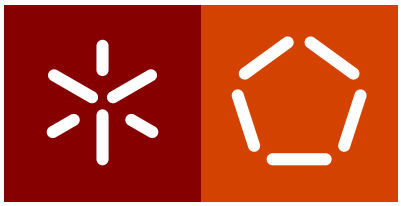
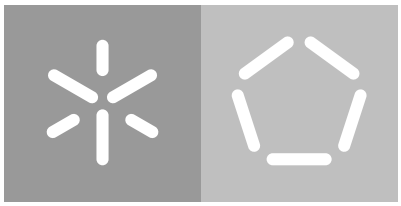


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Adriano Vaz de Carvalho Campinho

Automatic Speech Recognition for European Portuguese

July 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Adriano Vaz de Carvalho Campinho

Automatic Speech Recognition for European Portuguese

Master dissertation
Master Degree in Informatics Engineering

Dissertation supervised by
Professor Doctor Paulo Jorge Freitas de Oliveira Novais
Doctor Carlos Miguel Silva Couto Pereira

July 2021

COPYRIGHT AND TERMS OF USE FOR THIRD PARTY WORK

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisors, Paulo Novais and Carlos Pereira.

To Professor Doctor Paulo Novais, I express my gratitude for having accepted to be my advisor and for the wise suggestions.

To Doctor Carlos Pereira, I also express my gratitude for having accepted the challenge of guiding me in this curricular internship, for all the time availability and attention that he has given me, and for his encouragement and ideas.

I would like to thank NOS for providing me with this project, welcoming me with total availability, and for the enriching experience that he provided me in the context of work.

I would like to thank all my professors in both Masters in Computer Engineering and in Computer Science degrees for all the incentives they gave me and which motivated me to like this area of research.

I would like to thank my friends and family for all their support.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity.

I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

The process of Automatic Speech Recognition (ASR) opens doors to a vast amount of possible improvements in customer experience. The use of this type of technology has increased significantly in recent years, this change being the result of the recent evolution in ASR systems. The opportunities to use ASR are vast, covering several areas, such as medical, industrial, business, among others. We must emphasize the use of these voice recognition systems in telecommunications companies, namely, in the automation of consumer assistance operators, allowing the service to be routed to specialized operators automatically through the detection of matters to be dealt with through recognition of the spoken utterances. In recent years, we have seen big technological breakthrough in ASR, achieving unprecedented accuracy results that are comparable to humans. We are also seeing a move from what is known as the Traditional approach of ASR systems, based on Hidden Markov Models (HMM), to the newer End-to-End ASR systems that obtain benefits from the use of deep neural networks (DNNs), large amounts of data and process parallelization.

The literature review showed us that the focus of this previous work was almost exclusively for the English and Chinese languages, with little effort being made in the development of other languages, as it is the case with Portuguese. In the research carried out, we did not find a model for the European Portuguese (EP) dialect that is freely available for general use. Focused on this problem, this work describes the development of a End-to-End ASR system for EP. To achieve this goal, a set of procedures was followed that allowed us to present the concepts, characteristics and all the steps inherent to the construction of these types of systems. Furthermore, since the transcribed speech needed to accomplish our goal is very limited for EP, we also describe the process of collecting and formatting data from a variety of different sources, most of them freely available to the public. To further try and improve our results, a variety of different data augmentation techniques were implemented and tested. The obtained models are based on a PyTorch implementation of the Deep Speech 2 model.

Our best model achieved an Word Error Rate (WER) of 40.5%, in our main test corpus, achieving slightly better results to those obtained by commercial systems on the same data. Around 150 hours of transcribed EP was collected, so that it can be used to train other ASR systems or models in different areas of investigation. We gathered a series of interesting results on the use of different batch size values as well as the improvements provided by the use of a large variety of data augmentation techniques. Nevertheless, the ASR theme is

vast and there is still a variety of different methods and interesting concepts that we could research in order to seek an improvement of the achieved results.

