoder of the original architecture is replaced by the equivalent of a single linear layer. This also suggests that most of the acoustic prediction is done by the second half of the neural network.

The proposed measures allow to distinguish between DNN-BP and the two DNN-BP-* models. However, we still need to make sure that those measures allow to compare other "good" embeddings with those "bad" ones. The sames measures were applied to embeddings extracted from LSTM-BP. For both linguistic measures, the DNN-BP-* models have lower precision than LSTM-BP. Similarly, the corresponding linear acoustic models have higher MCD than the one of LSTM-BP. This confirms that the proposed measures allow to distinguish between "good" and "bad" embeddings. When comparing DNN-BP and LSTM-BP, the measured values are all slightly worse for LSTM-BP. This indicates that the proposed measures might also allow to compare "good" embeddings.

3.5 Chapter conclusion

Overall, this chapter explored the use of phone embeddings for unit selection synthesis. We trained two neural acoustic models of varying quality in order to extract embeddings from the linguistic description of phones. We then compared the perceptual quality of an expert unit selection system where the target cost is defined with linguistic expertise, to hybrid systems where the target cost is automatically defined, as the Euclidean distance in a phone embedding space. A listening test shows that the systems with the automatic cost performed as well or outperformed the expert system. Interestingly, defining the target cost using phone embeddings extracted from a bad acoustic model do not lead a decrease of performance compared to the expert system. We then performed a visualization of the phone embedding spaces. We observed that a good embedding space encodes linguistic aspects of phones by grouping phones corresponding to the same phoneme together. Furthermore, the position of phones in the embedding space follows a distribution according to whether they are a voiced consonant, unvoiced consonant or a vowel. Similar sounding vowels are also grouped together. This suggests a good embedding might encode both linguistic and acoustic characteristics of speech. We measured the linguistic aspect intrinsically thanks to a nearest neighbor search. The acoustic aspect was measured extrinsically by learning a simple acoustic linear model from phone embeddings. Both measures allow to distinguish between good and bad embeddings, as well as between good embeddings of different quality.

The method proposed to define the target cost of a unit selection system allows to lower the amount of linguistic expertise. Indeed, the target cost is then defined automatically rather than using expert knowledge on the linguistic features. However, on the scale of a TTS as a whole, some amount of linguistic expertise remains. First, the choice of the features used in the linguistic description and their extraction is done using linguistic expertise. Second, the phonetization usually relies on pronunciation rules and/or a pronunciation dictionary. Both methods require extensive linguistic expertise. As a next step toward our goal of universal TTS, the next chapter investigates the use of end-to-end TTS systems to further remove the linguistic expertise. More precisely, we aim to replace entirely the front-end by a model learned automatically.

Chapter 4

Removing linguistic expertise following the end-to-end paradigm

Thanks to the increasing computer power and diversity of neural network layers available, DNNs are nowadays able to accurately approximate complex pipelines of data processing. For example, (Yao and Zweig 2015) learns a neural G2P following the sequence-to-sequence architecture. Ultimately, the successives steps of expert systems can even be entirely replaced by one or multiple DNNs. This new paradigm is called end-to-end. In (Arik et al. 2017), every component of the traditional TTS pipeline is replaced by a dedicated DNN. At best, the TTS pipeline becomes the two step-process drawn on Figure 4.1: (1) the generation by a *TTS model* of a mel-spectrogram corresponding to a given text, then (2) the conversion from mel-spectrogram to audio waveform using a *vocoder*. Tacotron (Wang, Skerry-Ryan, et al. 2017) is an example of such systems. Optionally, the input text can be mapped to phonemes prior to the TTS model.

End-to-end systems allow to lower the required amount of linguistic expertise to solve a task. Although they can be trained on sequences of linguistic and contextual features as in SPSS (Tian, Jing Chen, and Liu 2019), training of the sole corresponding sequence of phoneme labels has shown to be successful as well (Yasuda et al. 2019). This allows to remove the expertise needed to design and extract the linguistic and contextual features. As previously explained, the model can even be directly trained on the sequence of characters corresponding to the text (Sotelo et al. 2017; B. Li et al. 2019). This allows to remove the entirety of the linguistic analysis module in a TTS, thus completely removing

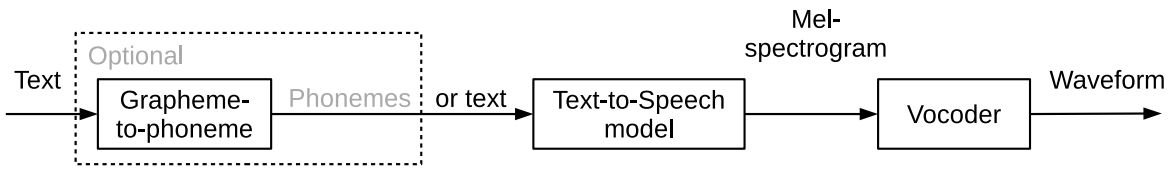


Figure 4.1 – TTS pipeline using end-to-end models.

linguistic expertise from the TTS pipeline. However, relying on the model to infer text readings may lead to pronunciation errors.

Few works involving end-to-end models have been conducted on the French language. The aim of this chapter is to study the potential use of characters instead of phonemes for French Tacotron-based speech synthesis. To do so, the difference between the two input types will be evaluated in terms of overall perceptual quality and pronunciation errors. Furthermore, the hidden representations learned by Tacotron models from characters are not yet well understood. For example, it is unclear whether Tacotron character embeddings are related to units of speech such as phonemes. Thus, we also perform an analysis of the character embedding space to gain insight into the kind of representations automatically learned by a Tacotron model.

In the remainder, Section 4.1 presents the Tacotron model used in this chapter. Section 4.2 introduces the French dataset on which it is trained. Section 4.3 compares a character-based model to a phoneme-based one through a listening test and Automatic Speech Recognition (ASR) experiments in order to evaluate the overall quality and amount of pronunciation errors. Finally, Section 4.4 performs an analysis of the character embedding space. It suggests that the Tacotron model encodes characters using a representation which is similar to phonemes. This leads us to investigate the use of character embeddings to train grapheme-to-phoneme modules.

4.1 Models

As previously stated, a TTS model and a vocoder are needed to get a complete end-to-end text-to-speech synthesis system. In this work, the former is trained using a slightly modified version of the Tacotron model, while the latter is trained using the WaveRNN architecture. The rest of this section describes these two neural networks.

4.1.1 Slightly Modified Tacotron TTS Model

In this chapter, we use an extension of the original Tacotron architecture presented in Section 2.3. The architecture is the same as in (Cooper, Lai, Yasuda, and Yamagishi 2020) and is drawn on Figure 4.2.

The model predicts a mel-spectrogram from a sequence of tokens describing a text to synthesize. We consider two versions of the model. In the first one, tokens are character

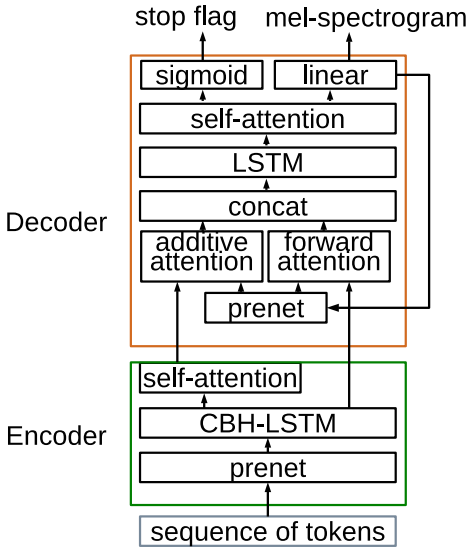


Figure 4.2 – Schema of the TTS model used to predict mel-spectrograms.

labels. For the second one, they are phoneme labels. These variants do not impact the architecture despite a different number of parameters in the first layer of the network due to the different vocabulary size of each type of token.

The architecture follows the sequence-to-sequence architecture. An encoder encodes the sequence of tokens as a sequence of token contextual embeddings that represents both a given token and the context it was written in. Then a decoder predicts a mel-spectrogram from those embeddings using attention alignment.

As in the original architecture, the sequence of tokens is first encoded as a sequence of independent token embeddings using a PreNet. Then, an extension of the CBHG is used to extract contextual token embeddings. The GRU layer is replaced by an LSTM layer with zone out regularization (Krueger et al. 2016), the module is thus dubbed *CBH-LSTM*. Finally, a self-attention module is added to the output of the CBH-LSTM. The addition of self-attention and the extension of the CBHG module allow to better capture long term dependencies (Yasuda et al. 2019). This results in the encoder having two outputs : the result of the self-attention and the outputs of the CBH-LSTM module.

In the decoder, each output of the encoder is attended over using two different attention mechanism with respect to the mel-spectrogram frame predicted at the previous timestep, encoded by a PreNet. The output of the self-attention is attented over using the additive attention defined in Section 2.2. The output of the CBH-LSTM module is attended over using forward attention (J.-X. Zhang, Ling, and Dai 2018). Then, the result of both

Encoder - Prenet	2 * (Dense(512, relu), Dropout(rate=0.5))
Encoder - CBHG	Convolution Bank : {Conv1D(128, t, relu), $t \in [1, 16]$} Max-Pool : MaxPooling1D(pool size=2, strides=1) Conv1D layers : Conv1D(512, 3, relu), Conv1D(512, 3, linear) Highway layers : 4 * {H: Dense(256, relu), T: Dense(256, sigmoid)}

Table 4.1 – Precise configuration of the Tacotron model

attention module is concatenated. According to (Yasuda et al. 2019), this configuration allows for a faster alignment learning while still capturing long-term information. The result of the concatenation is fed to an LSTM layer to model speech time-dependencies before its output is being attended over using self-attention. Finally, the result of the attention module is fed to two feed-forward layers. The first, with a *sigmoid* activation function, predicts whether the sample has been completely synthesized or is still ongoing. The second, with a linear activation function, predicts a frame of the mel-spectrogram.

The precise configuration of each of the modules of the Tacotron model can be found on Table 4.1.1.

4.1.2 WaveRNN Vocoder

The architecture of the neural vocoder follows the waveRNN model presented in Section 2.4, conditioned on mel-spectrograms. It predicts the audio waveform corresponding to a mel-spectrogram where each frame of the waveform is described as bits. Its architecture is drawn on Figure 4.3.

In order to condition the waveRNN model on mel-spectrograms, those features are first upsampled to match the sequence length of the audio waveforms using an upsampling network. It is composed of multiple instance of a module upsampling the features by repeating frames of the mel-spectrograms m s_i times, followed by a convolution layer of length $2 \times s_i$ to extract high level features. s_i is called the scale factor.

Then a frame m_t of the upsampled mel-spectrogram is concatenated to the coarse and fine bits c_{t-1} and f_{t-1} predicted at timestep $t-1$ in order to predict those of timestep t, c_t and f_t. Those features are used as the input of the RNN layer presented in Section 2.4. The dimensions of the output of that layer are separated into two halves $h_{t,1}$ and $h_{t,2}$. A classifier composed of a ReLU layer followed by a Softmax layer is used to predict c_t from $h_{t,1}$. An identical classifier is used to predict f_t from $h_{t,2}$.

In this work, the upsample network is composed of 3 upsampling modules with scaling

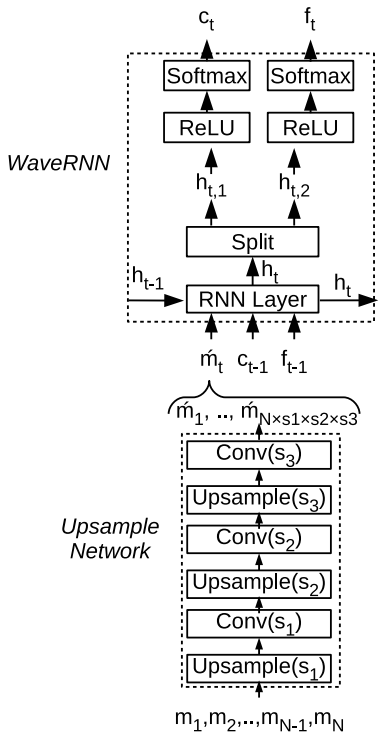


Figure 4.3 – Architecture of the neural vocoder.

factors 9, 8 and 4. In the waveRNN module, the RNN layer is of dimension 1024, the ReLU layers are of dimension 1024 and the softmax layers are each of dimension 256.

4.2 Data and Experimental Setup

This work has been made during an internship at the National Institute of Informatics, Tokyo, Japan. As a consequence, the private dataset used in Chapter 3 was unavailable. Instead, the data used in this chapter are taken from the SIWIS dataset (Honnet et al. 2017). This is a high-quality recording of around 11 hours of speech from a professional French female voice actor, with the corresponding texts. While the dataset is composed of 6 sections, we only used the 4 first part. The first section is readings of parliaments sessions, with 4500 sentences for 4h of audio. The makers of the dataset chose those sentences as to get *"the best possible diphone coverage"* and to contain only words among the 10 000 most common words in the French language [1]. The second part is the reading of five french books with 3500 sentences for 5h of audio. The third section is the reading of the french sentences

1. The rank of words were compiled on newspaper articles

Raw	M. Jean, âgé de 34 ans.
Normalized text	monsieur jean, âgé de trente-quatre ans.
Phonetic input	məsjø ʒɑ̃ <punctuation> aʒe də tʁɑ̃tkatʁ ɑ̃

Table 4.2 – Example of the pre-processing applied to a sentence.

in the SIWIS database (Goldman et al. 2016) with 75 sentences for 4 minutes of audio. Finally, the fourth section is the reading of 100 semantically unpredictable sentences for 5 minutes of audio. In total, we used 8175 sentences for a total 9h10 of audio. Hence, overall, the data used in this chapter come from a well-curated dataset. It covers a wide-range of useful vocabulary, with a good phone coverage, and recordings of high quality.

The dataset is split into train/validation/test subsets. All sentences from every part of the dataset are mixed. Then, 200 sentences are randomly chosen for the test set. Out of the remaining sentences, 400 are randomly drawn for the validation set. Finally, the training set is made of the remaining 7575 sentences.

To limit the use of linguistic expertise, the amount of textual preprocessing was kept as low as possible. An example is reported in Table 4.2. It involved only replacing common abbreviation by their full form (eg. "M." would be expanded into "*Monsieur*") and replacing the numbers written in arab numbers by their full form (eg. "34" would be expanded into "*trente-quatre*"). The punctuation was kept as it was and all characters were lower-cased. Since two versions of the models are to be trained, either on characters or on the phoneme transcription, phonetization of the cleaned sentences was performed using the phonetizer of the Espeak TTS as in Chapter 3. All punctuation mark in the phoneme sequence were replaced by a single special token. Finally, the sequences of tokens (characters or phonemes) were one-hot encoded using the respective size of their alphabet.

On the acoustic side, the samples were downsampled to 24kHz before removing leading and trailing silences. Silence removal is done by cutting audio at the beginning and end of the sample where the amplitude of the signal is below 20 dB. Finally, 80-dimensional mel-spectrograms are extracted with frame of 5ms.

Two version of the Tacotron models are trained. The first one is trained on character inputs, the second one on phoneme inputs. The two models are trained for 10 days on a GPU.

4.3 Comparison of Phoneme and Character-based Tacotron

The aim of this chapter is to assess to what point raw characters can be used as input of an end-to-end model for speech synthesis, rather than a phoneme sequence. To do so, samples synthesized following both methods are compared using a listening test in Section 4.3.1 and using objective measures in Section 4.3.2.

4.3.1 Listening test

A listening test is performed to compare speech synthesized using a phone transcription as input with speech synthesized from raw characters. A random subset of 50 samples from the test set are selected for the listening test. Mel-spectrograms corresponding to the text of those samples is synthesized using both the phoneme and character Tacotron model. The mel-spectrograms were manually checked for alignment errors. Out of the 50 samples, only 2 contain alignment errors for the phoneme model, while none were found for the character model. The mel-spectrograms are then converted to audio using the WaveRNN vocoder. For the purpose of evaluating the quality of the waveRNN model, reference mel-spectrograms are extracted for the 50 samples and converted back to audio using a waveRNN neural vocoder.

At each step of the listening step, the listeners get the text of a sentence and 4 corresponding audio samples in a random order. Those samples correspond to natural speech, the output of the WaveRNN model on the reference mel-spectrograms and speech synthesized by the two Tacotron models. The listeners are then asked to answer two questions. They are first asked to rate the overall quality of the audio samples on a 0-100 scale. The listeners are instructed that for each set of samples, at least one sample must be rated with a value of 100. The scale is annotated with the following values :

— 0-20 : Bad
— 20-40 : Poor
— 40-60 : Fair
— 60-80 : Good
— 80-100 : Excellent

The second question asks the listeners to rate the level of perceived pronunciation error by comparing the audio samples to the text of the synthesized sentence. Listeners

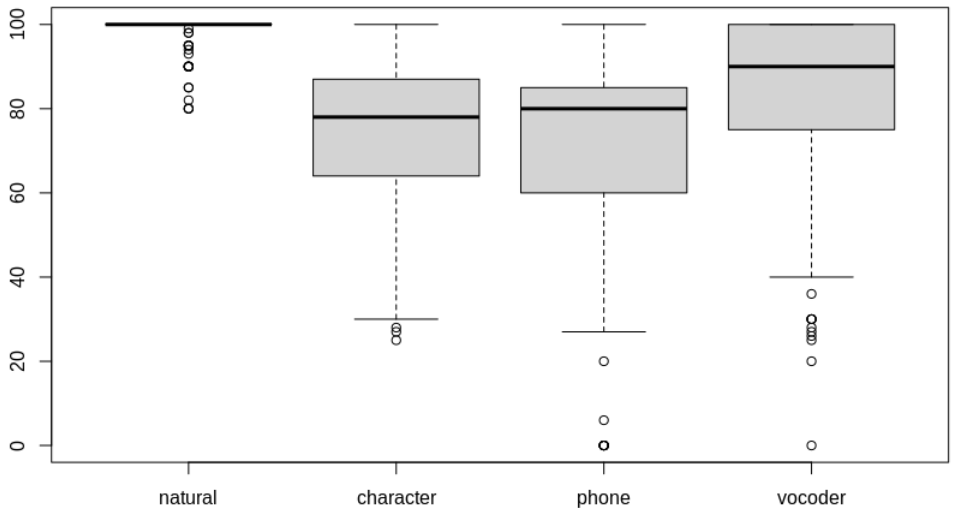


Figure 4.4 – Whisker plot of the scores for the overall quality

were free to consider what a pronunciation error was. However, they were informed that they could expect errors such as mismatch between the text and audio, as well as missing or superfluous liaisons. The listeners were asked to rate the level of pronunciation error on a 1-5 scale annotated as follow :

— 1 : Complete mismatch between the text and audio
— 2 : Lots of pronunciation errors
— 3 : Pronunciation error noticeable and bothering
— 4 : Pronunciation error noticeable but not bothering
— 5 : No perceived pronunciation error

Listeners were instructed that, unlike the first question, there is no need to have at least one sample per set rated with a value of 5. This question is meant as a proxy of the usual intelligibility test. This allows to avoid the lengthy and complex task of asking listeners to manually transcribe the speech signals.

12 french volunteer listeners took part in the listening test and rated 25 sets of samples each. Thus, each of the synthesized samples are covered by 6 listeners. Whisker plots reporting the results of the listening of the listening test can be found in Figure 4.4 (quality) and 4.5 (pronunciation errors).

Let us begin by examining the impact of using characters as inputs rather than

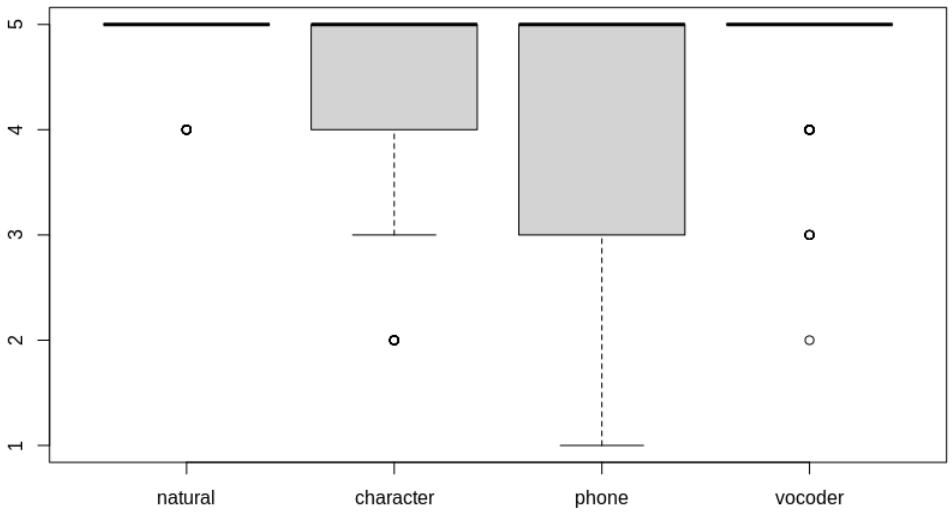


Figure 4.5 – Whisker plot of the scores for the pronunciation errors

phonemes on the overall quality. Unexpectedly, the natural speech samples were rated almost exclusively with value of 100, with only a couple values down to 80. In contrast, the samples synthesized using the reference mel-spectrograms and waveRNN as a vocoder were rated with a mean opinion score of 82.96. This indicates that the waveRNN model is well trained and gives an upper bound to the quality of synthesized speech. As seen on the whisker plot, speech synthesized from the character and phone models were rated fairly similarly with the character model receiving a mean score of 73.83 and the phone model a mean score of 71.93. According to a Student test with 95% confidence interval, the difference between the two mean values is not statistically significant since the p-value is higher than 0.005. From this listening test, we can conclude that using character as input does not lower the overall quality of synthetic speech in comparison with a model trained on a phoneme transcription.

One of the main reason of using phones as input rather than characters is to make sure that the model does not mispronounce words of the sentence to synthesize. Thus, it is important to observe whether or not listeners perceived noticeably more pronunciation errors for the character-based model. Unexpectedly, since the test data correspond to speech recorded by a professional voice actor, there is no pronunciation errors in the natural samples and so they received a mean opinion score of 4.93. Similarly, the samples

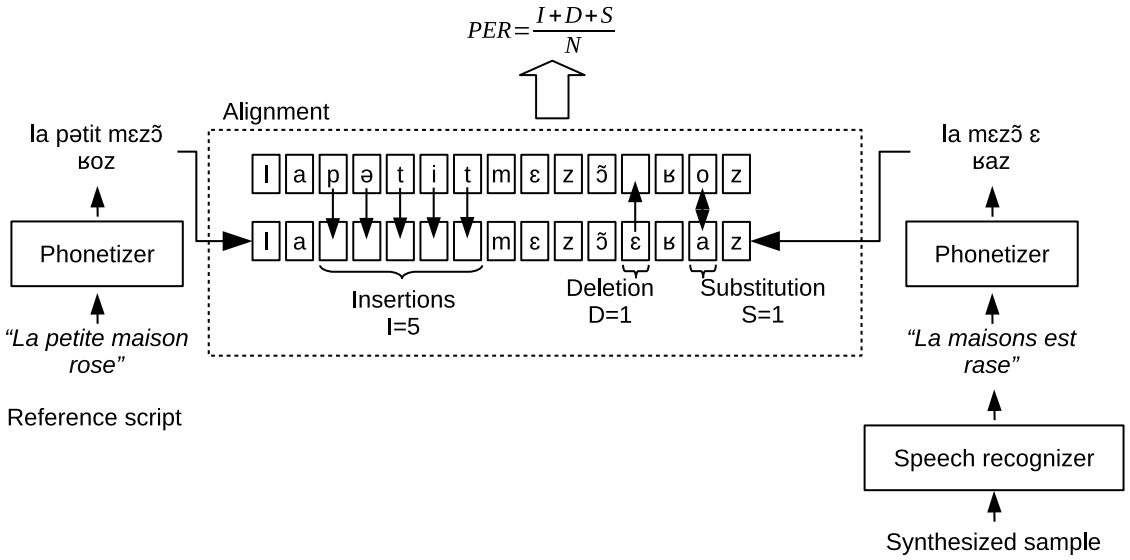


Figure 4.6 – Method applied to measure the amount of pronunciation errors introduced by the synthesizing process.

synthesized from the reference mel-spectrograms do not contain pronunciation errors and thus were rated with a mean opinon score of 4.79. The character and phone models were rated with mean scores of 4.26 and 4.06 respectively. Again, the difference between the two mean values is not statistically significant. This indicates that using character rather than phones does not increase the amount of pronunciation errors.

4.3.2 Further Investigations on the Pronunciation Errors

To strengthen the claim that character-based TTS does not degrade pronunciation prediction, we attempted to objectively measure the amount of pronunciation errors of the character and phone-based models following the method drawn on Figure 4.6. This is done by transcribing the synthetic speech samples using an ASR system. The amount of pronunciation errors can be measured extrinsically by comparing the predicted transcript to the actual text of the synthesized sentence and measuring how much they match.

For this study, we used a readily available speech recognizer : the IBM Watson Speech to Text service. This cloud-based service allows to upload audio samples in a variety of language (among which French) and receive the corresponding predicted sequence of words. The prediction is done using two models. The acoustic model predicts the most probable sounds corresponding to an audio sample. The language model predicts

	Natural	Character	Phoneme	Vocoder
SIWIS sentences	0.04926	0.07256	0.07712	0.07845
Tongue-twisters	NA	0.23267	0.23996	NA

Table 4.3 – Mean Phoneme Error Rate for sentences from the SIWIS listening test or from the tongue-twister dataset

what word those sounds are most likely making, taking into accounts the previously predicted words. The language model was not fine-tuned to our own dataset. To attempt to remove the errors of transcription due to the language model, the predicted transcript and reference text are both phonetized using the phonetizer of espeak. Espeak was also used to predict the phoneme sequences used as inputs of the phoneme-based Tacotron. This removes potential pronunciation errors introduced by the espeak phonetizer from the evaluation. The phonetization step allows to remove errors such as predicting the word "*maison*" in singular instead of the plural "*maisons*" which sound exactly the same. Similarly, it allows to remove error due to conjugation and homonyms. The pronunciation errors is then estimated by measuring the Phoneme Error Rate (PER) between the phoneme sequences of the predicted transcript and reference text. The PER is computed by aligning two phoneme sequences using the Levenshtein distance. Then, the number of substitutions S, deletions D and insertion operations I to transform a predicted sequence s_{pred} of length M into a reference sequence s_{ref} of length N are counted. Finally, the PER is computed as :

$$PER(s_{pred}, s_{ref}) = \frac{S + D + I}{N} \tag{4.1}$$

For this experiment, we used sentences from two dataset. In addition to using the sentences of the test set, we collected a set of 200 tongue-twister French sentences. This second dataset allows to study how both models perform on hard-to-pronounce sentences. The experiment is also applied on the natural sentences of the test set as well as the corresponding analysis-synthesis samples. In the case of the tongue-twister sentences, since the dataset is purely textual, there are no audio references. The result of the experiments can be found on Table 4.3.2.

When looking at the measures for the sentences from SIWIS used for the listening test, the differences between the PER of each system are not statistically significant. Despite not being able to conclude anything from the measure on this dataset, this allows to have a baseline for the mean PER. For the tongue-twister sentences, the PER measured for both the phoneme and character systems is much higher than the baseline value on

normal sentences. This is to be expected since tongue-twister sentences are both harder to pronounce and to comprehend. Interestingly, the difference between the measured values for the character and phoneme systems is not statistically significant. This reinforces the claim that using characters instead of a phone transcription as input of an end-to-end TTS french system does not lead to more pronunciation mistakes.

4.4 Analysis of the Embedding Space

4.4.1 Visual Analysis

The perceptual and objective evaluation of pronunciation errors showed that character can be used directly for end-to-end speech synthesis. However, this did not give any insight into why they are a good alternative. In particular, we have no idea about what kind of representation is learned by the end-to-end model. To that end, we propose to visually observe the contextual character embedding space.

In this work, we focus on the embeddings that can are extracted by the last layer of the encoder, the output of the bi-directionnal recurrent layer. For a sentence $s = (c_1, .., c_N)$ with N characters, the encoder extracts a sequence of N contextual character embeddings $(e_1, .., e_N)$. We extract embeddings for all the sentences in the test set using the character model. For visualization purposes, for each embedding, we register the identity of the corresponding character c_i as well as the previous and following characters (c_{i-1}, c_{i+1}) in order to deduce the pronunciation of the corresponding phoneme.

The result of a t-SNE visualization (Maaten and G. Hinton 2008) of the embedding space can be found on Figure 4.7. For the sake of visualization purpose, the amount of points is limited to 3000 randomly chosen contextual character embeddings. Overall, the t-SNE visualization allows to discern group of embeddings. When looking more precisely, those groups sometimes contain more than one type of characters, and the same character can be found in different groups. The t-SNE is used here rather than the PCA as in Chapter 3 in order to analyze the composition of groups rather than their relative positions.

On the upper right part of the figure, At least four different groups of the character "*n*" can be found. The first group contains only the character "*n*" and corresponds to the embedding of a character written in contexts such as "*nne*" or "*ène*". All of those corresponds to the "/n/" phone. Then the three other groups mix both "*n*" and "*m*"

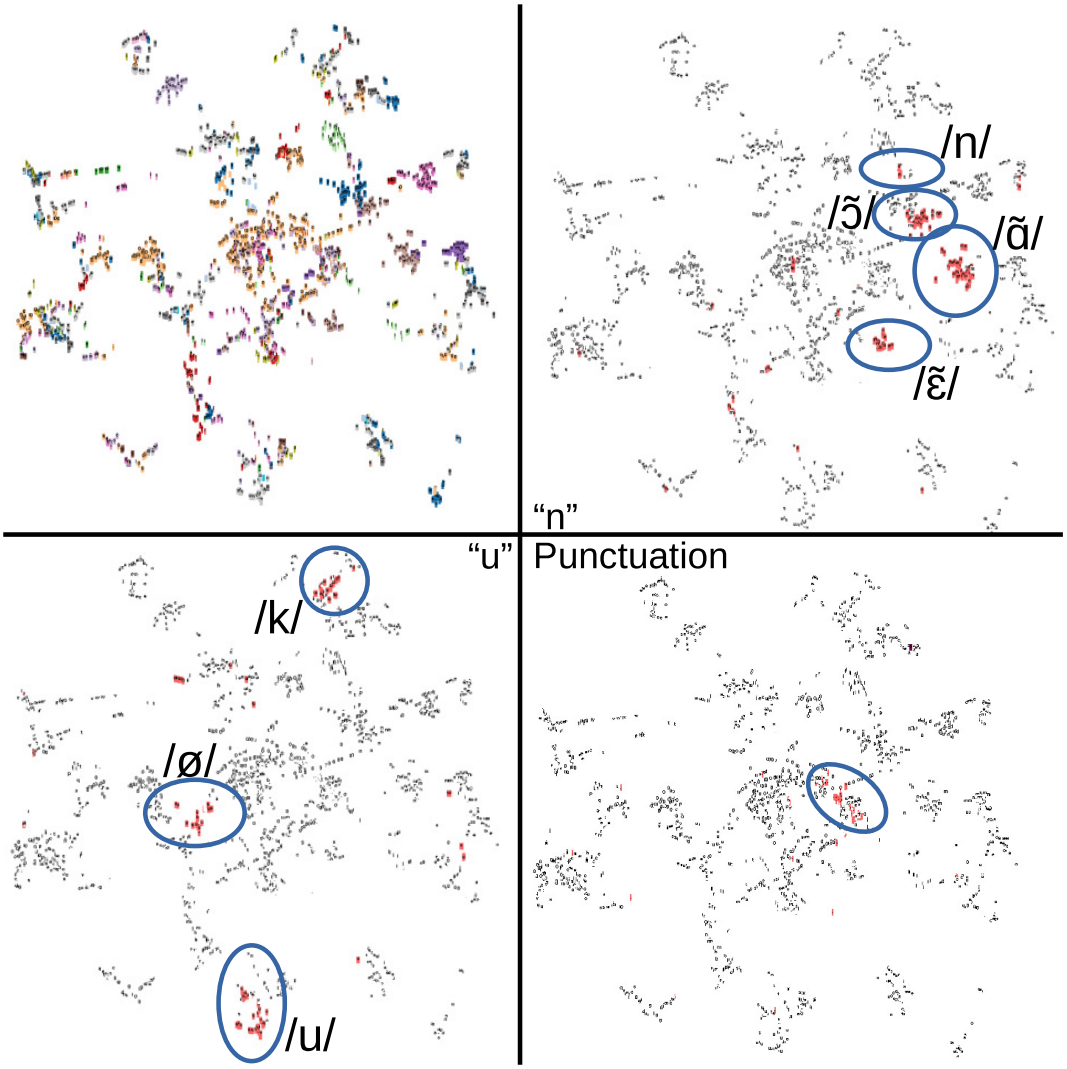


Figure 4.7 – t-SNE visualization of the contextual character embeddings. Top-left : All embeddings are highlighted. Top-right : All "n" characters are highlighted. Bottom-left : All "u" characters are highlighted. Bottom-right : All punctuation characters are highlighted.

characters. Each of those three groups also correspond to a different phoneme. For example, a first group contains the embedding of characters written in contexts such as "*emb*", "*end*" or "*ant*", which correspond to the phoneme /ɑ̃/. Similarly, the other two groups correspond to the phoneme /ɛ̃/ and /ɔ̃/. The same observation can be done for vowel characters. Looking at the lower left part of the figure, at least three groups of the "u" character can be found. They correspond respectively to the phonemes /u/, /ø/ and /k/. Finally, the lower right part of the figure shows the behavior of punctuation contextual characters embeddings. All punctuation, such as ".", "*?*" and "*,*" can be found in the same group of embeddings. Some non-punctuation characters can also be found in the group. Most of those are actually mute characters such as "*s*" or "*t*" ending a word without being pronounced, or the space character. Interestingly, embeddings corresponding to the space character can be found in most group of embeddings. However, their presence in a given group cannot always be explained by a similar textual context.

Overall, according to the observations made thanks to the t-SNE visualization, it seems that the structure of the contextual character embeddings follows a structure where they can be separated into group depending on their corresponding phoneme. This is not surprising since the Tacotron architecture simulates N-grams extraction thanks to the CBHG module (see Section 2.3). This allows to capture information at a higher level than the character one. It is interesting to notice that the Tacotron model seems to learn word pronunciation by discovering the phoneme structure of speech and the phonology of French, without any linguistic expertise.

4.4.2 Further Use of the Character Embeddings

The observation of the contextual character embedding space led us to hypothesize that the Tacotron model learns an internal representation tied to phonemes. In order to measure this property objectively, we propose to build a phonetizer working with contextual character embeddings or raw characters as input and compare the performance of the two systems.

To do so, as in Section 3.4.5 we aim to build the least complex model possible. The model (drawn on Figure 4.8) is a neural network built using the ESPNET (Watanabe et al. 2018) framework. It takes either a sequence of contextual character embeddings or one-hot encoded raw characters as input, and outputs a sequence of one-hot encoded phonemes. The model consists of a single LSTM layer with 64 units and a feed-forward softmax layer. Since the input and output sequence do not match temporally, the network

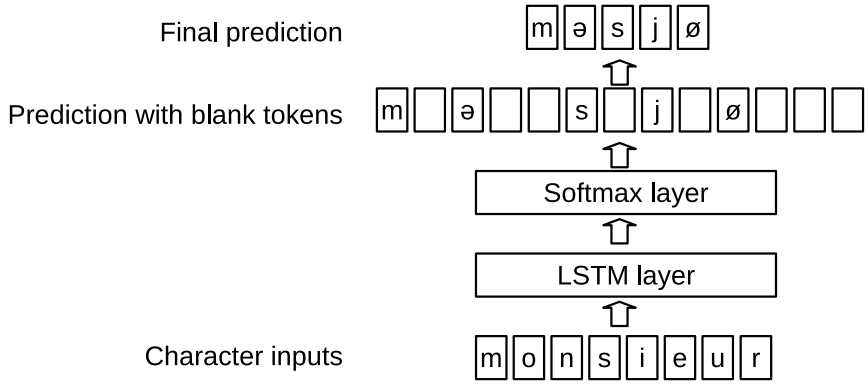


Figure 4.8 – Architecture of the model used to perform grapheme-to-phoneme conversion

	Raw Characters	Contextual Character embeddings
All training data	0.19	0.13
All validation data	0.30	0.16

Table 4.4 – Phoneme Error Rate of all models after G2P training. The lower the better

needs a mechanism to align the two sequences. To keep the model as simple as possible, rather than use an attention mechanism, the model makes use of the Connection Temporal Classification (CTC) loss (Graves et al. 2006). This method allows to predict label directly from unsegmented sequences by introducing a blank token between each of the output tokens. This token can be repeated to indicate that the preceding non-blank token is repeating. In the case of this experiment, the blank token indicates that the current character is joined to the previous one to form a phoneme.

The models are trained over 20 epochs on the entire training set. Then, the quality of the models is measured by comparing the predicted and reference sequence using PER over the test set. A second training is done for both models with a limited number of training data. The second training is done on the 400 sentences of the validation set, which were not seen during the training of the contextual character embedding space. The results of the experiment are reported on Table 4.4.2.

When training on all data available, the phonetizer working from contextual character embeddings slightly outperforms the one working on raw characters. The difference between the two mean values is statistically significant. This suggests that the contextual character embeddings are indeed a closer representation to phoneme than raw characters are. When trained on the validation set, the performance of the G2P working on embeddings slightly worsen. This suggests that either the model benefits from a larger amount of data or

that embeddings extracted from unseen data during training are slightly less related to phonemes than those extracted from the training data. The precision of the model trained on raw characters on the validation set worsened significantly when trained on the validation set. Comparing the two models trained on the validation set shows that contextual character embeddings are indeed more related to phoneme than characters are.

4.5 Conclusion

By design, the use of end-to-end models for speech synthesis allows to reduce the needed amount of linguistic expertise. Indeed, rather than to rely on hand-crafted linguistic features, the Tacotron model learns meaningful representations of phoneme sequences. Furthermore, the model can be trained on raw characters rather than on a phoneme sequence. Then, the Tacotron model replaces both the linguistic analysis and part of the audio generation module of the traditional pipeline. In the case of a well-curated French dataset, we have shown that using characters instead of phoneme does not lead to a lower quality or any increase of pronunciation errors. Such end-to-end models fulfill half of the requirements for universal speech synthesis that we had defined in the introduction of this thesis. Toward universal speech synthesis, the next chapter focuses on adding easily controllable variety in the synthesized output. Namely, speaker and accents variety.

Chapter 5

Adding Variety to End-to-End Speech Synthesis

Speech can be defined with three main components. The first is *content*, the message conveyed through speech. The second is *speaker voice*, the voice with which the sample is synthesized. The third is *expressiveness*, the different manners in which the message is conveyed. Expressiveness can be due to many factors, such as regional accent, emotion, etc. In the rest of this chapter, those factors of expressiveness and speaker voice will be called speech style factors. In order for speech synthesis to become universal, a single TTS system needs to be able to produce a wide variety of speech. The TTS systems built in Chapter 3 and 4 are already able to express a wide variety of content. However, they are limited to a single speaker voice and are not expressive. Toward universal speech synthesis, this chapter aims to enable the control of multiple style factors by conditioning an end-to-end model on embeddings modeling those style factors explicitly. More precisely, this work focuses on speaker voice and accents.

Following the works presented in Section 2.5, we consider that the control mechanism for speech style factors in a universal system should use reference samples. The resulting system is drawn in Figure 5.1. The use of reference samples allows the system to analyze a real example of speech to derive automatically a representation of the factor wished for. Then, this representation can be used to condition the production of a synthesized speech sample. In this chapter, we study the possibility of a system taking multiple style factors into account. Furthermore, we assume that each of the factors under consideration are labeled on the training set.

To allow control over each style factor independently, they must be modeled separately and must be disentangled from one another. To do so, we propose to train a Tacotron model conditioned on one embedding per factor considered. The style factor embeddings are obtained from encoders taking reference samples as input. In order to disentangle each factor, the encoders must be trained at the same time, in an adversarial manner. To only

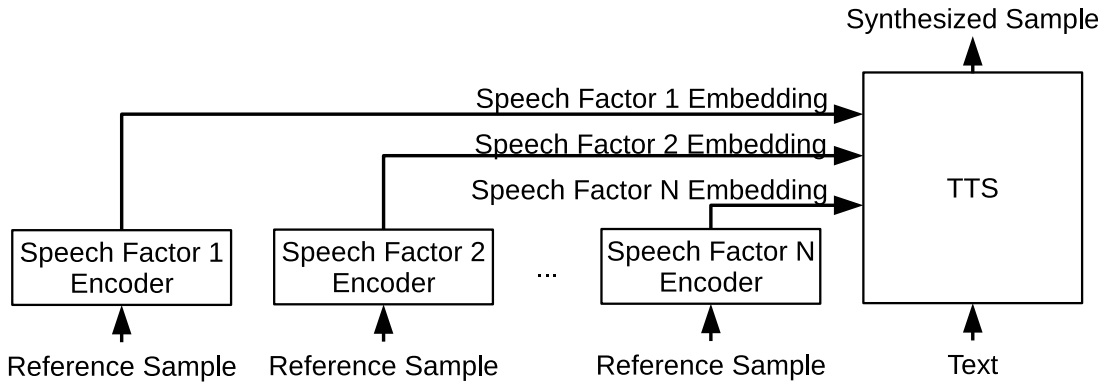


Figure 5.1 – Drawing of a TTS system modeling explicitly multiple speech factors.

study the impact of style factor embeddings, the Tacotron models used in this chapter take phoneme sequences as input rather than characters.

In particular, in this chapter, we focus on the case of a system able to explicitly model two style factors : the speaker's voice and regional accents. Section 5.1 will begin by building a system able to synthesize multiple speaker voices. Then, Section 5.2 attempts to extend the system to synthesize multiple speaker voices and accents.

5.1 Multi-speaker Speech Synthesis

Before attempting to model multiple style factors, this section focuses on a single factor and aims to tackle speaker voices. The goal of this section is to build a multi-speaker speech synthesis system. By conditioning a Tacotron model on speaker embeddings, the system should be able to synthesize speech with the different training voices. Furthermore, if the model learns to generalize well over the speaker embedding space, it should also be able to synthesize voices that were unseen during training with no additional training required. The remainder of this section introduces the system built for that purpose, the data used to train it, as well as an analysis of its performance. More precisely, Section 5.1.1 presents an extension of the model used in Chapter 4 to allow multi-speaker synthesis. Section 5.1.2 introduces the dataset used to train the model. Since the dataset originally contains a large amount of data unfit for training a speech synthesis system, Section 5.1.3 explores multiple data augmentation schemes to improve the quality of the attention model. Finally, Section 5.1.4 evaluates objectively how well the model reproduces the voices of speakers.

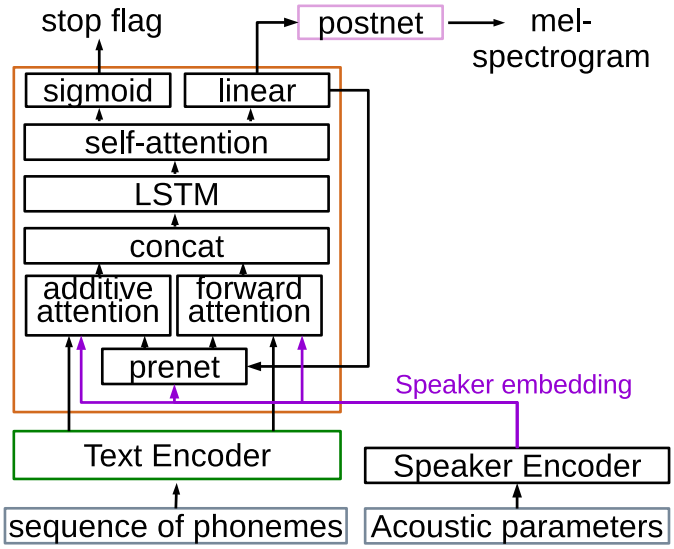


Figure 5.2 – Architecture of the tacotron model conditioned on speaker embeddings. The speaker encoder extracts LDE embeddings and is pre-trained before the tacotron model.

5.1.1 System

The overall architecture (drawn in Figure 5.2) of the model used in this section is an extension of the one used in Chapter 4, with the addition of a speaker embedding derived from a pre-trained speaker encoder. The speaker voice should only affect the spectrogram generation, not the analysis of the text pronunciation. Thus, only the decoder is conditioned on the speaker embedding. More precisely, as in (Cooper, Lai, Yasuda, Fang, et al. 2020), the embedding is concatenated with the inputs of the additive attention, forward attention, and Prenet. All hyper-parameters are the same as in Chapter 4.

The speaker encoder extracts LDE embeddings from one sequence of acoustic features. Similarly to x-vectors, LDE embeddings are extracted by applying convolution layers to the sequence of acoustic features at the frame level. Then, those features are pooled to obtain a single vector representing the whole utterance. This vector is either called x-vector or LDE embedding depending on the method of pooling. Finally, a classifier is trained to predict the identity of the speaker in the input utterance from the pooled vector. This allows the embedding to capture the speaker's voice characteristics. The mechanism for x-vectors and LDE embeddings is drawn on Figure 5.3.

In the case of x-vectors, the pooling method is usually statistical. The mean and standard deviation of the frame-level embeddings are computed. An x-vector is the concatenation of those values. For LDE embeddings (Cai, Jinkun Chen, and M. Li 2018), the

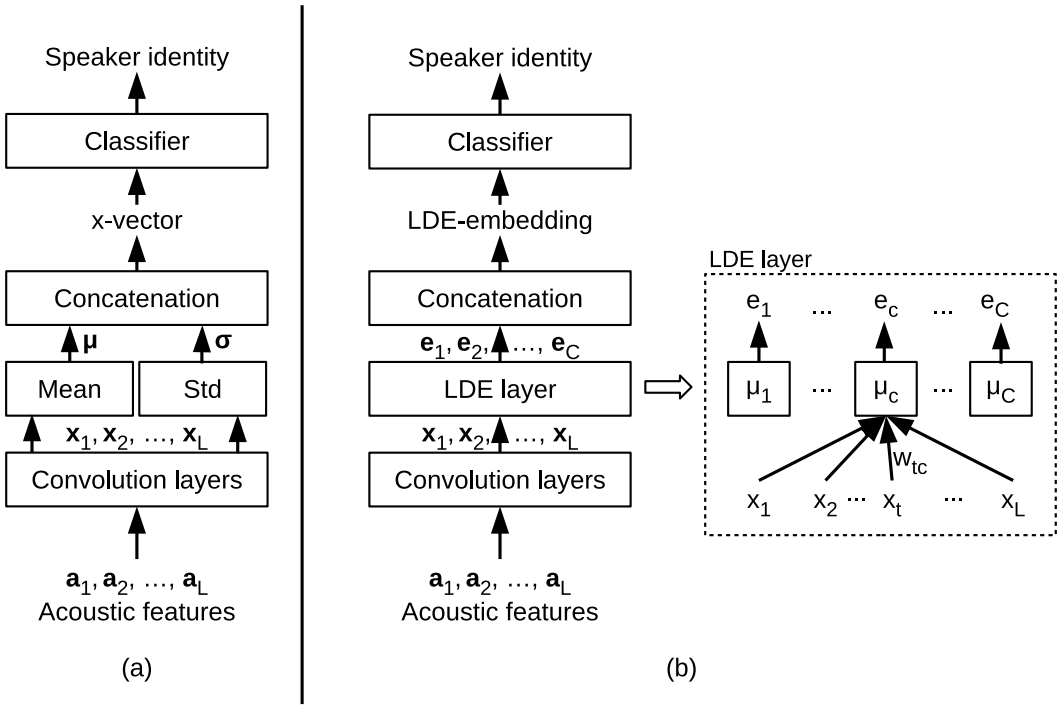


Figure 5.3 – Architecture of the neural networks used to extract x-vectors (a) and LDE embeddings (b).

pooling relies on a mechanism similar to attention, aligning the frame embeddings x to a sequence of C vectors μ_c learned during training. Each vector μ_c represents unspecified components of speaker characteristics. A set of weights is computed to express how much a frame-level embedding x_t is related to component μ_c:

$$w_{tc} = \frac{exp(-\|x_t - \mu_c\|^2)}{\sum_{m=1}^{C} exp(-\|x_t - \mu_m\|^2)} \tag{5.1}$$

Then, the L frame-level embeddings are pooled by each components to obtain C e_c vectors:

$$e_c = \frac{\sum_{t=1}^{L} w_{tc}(x_t - \mu_c)}{\sum_{t=1}^{L} w_{tc}} \tag{5.2}$$

Finally, the LDE embeddings are the concatenation of the pooled values : $E = (e_1, .., e_C)$ In the rest of this section, the speaker encoder extracts LDE-embeddings. x-vectors will be used for evaluation in Section 5.1.4.

5.1.2 Dataset and Experimental Setup

A large number of speakers is needed to obtain a system able to adapt to new voices without additional training. However, there is currently no open speech dataset available for French with a large number of speakers. Thus, we used a private dataset recording 206 speakers, initially designed for automatic speech recognition. The audio quality is reasonably good since the recordings were done in a silent environment with a high-end microphone. However, elements of the corpus might be ill-suited for high-quality speech synthesis. For example, the speakers used for the recordings are not professional voice actors. This led to some utterances containing disfluences such as missing or repeated words. Some sentences also contain mispronounced words. Also, depending on speakers, the audio corresponding to plosive phonemes might be saturated due to the lack of experience of the speakers. Another reason of why the dataset might not be the most adapted for speech synthesis is its textual content. A large part of the original dataset is made of voice commands recorded during the development of speech interactive telephones. The texts in the dataset can be sorted in five different categories:

1. short basic commands for hands-free telephone control
2. sequences of numbers and phone symbols
3. name of streets, cities, rivers and other geographical french landmarks
4. sentences selected from newspaper for phoneme covering
5. single words selected for phoneme covering

All but the fourth category might be undesirable for speech synthesis since they do not match the target domain which is continuous natural speech. The utterances in the first category are very short (one to five words) and are not always grammatically correct. The second, third and fifth categories are not actual sentences. Under the assumption that those categories might hinder the quality of a speech synthesis model, we limited ourselves to use only the fourth category : complete sentences, grammatically correct and similar to natural speech.

As such the main corpus used in the rest of this section is composed of 100 sentences for each of the 206 speakers. This results in a total of 20600 utterances for around 10 hours of speech. Due to the way in which the dataset was originally designed, the newspaper sentences for each speaker are randomly chosen among a set limited to 2282 sentences. As a result, each newspaper sentence is read by multiple speakers.

As in the previous chapter, the audio samples are downsampled to 24 kHz. Beginning and trailing silences are trimmed. 80 dimensional mel-spectrograms are extracted. The text sentences are normalized to extend simple abbreviations and numbers into their full form. Then, the corresponding phoneme sequences are predicted using the phonetizer of Espeak before encoding each of the phonemes using one-hot encoding.

Finally, the main corpus is divided in train/test/development subsets. The test set is composed of all the samples read by 28 speakers. Similarly, the validation set is composed of all the samples read by another group of 28 speakers. The training set is composed of the remaining samples. Following this split, the voices of the test speakers are entirely unseen by the system during training. However, the text of the samples read by the test speakers are also present in the train and development set. This is due to the fact that each sentence in the dataset is read by multiple speakers.

The speaker encoder used in this section is the same as in (Cooper, Lai, Yasuda, Fang, et al. 2020). We use the pre-trained weights given by the authors without any fine-tuning. Due to the architecture and training of the speaker encoder, the resulting embeddings are supposed to be extracted independently from the textual content of the reference samples. As such, they should also be language-agnostic. We extract the LDE embeddings corresponding to each audio samples in the dataset. Then, for each speaker, an average speaker embedding vector is computed from 10 randomly chosen embedding vectors of that speaker. This allows to alleviate features that might be related to a single sample, not shared with the other, and thus not representative of the speaker as a whole.

The weights of the conditioned Tacotron model are initialized using warm-starting (Baylor et al. 2017). It consists in initializing parts of the weights of a neural network using saved weights from a previous run of that neural network. Since the model used in this section is an extension the phoneme model trained in Chapter 4, it contains additional weights. All weights existing in both models are initialized with the values from the speaker dependent model, the remaining are initialized randomly. The model is trained during 10 days and gets poor results in an informal listening test due to a high rate of attention alignment errors. Section 5.1.3 aims to reduce the rate of alignment error, before measuring how well the system is able to reproduce a speaker's voice in Section 5.1.4.

5.1.3 Attention Alignment Errors

The first multi-speaker Tacotron model trained leads to poor performance because a high number of samples are synthesized with alignment errors. An example of such sample

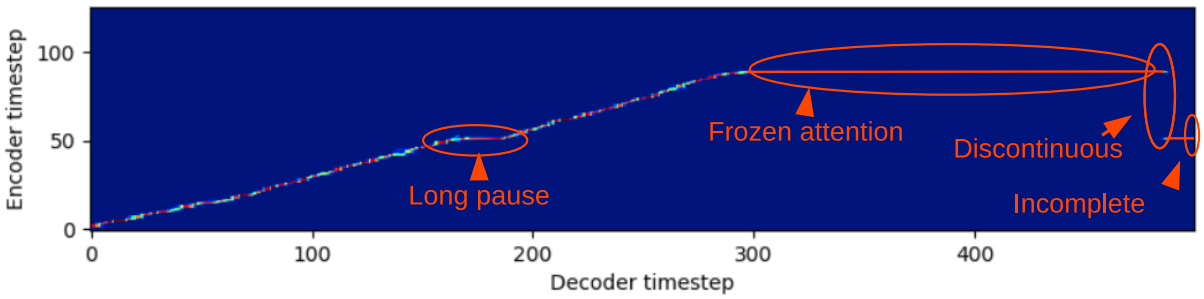


Figure 5.4 – Exemple of alignment errors

can be found in Figure 5.4. Three types of alignment errors are possible :

- Discontinuity: The most salient attention weights are discontinuous. It can lead to skipping and repeating words.
- Incompleteness: The most salient attention weights for the last frames of the mel spectrograms do not match the last character of sentences. It leads to sentences being partially read.
- Frozen attention: The most salient attention weights for consecutive mel-spectrogram frames are matched with the same character. This leads to long lingering sounds or pauses.

The aim of this section is to measure automatically the quality of the attention model for alignment. We compare multiple data augmentation schemes to decrease the number of alignment errors.

Attention Alignement Error Detection

For speech synthesis, the attention model computes an alignment between a text and the corresponding mel-spectrogram. When reading, the reader is focused on a couple of characters at most and focuses its attention on each of the characters sequentially, without suddenly going back to characters previously read or skipping words. Similarly, the alignment in the Tacotron model should be continuous. The errors can be detected automatically by looking at the attention weights. Using the same definitions as in Section 2.2, for each timestep i of the output sequence, the result of the alignment model is a weigthed sum over each timestep j of the input sequence $\boldsymbol{x}$:

$$\boldsymbol{c_i} = \sum_{j=1}^{N} \alpha_{ij} \boldsymbol{x_j}. \tag{5.3}$$

The most salient weight α_i for timestep i of the output sequence is found for timestep $\hat{j}$ of the input sequence :

$$\begin{aligned} \alpha_i = \alpha_{i\hat{j}} = max_j(\alpha_{ij}), \text{ with} \\ \hat{j} = argmax_j(\alpha_{ij}). \end{aligned} \tag{5.4}$$

For ease of notation, we note n the function mapping an input timestep i and the output timestep $\hat{j}$ for which the most salient weight is found :

$$n : i \mapsto argmax_j(\alpha_{ij}) \tag{5.5}$$

Then, the progress in alignment between two following input timesteps is computed as :

$$|n(i) - n(i+1)| \tag{5.6}$$

According to the continuity hypothesis, this progress should happen gradually. Thus, we can detect swift changes using a threshold T:

$$|n(i) - n(i+1)| < T \tag{5.7}$$

If equation 5.7 is true for each timestep i of the ouput sequence, then the alignment is continuous.

Furthermore, a text must be read completely, to its end. Similarly, the alignment process should match the last frames of a mel-spectrogram to the last characters of the sentence synthesized. For an mel-spectrogram of length M, the most salient weight is found for the character at index $n(M)$. For the alignment to be complete, that character should be close to the end of the sentence. According to the completeness property, For a sentence of length N :

$$n(M) \simeq N \tag{5.8}$$

This can be detected using the same threshold as previously :

$$N - n(M) < T \tag{5.9}$$

If equation 5.9 is true, then the alignment is complete.

Due to the difficulty of distinguishing between long pauses and frozen attention, this work does not attempt to detect those type of errors. However, we can observe that samples with frozen attention are also often incomplete. Finally, the attention alignment

error rate is defined as the number of samples where at least one type of alignment error has been automatically detected, over the number of samples considered.

Data Augmentation Schemes

The high amount of attention alignment errors might be due to the low amount of data available after the pruning of the original dataset. In order to lower the alignment error rate, we explore two data augmentation schemes. First, we can observe that the amount of original sentences in our dataset is quite low. In order to increase the amount of textual data, we consider adding all the utterances previously left out of the dataset (category 1, 2, 3 and 5). This augmentation scheme is referred to as *text augmentation*.

Alternatively, in order to increase the amount of audio data available without adding new sentences from a different dataset, we can derive new samples from the original ones by manipulating audio. Hence, from every original samples, we generated two new ones by speeding up or slowing down the signal by a factor 1.1 without pitch correction. The absence of pitch correction leads to the voice in the slowed down (respectively sped up) signal being perceived as deeper (respectively sharper). The result is an audio signal with natural sounding speech for two new virtual speakers. This augmentation scheme is referred to as *speaker augmentation*.

We consider each augmentation scheme independently, and their combination. In the case of the combination, we first augment the corpus with textual sentences. Then, we apply the speaker augmentation scheme to all textual data, including the ones added by the textual augmentation. When speaker augmentation is used, the mean speakers embeddings are also extracted, in the same way as for real speakers.

Comparison of the Data Augmentation Schemes Using Alignment Error Rate

The multi-speaker Tacotron model is trained on the main dataset and on each of the augmented corpus for 10 days. The quality of the alignments is then measured automatically. Since there are no unseen texts in the test set, we measure the alignment error rate on the sentences from the test set defined in Chapter 4. It is composed of 100 French sentences taken from books, parliament sessions, or semantically unpredictable sentences. The result for the models trained with different data augmentation scheme are reported in Table 5.1.

The rate of alignment errors for the model trained on the main corpus is of 22.9%. More than a fifth of the sentences are synthesized with at least one alignment error.

No augmentation	Text augmentation	Speaker augmentation	Both augmentations
22.9%	10.8%	13.8%	15.8%

Table 5.1 – Automatic alignment error rate measured for different augmentation schemes. The lower the better

While not all errors caught by the automatic detection correspond to total failure of the synthesizing process, most impact the quality of the synthesized speech.

When augmenting the dataset with every textual data available in the corpus, the rate of alignment errors falls down to 10.8%. While more data is usually considered better when training neural networks, it is surprising that utterances with only one word or sequences of numbers did not hinder the alignment process. A possible explanation is that the addition of textual data, in particular the name of French landmarks, increases the amount of vocabulary seen during training. As such, the generalization property of the network improves.

When augmenting the dataset with virtual speakers, we augment the amount of audio data without adding more vocabulary. The result is an improvement of the alignment error rate to 13.8%. This suggests that the main corpus does not contain enough data for optimal training. Interestingly, the difference between the error rates of the text and speaker augmentation scheme is statistically significant. This suggests that, for alignment errors, increasing the vocabulary is more effective than recording a higher number of speakers with the same text.

Finally, augmenting the dataset with text and speaker augmentation (in that order) also leads to an improvement. This suggests again that the main corpus is not big enough to train a convincing alignment model. The difference between this system and the ones trained with a single data augmentation is statistically significant. This result is surprising since the expected outcome would be that the combination of both augmentation schemes should outperform each scheme taken independently. Despite training this model with different sets of hyper-parameters, the result has always been the same. Our hypothesis is that, since the augmented textual data are repeated due to the speaker augmentation, there is a slight over-fitting on the augmented data (sequence of numbers, etc.).

Overall, this experiment suggests that a certain amount of data is needed to train a convincing attention mechanism for multi-speaker speech synthesis. Interestingly, it seems that increasing the size of the vocabulary, even through single-word utterances, increases the quality of the alignment process. However, there appears to be a trade-off on the

amount of those poorly constructed sentences. Finally, this experiment leads us to think that a corpus designed for multi-speaker speech synthesis should strive to record different texts for each speaker, in order to facilitate the training of the attention mechanism. In the rest of this work, we focus on the model trained with text augmentation as it gave the best alignment performances.

5.1.4 Speaker Similarity

The main goal of multi-speaker synthesis is to reproduce the voice of multiple speakers with high fidelity. This section aims to objectively evaluate the similarity between the voices of target speakers and the voices actually synthesized by the multi-speaker system. We follow the methodology described in (Cooper, Lai, Yasuda, Fang, et al. 2020) and assimilate speaker similarity to the cosine similarity between speaker embeddings. We first present the evaluation methodology before showing it allows to distinguish between natural speakers. Then, we apply this methodology to measure the speaker similarity between natural and synthesized speakers. Finally we investigate different speaker adaptation method to improve the performance of the voice cloning.

Methodology

The methodology applied to measure speaker similarity is drawn on Figure 5.5. For a given speaker, 10 samples are randomly selected to extract x-vectors. Then, a mean speaker embedding is computed as the average of those 10 x-vectors. We synthesize the texts corresponding to those samples using the multi-speaker Tacotron model conditioned on the average LDE embedding of that speaker. X-vectors are also computed from those 10 synthesized samples and averaged to obtain a mean x-vector corresponding to the synthesized voice. Finally, the speaker embedding derived from original data $\boldsymbol{e}_{orig}$ and from synthesized data $\boldsymbol{e}_{synth}$ can then be compared using the cosine similarity defined as:

$$cos(\boldsymbol{e}_{orig}, \boldsymbol{e}_{synth}) = \frac{\boldsymbol{e}_{orig}.\boldsymbol{e}_{synth}}{\|\boldsymbol{e}_{orig}\|\|\boldsymbol{e}_{synth}\|}. \tag{5.10}$$

The cosine similarity is 0 when the two vectors are orthogonal which means they are dissimilar. The similarity is 1 when the two vectors are co-linear which means they are entirely similar. It is important to note that if $e \in \mathbb{R}^N$, the cosine similarity can also be negative. Then, we interpret the range $[-1, 0]$ as dissimilarity.

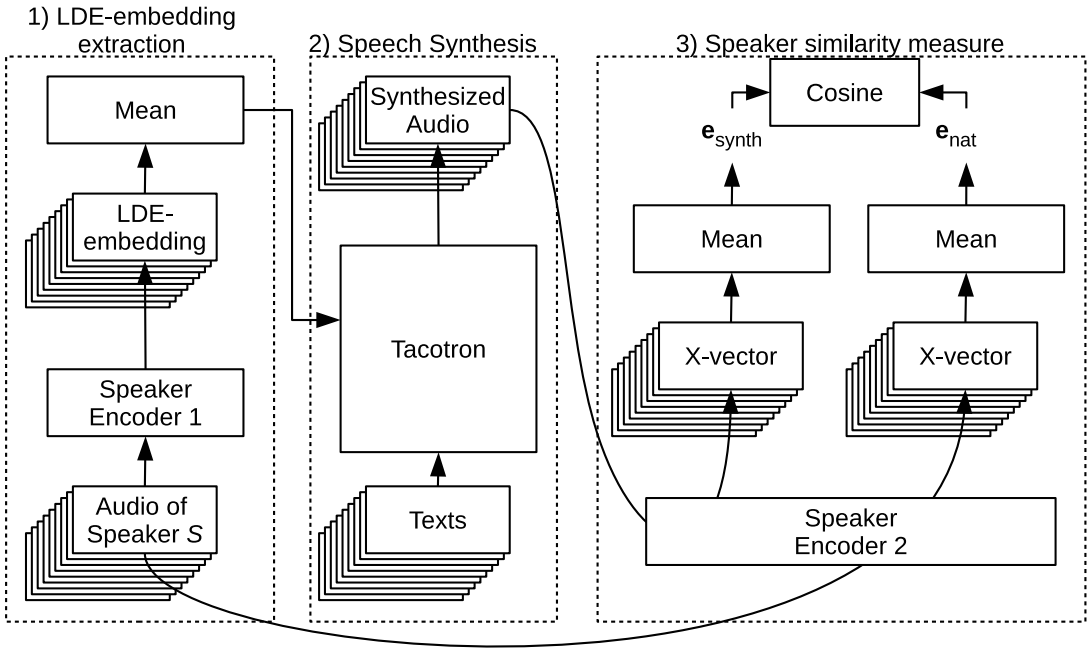


Figure 5.5 – Drawing of the methodology applied to objectively measure the speaker similarity between natural and synthesized speech.

	Woman 1	Woman 2	Man 1	Man 2
Woman 1	1	0.608	0.182	0.447
Woman 2	0.608	1	0.110	0.579
Man 1	0.182	0.110	1	0.523
Man 2	0.447	0.579	0.523	1

Table 5.2 – Comparison of 4 speakers using the cosine similarity on average x-vectors extracted from natural speech.

Similarity Between Natural Speakers

We start by asserting whether this measure is well-founded or not. To do so, the cosine similarity is used to compare two female and two male speakers with x-vectors derived from natural samples. Those speaker are chosen randomly from the test set. The resulting similarity measures can be found on Figure 5.2.

Overall, the cosine similarity between the mean x-vectors of each of the speaker is around or below 0.6. This suggests that this measure can indeed be used to distinguish between speakers. However, it is hard to interpret the actual scale of the values measured. For example, the similarity measured between speakers of the same sex is of 0.5 or 0.6. But the similarity between "Man 2" and both women is also around 0.5. Thus, this measure

Woman 1	Woman 2	Man 1	Man 2
0.883	0.820	0.810	0.775

Table 5.3 – Cosine similarity measure between average x-vectors embeddings computed from natural and synthesized speech

can help to distinguish between speakers but might not be completely interpretable as a proxy of perceptual similarity between two voices.

Similarity Between Natural and Synthesized Speakers

We can then measure the ability of the model to produce new voices by measuring the speaker similarity for unseen speakers between natural and synthetic speech. The multi-speaker Tacotron model can be adapted to new speakers by feeding it with their LDE embedding. Again, the speaker similarity is measured as the cosine similarity between x-vectors computed from natural and synthesized speech. The unseen speakers are the ones selected for the previous experiment. The samples used to compute the average synthesized x-vectors are randomly selected among samples that did not present alignment errors. The results of the experiment are reported on Table 5.3.

All cosine similarities are between 0.78 and 0.88. Thus, all of those values are greater than the threshold of 0.6 previously found. This suggests that for a speaker $s_{nat,i}$, the corresponding cloned voice $s_{synth,i}$ in synthesized samples is closer to $s_{nat,i}$ than to a different speaker $s_{nat,j}$. However, the values on the lower end suggest that perceptual speaker similarity might be quite low. Indeed, an informal listening test led to observe that listeners were not able to confidently say that the voices of the synthesized and natural samples were the same, nor the contrary.

Comparison of Different Adaptation Methods

The result of the previous experiment shows that the model perform poor speaker adaptation when it consists only in feeding the speaker embedding of an unseen speaker. However, the speaker similarity might be improved by fine-tuning the model on samples from the target speaker. This then raises the question of how much data is needed for the fine-tuning. Furthermore, if fine-tuning is necessary, are multi-speaker models beneficial in any way compared to a speaker dependant model trained to synthesize the voice of a single speaker ? In order to answer these questions, another experiment has consisted in fine-tuning the phone-based single-speaker Tacotron model obtained in Chapter 4 and

SSM	MSM without fine-tuning	MSM fine-tuning 10 sentences	MSM fine-tuning 20 sentences	MSM fine-tuning 50 sentences
0.948	0.803	0.947	0.914	0.906

Table 5.4 – Cosine similarity measured between average x-vectors extracted from natural and synthesized speech. Comparison a Single Speaker Model (SSM), a Multi-Speaker Model (MSM) adapted without fine-tuning, a multi-speaker model fine-tuned on a varying amount of data.

our Tacotron multi-speaker one. Both models, fine-tuned on a single speaker, can then be compared to evaluate these two speaker adaptation schemes.

Since the data of a single speaker from the corpus used in this section might not be enough, we instead use a small subset from the Synpaflex corpus (Sini et al. 2018). It contains recordings of audio-books read by an amateur female speaker. The audio quality is similar to that of the corpus used to train the multi-speaker Tacotron model. We consider a total of 943 sentences, 743 are used for the train set, 100 are kept the validation and test set each. An LDE-embedding vector is extracted for that new speaker and the multi-speaker Tacotron model is fine-tuned on a varying amount of data from the new speaker training set for 20 epochs. The single-speaker model is also fine-tuned for 20 epochs on the entire new speaker training set. The results of the experiment are reported on Table 5.4.

As expected, the speaker similarity between the samples synthesized from the single-speaker model and natural speech is quite high. This suggests that if a TTS system only needs to synthesize speech with a single voice, if a low amount of data (couple of hours) is available for that given voice, fine-tuning a high quality model is a good alternative to training from scratch. The cosine similarity for the multi-speaker model without any fine-tuning is of 0.8. This value is similar to those found in the previous experiment. This confirms the behavior of the model on unseen speakers. Then, for all amounts of data used for fune-tuning the multi-speaker model, the cosine similarity is higher than 0.9. In particular, the model fine-tuned on only 10 sentences has similar value as the single-speaker model. This suggests that fine-tuning a multi-speaker model, even on a really small amount of data (couple of minutes), is a good alternative to fine-tuning a speaker dependent model. This method has the benefit of being faster to train. Furthermore, it seems that the amount of data needed to compute an average speaker embedding is enough to perform the fine-tuning.

A universal TTS system must be able to synthesize a wide variety of speech. In this section, we have studied the use of speaker embeddings to condition a sequence-to-sequence model predicting mel-spectrograms to allow multi-speaker speech synthesis. However, a universal system should be able to add more variety with little or no effort. In theory, the use of speaker embeddings allows to synthesize speech with the voice of unseen speakers by using the corresponding audio as a reference. However, in practice, this type of speaker adaptation is not optimal. The adaptation can be improved by fine-tuning the multi-speaker model on the data used to compute the speaker embeddings. While this process is far from perfect, it serves as a first step toward universal speech synthesis. To get closer to that goal, the next section attempts to model two speech factors at the same time: speaker voice and accent.

5.2 Multi-speaker Multi-accent Speech Synthesis

After having modeled a first speech factor, we attempt to model two speech factors concurrently. The goal of this section is to build a multi-speaker multi-accent speech synthesis system. By conditioning a Tacotron model on speaker and accent embeddings, the system should be able to synthesize speech with different voices and accents. Furthermore, if the two speech factors are modeled independently, in such a way that their representations are disentangled, each speech factor should be controllable separately. In the case of a multi-speaker multi-accent model, this would allow to synthesize speech with the voice of a chosen speaker and an accent different than the speaker's original one. This is called *accent transfer*.

The remainder of this section introduces the multi-speaker multi-accent speech synthesis system, the data used to train it, as well as an analysis of its performance. More precisely, Section 5.2.1 introduces the architecture of a Tacotron model predicting mel-sprectrograms from text and conditioned on speaker and accent embeddings. Then, Section 5.2.2 presents the dataset used to train the model. Section 5.2.3 measures how much the speaker voice and accent speech factors were disentangled in the speaker and accent embeddings. Finally, Section 5.2.4 measures how well the speaker voices and accents are reproduced, as well as the quality of the accent transfer.

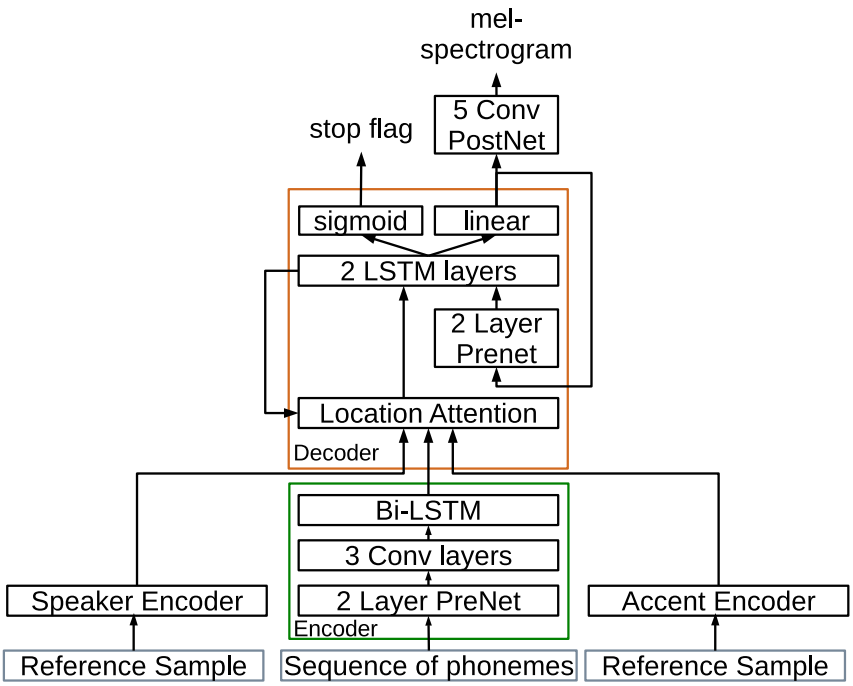


Figure 5.6 – Architecture of the multi-speaker multi-accent model.

5.2.1 Model

In this section again, the model used to predict mel-spectrograms from texts is based on the Tacotron architecture. It is drawn on Figure 5.6. We follow the modifications introduced by (Shen et al. 2018) to simplify the model. The CBHG module in the text encoder is replaced by 3 usual 1-dimensional convolution layer, followed by a bi-directional LSTM. This serves the same purpose as the CBHG module: to extract contextual phoneme embeddings. Furthermore, in order to explicitly model speaker and accent-related information, we condition the decoder on speaker and accent embeddings. The two embeddings are concatenated to the ouput of the text encoder before being fed to the attention model of the decoder. The speaker and accent embeddings are computed by a speaker encoder and an accent encoder respectively. Those encoders are trained at the same time as the Tacotron model in order to compute a single vector summarizing an entire reference mel-spectrogram while capturing speaker or accent characteristics.

In order to independently control the voice being synthesized and the accent with which the text is read, we aim to learn disentangled representation of those two speech factors. The aim is similar to (Hsu et al. 2019) where the authors attempt to disentangle the speaker and noise information. As such, we use the same architecture for our encoders.

It is drawn on Figure 5.7. The frame sequence $\boldsymbol{s}_{ref}$ of a mel-spectrogram is fed to a stack of two convolutional layers each with 512 filters of width 3 and stride 1. This allows to extract contextual features for each frame of the mel-spectrogram. Then, those features are fed to a stack of two bi-LSTM layers with 256 units in each temporal direction in order to model time dependencies both forward and backward in time. Finally, the sequence of frame-level embeddings is pooled using statistics. The embedding vector is the concatenation of the mean and standard deviation of the features over time. In the case of the speaker encoder, this process is similar to the training and extraction of x-vectors. The main difference is in the layers used in the encoder. The speaker and accent encoders are trained at the same time as the Tacotron model

MB In order to constrain the speaker and accent encoder to learn characteristics corresponding to speaker voice and accent respectively, a classifier is added to each encoder during training. The architecture of these classifiers is drawn on Figure 5.7. A reference mel-spectrogram $\boldsymbol{s}_{ref}$ is encoded by the speaker and accent encoder to obtain the embeddings $\boldsymbol{e}_{spk}$ and $\boldsymbol{e}_{acc}$. Then, the first classifier predicts the ID $\boldsymbol{id}_{spk}$ of the speaker that uttered the sample, from $\boldsymbol{e}_{spk}$. The ID is a one-hot vector of size N_{spk}, the number of speakers in the training set. Similarly, the second classifier predicts the ID $\boldsymbol{id}_{acc}$ of the accent spoken from the embedding $\boldsymbol{e}_{acc}$, where $\boldsymbol{id}_{acc}$ is a one-hot vector of size N_{acc}, the number of accents in the training set. The presence of those two classifiers during training results in a multi-task model predicting a mel-spectrogram, a speaker ID and an accent ID. The model is trained to minimize the sum of three losses corresponding to each task of the model:

- The reconstruction loss L_{rec}. It is computed as the mean square error between the predicted and ground-truth mel-spectrograms $\boldsymbol{s}_{pred}$ and $\boldsymbol{s}_{true}$,
- The speaker classification loss L_{spk}. It is computed as the categorical cross-entropy between the predicted and ground-truth speaker IDs $\boldsymbol{id}_{spk,pred}$ and $\boldsymbol{id}_{spk,true}$,
- The accent classification loss L_{acc}. It is computed as the categorical cross-entropy between the predicted and ground-truth speaker IDs $\boldsymbol{id}_{acc,pred}$ and $\boldsymbol{id}_{acc,true}$.

Then, the overall loss L_{tot} of the model is defined as:

$$L_{tot} = L_{rec}(\boldsymbol{s}_{true}, \boldsymbol{s}_{ref}) + L_{spk}(\boldsymbol{id}_{spk,true}, \boldsymbol{id}_{spk,ref}) + L_{acc}(\boldsymbol{id}_{acc,true}, \boldsymbol{id}_{acc,ref}). \quad (5.11)$$

The classifiers are composed of two feed-forward layers. The first is a linear projection of dimension 256, the second is a softmax prediction layer. Since this model does not

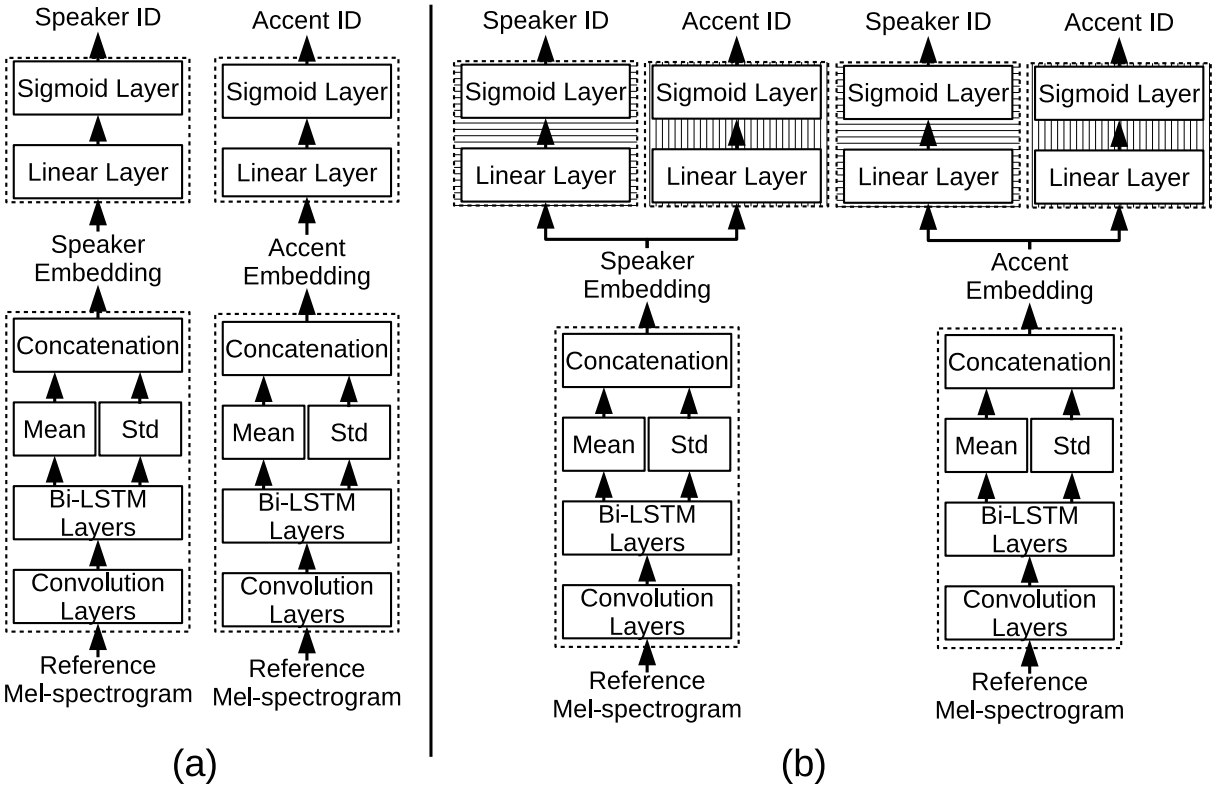


Figure 5.7 – Architectures of the encoder and classifier used to model speech factors. Model MB follows the architecture drawn in (a). Model MP and MPS follow the architecture drawn in (b). Additionally, the weights of the classifiers with the same hatching are shared for model MPS.

have any mechanism to disentangle the two speech factors, it will serve as a baseline in the following experiments. Thus, the Tacotron model and the associated encoders will be called Model Baseline (MB).

MP Learned in such a manner, the speaker and accent embeddings are allowed to capture speaker and accent characteristics respectively. However, there is no assurance that each embedding encodes only characteristics corresponding to the speech factor it models. In order to attempt to disentangle speaker and accent characteristics, we add a second classifier after each encoder. The resulting model is drawn on Figure 5.7. The first additional classifier predicts the ID of the speaker $\boldsymbol{id}_{spk}$ from the accent embedding $\boldsymbol{e}_{acc}$. The second additional classifier predicts the ID of the accent $\boldsymbol{id}_{acc}$ from the speaker embedding $\boldsymbol{e}_{spk}$. Those classifiers add two additional tasks to the model. They are taken

into account through the following loss functions:

— The speaker classification loss L_{spk}^{acc}. It is computed as the categorical cross-entropy between the ground-truth speaker ID $\boldsymbol{id}_{spk,true}$ and the speaker ID predicted from the accent embedding $\boldsymbol{id}_{spk,pred}^{acc}$,

— The accent classification loss L_{acc}^{spk}. It is computed as the categorical cross-entropy between the ground-truth accent ID $\boldsymbol{id}_{acc,true}$ and the accent ID predicted from the speaker embedding $\boldsymbol{id}_{acc,pred}^{spk}$.

In order to penalize the speaker (respectively accent) encoder if it encodes accent (respectively speaker) characteristics, these two losses are added to the overall loss L_{tot} as penalty terms:

$$L_{tot2} = L_{tot} + \frac{1}{\lambda + L_{acc}^{spk}(\boldsymbol{id}_{acc,true}, \boldsymbol{id}_{acc,pred}^{spk})} + \frac{1}{\lambda + L_{spk}^{acc}(\boldsymbol{id}_{spk,true}, \boldsymbol{id}_{spk,pred}^{acc})} \tag{5.12}$$

where λ is a constant value added to avoid a division by zero. Indeed, the categorical cross entropy takes values greater or equal to zero. Since the model attempts to disentangle the two speech factors using penalty losses, it will be called Model Penalty (MP).

MPS Finally, we consider a second way to disentangle the two speech factors. We attempt to train model MP while sharing the weights of the classifiers predicting the same type of IDs. The speaker classifier predicting from speaker embeddings and the speaker classifer predicting from accent embeddings share the same weights. We apply the same process for the accent classifiers. This model will be referred to as Model Penalty Shared (MPS).

5.2.2 Dataset and Experimental Setup

To the best of our knowledge, there is currently no open dataset containing a high number of French speakers with different accents, with a sufficient total amount of data to train end-to-end systems. In this section we use VCTK (Veaux, Yamagishi, MacDonald, et al. 2016), an anglophone public dataset, to train our models and evaluate them. It contains the text and recordings of anglophone speakers from different regions of the world, leading to different accents. More precisely, the dataset contains recordings of 110 speakers, coming from one of 11 regions, and with around 400 utterance per speaker. The accent represented are from the United Kingdom (English, Welsh, Scottish, Northern Irish), Ireland, USA, Canada, South Africa, Australia, New Zealand and India. Additional information such as

age and region of living are present but were not used in our experiments. The recordings are of high quality, but the speakers are not professional voice actors. The number of speakers per accent is not balanced. English and American are the most covered with 33 and 22 speakers respectively. The least represented ones are New Zealand, and Welsh, with a unique speaker each. The dataset's gender distribution is overall well balanced with 61 women and 47 men. All speakers first read a short excerpt called the Rainbow Passage (Fairbanks 1940). Then, different sentences from a set of newspaper excerpts are selected for each speaker, in a way that maximizes the phonetic coverage.

Before extracting acoustic features from the audio, the samples were down-sampled to 16 kHz and the beginning and trailing silences were trimmed. Then, 80-dimensional mel-spectrograms are extracted. Phoneme sequences are extracted from the texts using the phonetizer from Espeak.

The corpus is divided in train/development/test subsets. The test set is composed of all samples from 11 speakers, 1 per accent, with 6 women and 5 men. Due to the unbalanced nature of the dataset, the Welsh and New Zealand accents are only present in the test set. The validation set is composed of a random 10% of the remaining data. Finally, the training set is composed of the remaining samples. The test set allows to evaluate the model on unseen speakers and unseen accents. The validation set allows to evaluate the model on seen speakers, but unseen reference waveforms and texts.

The three models presented in Section 5.2.1 are trained for 10 days. During training, for each training sample, a mel-spectrogram from the speaker to synthesize is chosen randomly and used as the input of both the speaker and accent encoders.

5.2.3 Experiment: Disentangling Speaker and Accent Embeddings

Three models conditioned on speaker and accent embeddings have been trained to allow multi-speaker multi-accent speech synthesis. In order to control both speech factors independently, the two embedding spaces need to be disentangled. The aim of this section is to evaluate whether the methods proposed to train the speaker and accent embeddings were successful in learning disentangled representations. The evaluation is performed objectively, first by measuring the accuracy of the classifiers trained on speech factor embeddings, second by using KNN classification on the speech factor embeddings spaces themselves.

classification task	embedding space	MB	MP	MPS
speaker	speaker	91 %	99 %	99 %
accent	accent	89 %	99 %	99 %
speaker	accent	/	0.30 %	2.6 %
accent	speaker	/	3.8 %	11 %

Table 5.5 – Comparison of the 3 tacotron models according to the accuracy of the underlying classifiers. In the first two rows, the higher the better. In the last two row, the lower the better.

Accuracy of the Classifiers

We first investigate the disentanglement by evaluating the precision of the classifiers trained at the same time as the embedding encoders and Tacotron model. Since the test set only contains unseen speakers, the accuracy of the classifiers cannot be measured on the test set. Thus, we compute the accuracy of the classifiers associated to each model using the development set. The measured values are reported in Table 5.5

The first model trained, MB, does not attempt to disentangle the speaker and accent dimensions. As such, it can be used as a baseline to evaluate the effectiveness of the other proposed methods. The model possesses only one classifier per type of variety. The baseline is an accuracy of 90% for a classifier predicting the identity of a speech style according to the corresponding speech style embedding.

The second model, MP, adds a second classifier to each embedding in order to act as a penalty during the learning. Compared to the baseline, this method allows for an improved precision of the classifiers working on the corresponding speech style embedding. The accuracy of the speaker classifier working on accent embeddings is close to 0%, while that of the accent classifier working on speaker embeddings is 3.8%. This means that the speaker classifier is not able to predict the speaker identity from an accent embeddings (similarly for the accent classifier). This hints that the regional accent embeddings might not contain speaker characteristics. Thus, the speaker and accent speech factors might have been untangled.

The third model, MPS, adds an additional constraint to the training process by constraining the speaker classifier working on accent embeddings to share its weights with the speaker classifier working from speaker embeddings (similarly for the accent classifier). When predicting from the corresponding speech factor, the accuracy of the classifiers is similar to that of MP. Interestingly, the precision of the speaker classifier predicting from accent embeddings is significantly higher for model MPS than model MP. Idem for the

accent classifier predicting from speaker embeddings. This means that if the untangling process was successful, the speech and accent factors are probably more correlated in the embedding spaces of model MPS than those of model MP.

We proposed to disentangle the speaker and accent embeddings by learning penalty classifiers attempting to predict the accent or speaker ID from the embeddings. The measures show that the penalty classifiers were successfully trained, in such a way that they are not able to predict the speaker ID from the accent embedding (and vice-versa). This suggests the underlying embedding might not encode anything but the characteristics of the speech factor it is supposed to encode. However, those measures are not based on the embeddings themselves but on classifiers trained to fail. As such, it is important to validate those observations by investigating the embeddings directly.

Investigation of the Speech Factor Embeddings

Measuring the accuracy of the classifiers showed hints that the disentangling process might have been successful since classifiers predicting from embeddings of the opposite speech factor have bad accuracy. However, those classifiers might have learned a strategy to disregard the embeddings they are supposed to predict from. In this experiment, we evaluate the disentanglement process via a KNN classification in the embedding spaces, similarly to section 3.4.6. The reasoning is that if the speaker and accent embeddings are truly disentangled, then the distribution of the embeddings in the accent embedding space should be independent of the speaker identity. As such, the nearest neighbors of an acoustic embedding e uttered by a speaker s with accent a should be embeddings uttered by speakers with accent a but who are not speaker s. We project utterances from the development set in either the acoustic or speaker embedding space defined by the utterances in the training set. For each utterance in the development set, we search the 20-nearest neighbors among the training embeddings. The mean accuracy of this 20-NN classification is reported in Table 5.6

The first row corresponds to the accuracy of the KNN speaker classification in the speaker embedding space. Both models MP and MPS provide an improvement of the accuracy over the baseline. This suggests that the speaker encoder of those models capture speaker characteristics more efficiently than the baseline. This is coherent with the observation made in the previous experiment. Similarly, the accent classification performed in the accent embedding space shows that both models also improve the baseline results.

The third row of Table 5.6 evaluates the accent classification in the speaker embedding

classification task	embedding space	MB	MP	MPS
speaker	speaker	57 %	72 %	69 %
accent	accent	81 %	98 %	90 %
speaker	accent	38 %	56 %	45 %
accent	speaker	68 %	83 %	74 %

Table 5.6 – Comparison of the 3 tacotron models according to the accuracy of a KNN classification in the embedding spaces. In the first two rows, the higher the better. In the last two row, the lower the better.

space. All models have an accuracy between 80 and 90%. Such a high value means that almost all neighbors of an embedding in the speaker embedding space are utterances spoken with the same accent. This is not surprising since the speaker embedding space should gather utterances spoken by similar speakers together. In particular, utterances from the same speaker should be close to each other. Since all utterances from a given speaker share a same accent, it is not surprising to have such a high accuracy.

Finally, the fourth row measures the speaker classification in the accent embedding space. Both the MP and MPS model have an accuracy between 50 and 60%. This means that half the neighbours of a given utterance in the accent embedding space were spoken by the same speakers. While this number is high, it is significantly lower than the 84 or 86% accuracy of the speaker classification in the speaker embedding space. This means that the accent embedding space encodes less speaker characteristics than the speaker embedding space. Surprisingly, the accuracy of the baseline is only of 34%. This suggests that the schemes introduced to disentangle the speaker and accent dimensions might have hindered the process instead of helping it.

Overall, we investigated the disentanglement of the speaker and accent embeddings with two different approaches. The first approach is based on measuring the accuracy of the classifiers trained from the embeddings. It hints that the disentanglement process is successful. However, it is doubtful whether a measure based on a classifier trained to fail is reliable. The second approach is based on performing a nearest neighbor search in the embedding space. It confirms that the accent embedding space encodes less speaker characteristics than the speaker embedding space. However, it also suggests that the schemes introduced are hurtful for the disentangling process. It is also doubtful whether this measure is the best proxy for evaluating disentanglement. Independently of whether the speaker and accent dimensions are still entangled, we next proceed to evaluate the

ability of the systems to synthesize speech for different speakers and accents.

5.2.4 Evaluation of Voice Cloning and Accent Transfer

Since the multi-speaker multi-accent models are conditioned on two separate embeddings, two use-cases are possible :

— The embedded accent matches the accent of the embedded speaker. This case is similar to Section 5.1 with an additional explicit modeling of the accent. This is voice adaptation, or voice cloning.

— The embedded accent does not match the accent of the embedded speaker. In this case, we try to suppress the original accent of the embedded speaker and to apply a different one. This is accent transfer.

The aim of this section is to evaluate objectively the capacity of the model to perform both use cases, starting with voice cloning. To do so, as in Section 5.1.4, we compare average speaker embeddings derived from natural speech to those derived from synthesized speech using cosine similarity. Additionally, we perform the same comparison on average accent embeddings to investigate whether the system is able to correctly synthesize a speaker's accent during voice cloning, then to check if the accent was efficiently reproduced during accent transfer. But first, the objective measure is used to compare the speaker and accent embeddings of natural samples as a baseline.

Speaker and Accent Similarities Between Natural Samples

In order to avoid biases, the average embeddings must be extracted from the same neural network. Furthermore, this neural network must not be models MB, MP, or MPS since they are under evaluation. We train two separate neural networks following the encoder/classifier architecture presented in Section 5.2.1 over the training set, without any Tacotron model or any disentangling scheme. The first DNN is trained to extract speaker embeddings and will be referred to as E_{spk}, the second is trained to extract accent embeddings and will be referred to as E_{acc}. We begin by asserting the usefulness of those encoders by comparing average embeddings derived from natural speech. For each speaker i in the development set, the corresponding natural samples are fed to E_{spk} to derive speaker embeddings which are then averaged to obtain a single average speaker embedding per speaker $\boldsymbol{s}_i^{nat}$. Similarly, for each accent j in the development set, we compute the average accent embedding $\boldsymbol{a}_j^{nat}$ by averaging the embeddings extracted

	p334	p310	p244	p251	p271	p230
p334	1	0.01	-0.18	0.31	0.28	-0.10
p310	0.01	1	0.20	0.19	-0.04	0.04
p244	-0.18	0.20	1	-0.06	-0.17	0.31
p251	0.31	0.19	-0.06	1	0.48	-0.11
p271	0.28	-0.04	-0.17	0.48	1	0.19
p230	-0.10	0.04	0.31	-0.11	0.19	1

Table 5.7 – Subset of the cosine similarity computed between pair of speakers from natural speech. The letter "p" followed by an integer is the ID of the speaker in VCTK.

	American	English	Indian	Scottish	South African	Irish	Canadian	Northern Irish	Australian
American	1	-0.01	-0.29	0.06	0.05	0.06	**0.30**	0.10	-0.35
English	-0.01	1	0.09	0.14	-0.05	0.28	0.13	0.08	-0.33
Indian	-0.29	0.09	1	-0.11	-0.17	-0.16	-0.25	-0.24	-0.02
Scottish	0.06	0.14	-0.11	1	-0.36	-0.03	-0.12	-0.17	-0.13
South African	0.05	-0.05	-0.17	-0.36	1	0.11	-0.23	0.08	-0.56
Irish	0.06	0.28	-0.16	-0.03	0.11	1	-0.17	-0.18	-0.25
Canadian	**0.30**	0.13	-0.25	-0.12	-0.23	-0.17	1	0.05	0.28
Northern Irish	0.10	0.08	-0.24	-0.17	0.08	-0.18	0.05	1	-0.16
Australian	-0.35	-0.33	-0.02	-0.13	-0.56	-0.25	0.28	-0.16	1

Table 5.8 – Cosine similarity between each pair of average accent embedding. The highest similarity is highlighted in bold

from E_{acc} over all speakers speaking that accent. We compute the cosine similarity between each pair of speakers and each pair of accents in the development set.

Table 5.7 presents a small subset of the cosine similarity computed on speaker embeddings between pairs of speaker. Since the cosine similarity is a symmetric measure, so is the similarity matrix, with 1 on the diagonal. Overall, the smallest cosine similarity measured is -0.62, the highest is 0.78, for a mean value of 0.04. This quick verification shows that comparing average speaker embeddings extracted from E_{spk} using the cosine similarity allows to distinguish between two speakers. In particular, a cosine similarity higher than 0.78 should be a good indicator that the average embeddings are derived from samples uttered by the same speaker.

Similarly, Table 5.8 presents the cosine similarity computed between the average accent embeddings extracted with E_{acc} for all pairs of accent. The highest measured value is 0.30

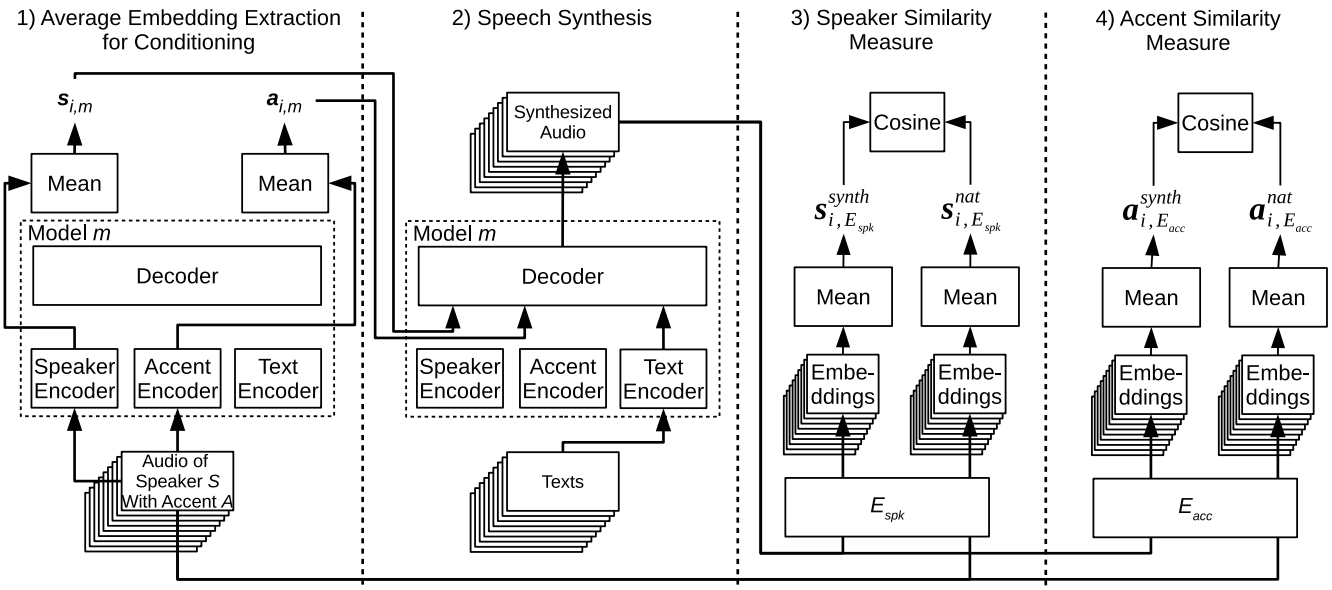


Figure 5.8 – Drawing of the methodology applied to objectively measure the speaker and accent similarity between natural and synthesized speech.

between the Canadian and American accents. The lowest value measured is -0.56 between the South African and Australian accents. Overall, the mean measured value is -0.07. It seems that computing the cosine similarity between average accent embedding extracted from E_{acc} allows to distinguish between accents. Furthermore, a cosine similarity higher than 0.30 can be interpreted as the embeddings being extracted from samples uttered by speakers with a same accent.

Evaluation of the Voice Cloning

By extracting average embeddings from natural samples using the encoders E_{spk} and E_{acc}, and by computing cosine similarity, one can decide objectively if natural audio samples were uttered by the same speakers, and if they were uttered by speakers with the same regional accent. A similar method (drawn on Figure 5.8) is applied between embeddings extracted from natural and synthesized speech to evaluate whether the voice of a speaker was cloned correctly and if her accent has been reproduced faithfully. We first use this method to evaluate our models' ability to clone voices. We select 9 speakers from the development set randomly so as to get one speaker per accent.

For a given model m among MB, MP and MPS, for each speaker i, we extract an average accent $a_{i,m}$ and speaker embedding vector $s_{i,m}$ from 10 natural speech samples

		p226	p234	p293	p245	p248	p299	p316	p323	p374
MB	speaker similarity	0.87	0.91	0.79	0.86	0.95	0.95	0.95	**0.88**	0.96
	accent similarity	0.83	**0.96**	0.69	**0.89**	0.70	0.82	0.85	**0.73**	0.94
MP	speaker similarity	**0.98**	**0.97**	**0.84**	**0.95**	0.95	0.92	**0.97**	0.83	**0.97**
	accent similarity	0.89	0.95	**0.83**	0.88	0.80	0.86	0.88	0.48	**0.94**
MPS	speaker similarity	0.96	0.92	0.81	0.91	**0.96**	**0.96**	0.97	0.74	0.97
	accent similarity	**0.91**	0.91	0.77	0.76	**0.84**	**0.87**	**0.89**	0.59	0.94

Table 5.9 – For each model, cosine similarity between embeddings (accent or speaker) extracted from synthesized and natural speech for 9 speakers. For each speaker, the best similarity for each speech factor is highlighted.

using the encoders of m. We synthesize 10 samples per speaker by conditioning the model m on $a_{i,m}$ and $s_{i,m}$. Finally, the average speaker and accent embeddings are extracted from the synthesized samples using E_{spk} and E_{acc} to obtain $s^{synth}_{i,E_{spk}}$ and $a^{synth}_{i,E_{acc}}$. The voice cloning process can then be evaluated by measuring the cosine similarity between $s^{synth}_{i,E_{spk}}$ and $s^{nat}_{i,E_{spk}}$. Similarly for accents by comparing $a^{synth}_{i,E_{acc}}$ and $a^{nat}_{i,E_{acc}}$.

The first row of Table 5.9 reports the cosine similarity computed between average speaker embeddings extracted from natural speech or from speech synthesized by model MB. All 9 speakers were measured with a similarity higher than 0.78. This value was the threshold found previously, under which two average speaker embeddings can be said to correspond to the different speakers. This suggests the voice of those 9 speakers has been cloned successfully by model MB. The average speaker similarity measured for model MB is 0.89. Similarly, all speaker similarities measured for model MP and MPS are also above the threshold. This means our 3 models are able to reproduce a speaker's voice when used in a speaker adaptation manner. The average speaker similarity measured for model MP and MPS is of 0.93 and 0.91. On average, no model outperformed the others significantly. Thus, all models perform speaker adaptation with a similar level of quality.

The second row of Table 5.9 reports the cosine similarity computed between average accent embeddings extracted from natural speech or extracted from samples synthesized by model MB. Similarly to the observation for speaker similarity, the values for all 9 speakers are higher than the threshold of 0.30 for accent embeddings. This suggests that the accent of those 9 speakers was reproduced faithfully. The same observation can be done for model MP and MPS. The average accent similarity measured for model MB, MP and MPS is of 0.82, 0.83 and 0.83 respectively. Again, no model outperforms the others significantly.

	English	Scottish	Northern Irish	Irish	Indian	American	Canadian	South African	Australian
MB	0.17	0.28	0.04	0.18	-0.01	**0.30**	0.03	0.15	-0.16
MP	**0.39**	**0.37**	0.13	**0.31**	0.23	0.23	0.09	-0.01	-0.01
MPS	**0.39**	**0.37**	0.13	**0.31**	0.23	0.23	0.09	-0.01	-0.01

Table 5.10 – Mean cosine similarity between accent embeddings extracted from natural speech and from speech synthesized with accent transfer. Values higher than 0.3 are highlighted

Overall, this experiment shows that the three models trained can be used for voice cloning. Then, the voice of each speaker and their accent will be reproduced faithfully according to this objective measure. Interestingly, no model outperforms the others significantly. This implies that the scheme implemented to disentangle the speaker and accent dimensions do not alter the behavior of the model for voice adaptation.

Evaluation of the Accent Transfer

Finally, the second use-case of a multi-speaker multi-accent TTS system is accent transfer : the reference used for the accent embedding does not match the accent of the speaker in the reference used for the speaker embedding. To evaluate the behavior of the models on this use case, we select the same 9 speakers as previously. The evaluation method is similar to the one used for the first use-case. For a given model m among MB, MP and MPS, for each speaker i, we extract an average accent $a_{i,m}$ and speaker embedding vector $s_{i,m}$ from 10 natural speech samples using the encoders of m. To test the accent transfer of each accent to each speaker, we synthesize 10 samples for each permutation of $a_{i,m}$ and $s_{i,m}$ to condition the model m. Finally, the average accent embeddings are extracted from the synthesized samples using E_{acc} to obtain $a^{synth}_{i,E_{acc}}$. We can then evaluate the accent transfer process by measuring the cosine similarity between $a^{synth}_{i,E_{acc}}$ and $a^{nat}_{i,E_{acc}}$. Table 5.10 reports the mean similarity measured per accent for each of our 3 models.

The highest mean similarity value measured for model MB is 0.3 when transferring to the American accent. For model MP and MPS, the mean similarities for English, Scottish and Irish are also slightly higher than 0.3. The baseline highest value of similarity between two different natural accent was 0.3. This means that almost all synthesized accents are objectively dissimilar from their corresponding natural accents. In the case of the three accents that were measured with a value above 0.3, this means the synthesized accents are somewhat similar to the corresponding natural target accent. However, since the value is

so low, it is doubtful whether they are perceptually similar.

Overall, the accent transfer did not lead to faithfully reproducing a given target accent. This failure might be due to multiple reasons. First, the disentangling process of the voice and accent speech factor was incomplete. This could hinder the accent transfer since the average accent embedding would encode information other than accent characteristics. Second, the conditioning of the Tacotron model on the accent embeddings might also be incomplete. It might be useful to also condition the text encoder on accent embeddings for example.

5.3 Chapter conclusion

In this chapter, we attempted to build a TTS system which explicitly models multiple speech components through the use of speech factor embeddings. The aim is to obtain a system where each speech component is controllable independently. In particular, we investigated the case of multi-speaker multi-accent TTS. As a first step, we have built a multi-speaker Tacotron model conditioned on speaker embeddings. Different data augmentation schemes were investigated in order to stabilize the attention alignment process. The results suggest that data usually unsuited for the training of high speech synthesis system might still be useful to learn more stable alignment models. We then objectively evaluated the voice adaptation ability of the multi-speaker model. The results suggest that in order to reproduce voices faithfully, the model needs to be fine-tuned. Interestingly, it seems that fine-tuning over 10 sentences is enough to be comparable to a system solely designed to synthesize the same voice. Then, we presented an extension of the model to perform multi-speaker multi-accent speech synthesis. This extension consists in conditioning the Tacotron model on both speaker and accent embeddings. We showed that despite implementing training schemes to disentangle the speaker and accent dimensions, both speech factors are still entangled. This failure might be due to the fact that speaker identity and accents are intrinsically correlated. Indeed, the accent of a speaker can be deduced from a speaker's identity since each speaker speaks with only one accent. However, a speaker's identity cannot be deduced solely from its accent, so disentangling speaker and accent should at least be possible in the accent embedding space. We evaluated objectively the capacity of the model to perform speaker adaptation and accent transfer. The results suggest that the model is able to faithfully reproduce a given target speaker and her/his accent. Finally, we attempted to adapt the original accent of a speaker to a

target one. However, the results showed that the synthesized accent shares characteristics of the target accent while still being recognized as a different one.

Chapter 6

Conclusion and Perspectives

Conclusion

The aim of this thesis was to investigate the use of embeddings toward universal speech synthesis. We identified two aspects of universality where embeddings derived from neural networks can be used in speech synthesis. First, it should be easy to understand how something universal work, and it should be easy to modify it. However, building a TTS system usually implies an heavy pipeline where one needs to have both linguistic and acoustic expertise. We investigated ways to use embeddings to lower the amount of linguistic expertise needed. The second aspect is that something universal must be usable in every situations. Then, a universal TTS must be able to synthesize speech corresponding to a wide variety of speaking style. We investigated ways of using embedding to explicitly model those speech components, thus enabling a TTS system to synthesize speech with different styles.

This thesis began at the same time as disruptive advances were made to speech synthesis with the introduction of end-to-end systems, replacing unit selection and statistical parametric systems as the state of the art. In that context, we began by showing that embeddings can be used to lower the amount of linguistic expertise needed to build a unit selection TTS. More precisely, the traditional target cost of a unit selection system is defined using linguistic expertise. We showed that phone embeddings extracted from an acoustic neural network can be used to defined the target cost automatically as the Euclidean distance in the phone embedding space. We made three contributions to the unit selection guided by neural networks method. First, we compared the hybrid and traditional approach. This showed that, despite lowering the amount of expertise needed, the automatically defined cost leads to synthesized speech of similar quality as the expert cost. Second, we compared the use of acoustic models with different quality to define the target cost. This proved that even bad acoustic models can lead to good unit selection speech synthesis. Third, we investigated the notion of quality for phone embeddings. We

showed that embeddings leading to good speech synthesis encode both linguistic and acoustic aspects of speech. However, pieces of linguistic expertise remain in the unit selection pipeline. It is still needed to predict the pronunciation of a text and to define the linguistic features used as input of the acoustic model.

We then showed that embeddings can further lower, if not completely remove, the need for linguistic expertise thanks to end-to-end systems. Such systems predict an acoustic description of speech directly from the text or from its phonetization. If using phoneme as inputs, the expertise is lowered by removing the need to explicitly define linguistic features for acoustic prediction. However, some expertise still remains in the phonetization process. We showed that, in the case of a well-curated French dataset, the end-to-end model can be learned directly from characters with no loss in speech quality, and without additional pronunciation errors. This work made two main contributions. First, it applied end-to-end speech synthesis to the French language, which is rarely done. Second, we investigated the character embedding space to understand how it relates to pronunciation. We found that the end-to-end model learns contextual character embeddings that encode phoneme characteristics without phonetic label supervision.

Finally, we investigated the use of embeddings to explicitly model speech components. We began by conditioning an end-to-end model on speaker embeddings to allow multi-speaker synthesis. Two main contributions were made. First, the particular dataset used to train the models led us to investigate different data augmentation schemes in order to stabilize the learning of the attention alignment module. We showed that all types of textual data are actually useful to learn a stable alignment model. Our second contribution is the objective evaluation of the model's ability to adapt to new speakers. We showed that while fine-tuning is still needed to adapt to new voices, the fine-tuning can be done on a small amount of audio data. We then extended the system to model a second type of speech style factor: accents. We made two main contributions. First, in order to control both factors independently, we proposed different training scheme to disentangle the speaker and accent embeddings. However, it is inconclusive whether the disentanglement was successful because the identity of a speaker (his/her voice) can lead to guess the accent. The second contribution is the evaluation of the voice cloning and accent transfer processes. Through objective measures, we showed that the system is able to correctly synthesize the true accent of speakers. However, when trying to replace a speaker's accent to a target one, our approaches fail to actually transfer the accent, due to the lack of disentanglement.

Perspectives

Following this thesis, perspectives are opened in different directions. First, the notion of embeddings has shown, as in many other domains, to outperform several aspects of a TTS system by removing linguistic expertise. Hence, one can wonder if this could not be applied for other aspects which were not tackled in this PhD work. Second, our difficulties to mix multiple voices and accents shows that the problem is still not solved, calling for potential solutions. Finally, once TTS will have reached the universality objective, it seems to me that the next step to generalize it to multi-modal synthesis in order to progress toward better artificial avatars. The following details these three lines of future research.

Use of embeddings to remove expertise in other problems

Even if unit selection is almost not used anymore (or at least not a major research focus), further work could be conducted to generalize the hybrid approach proposed in Chapter 3. While we have explored the use of embeddings to define the target cost of a unit selection system, the selection process also makes use of a join cost to predict how well two units can be defined together. Embeddings could then also be used to define the join cost in addition to the target cost. Indeed, the join cost is usually defined as a comparison between acoustic descriptors. As such, if phone embeddings successfully encode acoustic characteristics, then the join cost can probably be defined as a distance between phone embeddings, thus removing acoustic expertise.

Then, we have shown that character embeddings learnt by Tacotron systems relate to phonemes. This means that phonetizers could be trained based on those embeddings rather than on characters directly, possibly improving the performance of phonetization systems. Furthermore, this method could allow to learn phonetizers for languages and dialects where no phonetically annotated data exist (pronunciation dictionaries or phonetically labelled speech). Indeed, if a corpus containing textual and audio data does exist, a Tacotron model can be learned. Then, a phonetizer could be derived from the underlying character embeddings.

Integrating more speech style factors

The experiments on disentangling the speaker's voice and regional accent were inconclusive. This raises the question of whether the approach might not still be relevant for

other speech style factors, and what might be solutions to enable a generic disentangling approach.

On the one hand, I think that the proposed architecture might still be applied to other speech style factors with less intrinsic correlation. For example, the same method could be applied to disentangle speaker's voice and emotion in acted speech. A successful model could then be used to generate audiobooks, where speech styles are different between acted dialogues and neutral narration.

On the other hand, further work is necessary to make accent transfer successful. In this work, we tried to condition the synthesis based on accent embeddings. A strong assumption was that these accent embeddings should not contain any other information than the one related to accent (especially, nothing about the speaker's voice). However, this does not relate enough to the use case of accent transfer. At training time, the models should probably also to be told that the generated speech signal must sound like the target accent. To do this, an additional classifier could be appended to the model in order to analyze the generated mel-spectrogram, for instance the same as the one used in the accent encoding step. When considering several factors, e.g. the accent and the speaker's voice, this architecture should furthermore guarantee that (1) the accent from the speaker sample is not taken into account, and (2) the voice from the accent sample is neither. Again, this can be expressed as constraints on back-end classifiers. This whole approach is actually close to what can be found in the textual style tranfer litterature, where style-tranferred sentences are a posteriori analysed to check that the target style is met whereas the meaning is still the same. While this seems to be a promising perspective, the next question would probably be how to enable this approach based on a larger number of different factors (voice, accent, age, spontaneousness, emotion, etc.). Indeed, issues would be raised during training because of the exponential number of possible combinations of factors.

Multi-modal Synthesis

More personally, in the long term, I believe that speech synthesis should be included in the larger context of multi-modality. While related work already shows promising results on audio-visual speech synthesis, facial or gesture animation, and body motion, the question of full-body multi-modal speech synthesis does not seem to be addressed yet. This synthesis would for instance enable an avatar to speak while acting mental or physical characteristics. One could imagine an avatar who would talk while running and

being in an happy mood. To be realistic, he/she would be hopping while running, and the speech could be halted at times due to shortage of breath. The avatar generation may be conditioned on various precise instructions (here for instance, textual message, motion type, emotion). Alternatively, the generation could be conditioned on only a few of those instructions, and then the others should be inferred (e.g., text message only).

Overall, this objective calls for unresolved questions regarding modeling, data and evaluation.

Modeling Combining the state-of-the-art approaches from the speech synthesis and animation domains would probably be enough to build a proof-of-concept for full-body audio-visual synthesis. However, the quality of the resulting model would probably be debatable as the key aspect here is that inter-dependencies should be taken into account between all the dimensions under study. This is asking questions about how to interconnect the models (joint training) or even to build a unified model (joint model). For example, how should the different timing between each domains be addressed ? Deep learning offers the first avenues to study in order to respond, for example through multi-tasking models.

Data Inevitably, data is required, especially when thinking about complex unified models. On the one hand, the separate task involved in full-body audio-visual synthesis rely on interesting (and sometimes massive) datasets that could help to bootstrap task-specific components. On the other hand, several technologies exist to collect clean multi-modal data acted in studios. Nonetheless, it seems to be more realistic to think about capturing real (i.e. non-acted) situations to build very massive and diverse datasets. To automatically extract annotations, methods that try to estimate information from traditional video (Kocabas, Athanasiou, and M. J. Black 2020) or far field audio recordings (Peddinti et al. 2016) are particularly interesting.

Evaluation Finally, evaluation methods have to be designed. Again, objective measures coming from existing tasks can be used but specific metrics taking into account the dependence between modalities should probably be proposed also. Then, as we are dealing with reproducing human behaviors, perceptual evaluations would be certainly needed. For example, human annotators should be asked to infer the type of motion, the emotion carried and the quality of speech in a synthesized video. Ideally, the quality of the produced avatars should be measured through a satisfaction degree of users when using an application.

PUBLICATIONS

Alain, Pierre, Gwénolé Lecorvé, et al. (2018), « The IRISA Text-To-Speech System for the Blizzard Challenge 2018 », *in*: *Blizzard Challenge Workshop.*

Perquin, Antoine, Erica Cooper, and Junichi Yamagishi (2020), « Grapheme or phoneme? An Analysis of Tacotron's Embedded Representations », *in*: *arXiv preprint arXiv:2010.10694, Under review for ICASSP 2021.*

Perquin, Antoine, Gwénolé Lecorvé, et al. (2018), « Phone-level embeddings for unit selection speech synthesis », *in*: *International Conference on Statistical Language and Speech Processing (SLSP).*

— (2019), « Évaluation objective de plongements pour la synthèse de parole guidée par réseaux de neurones », *in*: *Traitement automatique du langage naturel (TALN).*

BIBLIOGRAPHY

Ainsworth, W (1973), « A system for converting English text into speech », *in*: *Transactions on Audio and Electroacoustics*.

Alain, Pierre, Jonathan Chevelu, et al. (2016), « The IRISA Text-To-Speech System for the Blizzard Challenge 2016 », *in*: *Blizzard Challenge Workshop*.

Arik, Sercan Ö et al. (2017), « Deep voice: real-time neural text-to-speech », *in*: *34th International Conference on Machine Learning (ICML)*.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015), « Neural machine translation by jointly learning to align and translate », *in*: *International Conference on Learning Representations (ICLR)*.

Battenberg, Eric et al. (2020), « Location-relative attention mechanisms for robust long-form speech synthesis », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Baylor, Denis et al. (2017), « Tfx: A tensorflow-based production-scale machine learning platform », *in*: *23rd International Conference on Knowledge Discovery and Data Mining*, ACM/SIGKDD.

Cai, Weicheng, Jinkun Chen, and Ming Li (2018), « Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System », *in*: *Odyssey: The Speaker and Language Recognition Workshop*, ISCA.

Caruana, Rich (1997), « Multitask Learning », *in*: *Machine Learning*.

Chapaneri, Santosh V (2012), « Spoken digits recognition using weighted MFCC and improved features for dynamic time warping », *in*: *International Journal of Computer Applications (IJCA)*.

Chevelu, Jonathan, Gwénolé Lecorvé, and Damien Lolive (2014), « ROOTS: a toolkit for easy, fast and consistent processing of large sequential annotated data collections. », *in*: *International Conference on Language Resources and Evaluation (LREC)*.

Cho, Kyunghyun et al. (2014), « Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation », *in*: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Cooper, Erica, Cheng-I Lai, Yusuke Yasuda, Fuming Fang, et al. (2020), « Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Cooper, Erica, Cheng-I Lai, Yusuke Yasuda, and Junichi Yamagishi (2020), « Can Speaker Augmentation Improve Multi-Speaker End-to-End TTS? », *in*: *arXiv preprint arXiv:2005.01245*.

Davis, Steven and Paul Mermelstein (1980), « Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences », *in*: *Transactions on acoustics, speech, and signal processing*.

Dehak, Najim et al. (2010), « Front-end factor analysis for speaker verification », *in*: *Transactions on Audio, Speech, and Language Processing (TASLP)*.

Desai, Srinivas et al. (2009), « Voice conversion using artificial neural networks », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Dos Santos, Cicero and Maira Gatti (2014), « Deep convolutional neural networks for sentiment analysis of short texts », *in*: *25th International Conference on Computational Linguistics: Technical Papers (COLING)*, ACL.

Fairbanks, Grant (1940), *Voice and articulation drillbook*, Harper & Brothers.

Fan, Yuchen et al. (2015), « Multi-speaker modeling and speaker adaptation for DNN-based TTS synthesis », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Goldman, Jean-Philippe et al. (2016), « The SIWIS database: a multilingual speech database with acted emphasis », *in*: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, ISCA.

Graves, Alex et al. (2006), « Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks », *in*: *23rd International Conference on Machine learning (ICML)*.

Gu, Jiuxiang et al. (2018), « Look, imagine and match: Improving textual-visual cross-modal retrieval with generative models », *in*: *Conference on Computer Vision and Pattern Recognition (CVPR)*, IEE.

Hershey, John R and Peder A Olsen (2007), « Approximating the Kullback Leibler divergence between Gaussian mixture models », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Hochreiter, Sepp and Jürgen Schmidhuber (1997), « Long short-term memory », *in*: *Neural computation*.

Holmes, Wendy (2001), *Speech synthesis and recognition*, CRC press.

Honnet, Pierre-Edouard et al. (2017), *The siwis french speech synthesis database? design and recording of a high quality french database for speech synthesis*, tech. rep., Idiap.

Hsu, Wei-Ning et al. (2019), « Disentangling correlated speaker and noise for speech synthesis via data augmentation and adversarial factorization », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Hunt, Andrew J and Alan W Black (1996), « Unit selection in a concatenative speech synthesis system using a large speech database », *in*: *International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings (ICASSP)*, IEEE.

Iskarous, Khalil et al. (2003), « CASY: The Haskins configurable articulatory synthesizer », *in*: *International congress of phonetic sciences (ICPhS)*, IPA.

Kalchbrenner, Nal et al. (2018), « Efficient neural audio synthesis », *in*: *arXiv preprint arXiv:1802.08435*.

Kocabas, Muhammed, Nikos Athanasiou, and Michael J Black (2020), « VIBE: Video inference for human body pose and shape estimation », *in*: *Conference on Computer Vision and Pattern Recognition*, IEE.

Kominek, John, Tanja Schultz, and Alan W Black (2008), « Synthesizer voice quality of new languages calibrated with mean mel cepstral distortion », *in*: *First International Workshop on Spoken Languages Technologies for Under-Resourced Languages (SLTU)*, ISCA.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012), « Imagenet classification with deep convolutional neural networks », *in*: *Advances in neural information processing systems (NIPS)*.

Krueger, David et al. (2016), « Zoneout: Regularizing rnns by randomly preserving hidden activations », *in*: *arXiv preprint arXiv:1606.01305*.

Larreur, Danielle, Françoise Emerard, and F Marty (1989), « Linguistic and prosodic processing for a text-to-speech synthesis system », *in*: *First European Conference on Speech Communication and Technology (EUROSPEECH)*, ISCA.

Li, Bo et al. (2019), « Bytes are all you need: End-to-end multilingual speech recognition and synthesis with bytes », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Luong, Hieu-Thi et al. (2017), « Adapting and controlling DNN-based speech synthesis using input codes », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Maaten, Laurens van der and Geoffrey Hinton (2008), « Visualizing data using t-SNE », *in*: *Journal of machine learning research (JMLR)*.

Mehri, Soroush et al. (2016), « SampleRNN: An unconditional end-to-end neural audio generation model », *in*: *arXiv preprint arXiv:1612.07837*.

Merritt, Thomas et al. (2016), « Deep neural network-guided unit selection synthesis », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Mikolov, Tomas et al. (2013), « Distributed representations of words and phrases and their compositionality », *in*: *Advances in neural information processing systems (NIPS)*.

Morise, Masanori, Fumiya Yokomori, and Kenji Ozawa (2016), « WORLD: a vocoder-based high-quality speech synthesis system for real-time applications », *in*: *Transactions on Information and Systems*.

Olive, Joseph (1977), « Rule synthesis of speech from dyadic units », *in*: *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE.

Oord, Aäron van den et al. (2016), « WaveNet: A Generative Model for Raw Audio », *in*: *Speech Synthesis Workshop (SSW)*, ISCA.

Peddinti, Vijayaditya et al. (2016), « Far-Field ASR Without Parallel Data. », *in*: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*.

Perraudin, Nathanaël, Peter Balazs, and Peter L Søndergaard (2013), « A fast Griffin-Lim algorithm », *in*: *Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, IEEE.

Ping, Wei et al. (2018), « Deep voice 3: 2000-speaker neural text-to-speech », *in*:

Rabiner, L (1969), « A model for synthesizing speech by rule », *in*: *Transactions on Audio and Electroacoustics*.

Rosenblatt, Frank (1958), « The perceptron: a probabilistic model for information storage and organization in the brain. », *in*: *Psychological review*.

Shen, Jonathan et al. (2018), « Natural tts synthesis by conditioning wavenet on mel spectrogram predictions », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Sini, Aghilas et al. (2018), « Synpaflex-corpus: An expressive french audiobooks corpus dedicated to expressive speech synthesis », *in*: *International Conference on Language Resources and Evaluation (LREC)*.

Snyder, David et al. (2018), « X-vectors: Robust dnn embeddings for speaker recognition », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Sotelo, Jose et al. (2017), « Char2wav: End-to-end speech synthesis », *in*: *International Conference on Learning Representations (ICLR)*.

Srivastava, Rupesh K, Klaus Greff, and Jürgen Schmidhuber (2015), « Training very deep networks », *in*: *Advances in neural information processing systems (NIPS)*.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014), « Sequence to sequence learning with neural networks », *in*: *Advances in neural information processing systems (NIPS)*.

Taylor, Paul (2009), *Text-to-speech synthesis*, Cambridge university press.

Tian, Qiao, Jing Chen, and Shan Liu (2019), « The Tencent speech synthesis system for Blizzard Challenge 2019 », *in*: *Blizzard Challenge Workshop*.

Veaux, Christophe, Junichi Yamagishi, Kirsten MacDonald, et al. (2016), « Superseded-cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit », *in*:

Wan, Vincent et al. (2017), « Google's Next-Generation Real-Time Unit-Selection Synthesizer Using Sequence-to-Sequence LSTM-Based Autoencoders », *in*: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, ISCA.

Wang, Yuxuan, RJ Skerry-Ryan, et al. (2017), « Tacotron: Towards End-to-End Speech Synthesis », *in*: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, ISCA.

Wang, Yuxuan, Daisy Stanton, et al. (2018), « Style Tokens: Unsupervised Style Modeling, Control and Transfer in End-to-End Speech Synthesis », *in*: *35th International Conference on Machine Learning*.

Watanabe, Shinji et al. (2018), « ESPnet: End-to-End Speech Processing Toolkit », *in*: ISCA.

Wu, Zhizheng and Simon King (2016), « Improving trajectory modelling for DNN-based speech synthesis by using stacked bottleneck features and minimum generation error training », *in*: *Transactions on Audio, Speech, and Language Processing (TASLP)*.

Wu, Zhizheng, Pawel Swietojanski, et al. (2015), « A study of speaker adaptation for DNN-based speech synthesis », *in*: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, ISCA.

Wu, Zhizheng, Oliver Watts, and Simon King (2016), « Merlin: An Open Source Neural Network Speech Synthesis System. », *in*: *Speech Synthesis Workshop (SSW)*, ISCA.

Yamagishi, Junichi and Takao Kobayashi (2007), « Average-voice-based speech synthesis using HSMM-based speaker adaptation and adaptive training », *in*: *Transactions on Information and Systems*.

Yao, Kaisheng and Geoffrey Zweig (2015), « Sequence-to-Sequence Neural Net Models for Grapheme-to-Phoneme Conversion », *in*: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, ISCA.

Yasuda, Yusuke et al. (2019), « Investigation of enhanced Tacotron text-to-speech synthesis systems with self-attention for pitch accent language », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Yoshimura, Takayoshi et al. (1999), « Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis », *in*: *Sixth European Conference on Speech Communication and Technology (EUROSPEECH)*, ISCA.

Ze, Heiga, Andrew Senior, and Mike Schuster (2013), « Statistical parametric speech synthesis using deep neural networks », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

Zen, Heiga et al. (2007), « The HMM-based speech synthesis system (HTS) version 2.0. », *in*: *Speech Synthesis Workshop (SSW)*, ISCA.

Zhang, Jing-Xuan, Zhen-Hua Ling, and Li-Rong Dai (2018), « Forward attention in sequence-to-sequence acoustic modeling for speech synthesis », *in*: *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE.

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Toward universal speech synthesis : harnessing linguistic and stylistic embeddings for expertise-free and flexible systems

Nom Prénom de l'auteur : PERQUIN ANTOINE

Membres du jury :
- Madame SÉBILLOT Pascale
- Monsieur AMSALEG Laurent
- Monsieur LINARÈS Georges
- Monsieur GARNER Philip N.
- Madame DAMNATI Géraldine
- Madame GUINAUDEAU Camille

Président du jury : Pascale Sébillot

Date de la soutenance : 12 Février 2021

Reproduction de la these soutenue

☒ Thèse pouvant être reproduite en l'état
☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 12 Février 2021

Signature du président de jury

Le Directeur,

Abdellatif MIRAOUI

DOCTORAT BRETAGNE LOIRE / MATHSTIC

INSA RENNES

Titre : Vers une synthèse de parole universelle : utilisation d'*embeddings* linguistiques et stylistiques pour des systèmes flexibles et sans expertise.

Mots clés : Synthèse vocale, Réseaux de neurones, *Embeddings*, Sélection d'unités, Tacotron

Résumé : La synthèse vocale est une technologie permettant de générer un échantillon de parole correspondant à la lecture d'un texte. La majorité des systèmes commerciaux repose sur une expertise linguistique, et sont limités à générer des échantillons avec une voix unique, dans un seul style de parole. Pour que la synthèse vocale devienne universelle, elle doit être facilement personnalisable et permettre de produire de nombreux styles de parole. Cette thèse à deux buts. 1) Étudier la possibilité de diminuer l'expertise linguistique nécessaire pour construire ou modifier un système de synthèse vocale. 2) Étudier la possibilité de synthétiser de la parole pour différents locuteurs, avec leur voix et accents régionaux respectifs. Ce manuscrit propose trois contributions. Premièrement, l'utilisation de la propriété d'*embedding* des réseaux de neurones pour diminuer l'expertise linguistique d'un système de synthèse par sélection d'unités. Deuxièmement, l'utilisation d'*embeddings* de caractères pour éliminer toute expertise linguistique d'un système de synthèse bout en bout. Enfin, la modélisation explicite des caractéristiques de locuteurs et d'accents à l'aide d'*embeddings* pour conditionner un modèle bout en bout et ainsi construire un système de synthèse vocale multi-locuteurs multi-accents.

Title : Toward universal speech synthesis : harnessing linguistic and stylistic embeddings for expertise-free and flexible systems.

Keywords : Text-to-speech synthesis, Neural networks, Embeddings, Unit selection, Tacotron

Abstract : Text-to-speech synthesis (TTS) turns a written text into an audio speech signal. Many commercial systems rely on human linguistic expertise, while being limited to synthesize speech for a single speaker voice and speaking style. For speech synthesis to become universal in its usage and abilities, it must be easily customizable while being able to produce widely varied speech. The goal of this thesis is two-fold. 1) To study whether it is possible alleviate the need for human linguistic expertise to build or modify a TTS system. 2) To study whether it is possible to produce speech corresponding to different speakers, with their respective tone and regionalism accent. This manuscript present three contributions. First, we show that the embedding property of neural networks can be used to lower the amount of expertise in unit selection speech synthesis. Second, we show that character embeddings can remove all linguistic expertise for end-to-end systems. Finally, we attempt to explicitly model speaker an accent characteristics in order to build a multi-speaker multi-accent end-to-end speech synthesis system.