KEYWORDS Automatic Speech Recognition, European Portuguese, End-to-End Learning, Data Collection

RESUMO

O processo de Reconhecimento Automático de Fala (ASR) abre portas para uma grande quantidade de melhorias possíveis na experiência do cliente. A utilização deste tipo de tecnologia tem aumentado significativamente nos últimos anos, sendo esta alteração o resultado da evolução recente dos sistemas ASR. As oportunidades de utilização do ASR são vastas, abrangendo diversas áreas, como médica, industrial, empresarial, entre outras. É de realçar que a utilização destes sistemas de reconhecimento de voz nas empresas de telecomunicações, nomeadamente, na automatização dos operadores de atendimento ao consumidor, permite o encaminhamento automático do serviço para operadores especializados através da detecção de assuntos a tratar através do reconhecimento de voz. Nos últimos anos, vimos um grande avanço tecnológico em ASR, alcançando resultados de precisão sem precedentes que são comparáveis aos atingindos por humanos. Por outro lado, vemos também uma mudança do que é conhecido como a abordagem tradicional, baseados em modelos de Markov ocultos (HMM), para sistemas mais recentes ponta-a-ponta que reúnem benefícios do uso de redes neurais profundas, em grandes quantidades de dados e da paralelização de processos.

A revisão da literatura efetuada mostra que o foco do trabalho anterior foi quase que exclusivamente para as línguas inglesa e chinesa, com pouco esforço no desenvolvimento de outras línguas, como é o caso do português. Na pesquisa realizada, não encontramos um modelo para o dialeto português europeu (PE) que se encontre disponível gratuitamente para uso geral. Focado neste problema, este trabalho descreve o desenvolvimento de um sistema de ASR ponta-a-ponta para o PE. Para atingir este objetivo, foi seguido um conjunto de procedimentos que nos permitiram apresentar os conceitos, características e todas as etapas inerentes à construção destes tipos de sistemas. Além disso, como a fala transcrita necessária para cumprir o nosso objetivo é muito limitada para PE, também descrevemos o processo de coleta e formatação desses dados em uma variedade de fontes diferentes, a maioria delas disponíveis gratuitamente ao público. Para tentar melhorar os nossos resultados, uma variedade de diferentes técnicas de aumento de dados foram implementadas e testadas. Os modelos obtidos são baseados numa implementação PyTorch do modelo Deep Speech 2.

O nosso melhor modelo obteve uma taxa de erro de palavras (WER) de 40,5% no nosso corpus de teste principal, obtendo resultados ligeiramente melhores do que aqueles obtidos por sistemas comerciais sobre os mesmos dados. Foram coletadas cerca de 150 horas de PE transcritas, que podem ser utilizadas para treinar outros sistemas ou modelos de ASR em diferentes áreas de investigação. Reunimos uma série de resultados interessantes sobre o

uso de diferentes valores de batch size, bem como as melhorias fornecidas pelo uso de uma grande variedade de técnicas de data augmentation. O tema ASR é vasto e ainda existe uma grande variedade de métodos diferentes e conceitos interessantes que podemos investigar para melhorar os resultados alcançados.

PALAVRAS-CHAVE Reconhecimento Automático de Fala, Português Europeu, Aprendizagem Ponta a Ponta, Recolha de Dados

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AM Acoustic Model.
ANN Artificial Neural Network.
ARQ Automatic Repeat Request.
ASG Auto Segmentation Criterion.
ASR Automatic Speech Recognition.
ASRS Automatic Speech Recognition System.

BP Brazilian Portuguese.

CER Character Error Rate.
CNN Convolutional Neural Network.
CRS Continuous (automatic) Speech Recognition.
CTC Connectionist Temporal Classification.
CTS Conversational Telephone Speech.

DAT Discriminative Adaptive Training.
DBN Dynamic Bayesian Network.
DCT Discrete Cosine Transform.
DFA Deterministic Finite Sate Automaton.
DNN Deep Neural Networks.
DWT Discrete Wavelet Transform.

EP European Portuguese.

FFT Fast Fourier Transform.
FSA Finite State Automaton.

GD Gender Dependent.
GI Gender Independent.
GMM Gaussian Mixture Model.

HMM Hidden Markov Model.
HSR Human Speech Recognition.

IPA International Phonetic Alphabet.

LM Language Model.

LPC Linear Prediction Coefficients.
LPCC Linear Prediction Cepstral Coefficients.
LSF Line Spectral Frequencies.
LSTM Long Short Term Memory.
LV Large Vocabulary.
LVCSR Large Vocabulary Continuous Speech Recognition.

MCE Minimum classification Error.
MFCC Mel Frequency Cepstral Coefficients.

OOV Out of Vocabulary.

PLP Perceptual Linear Prediction.

RNN Recurrent Neural Network.

SAMPA Speech Assessment Methods Phonetic Alphabet.
SAT Speaker Adaptive Training.
SD Speaker Dependent.
SI Speaker Independent.

TCN Temporal Convolutional Neural Network.
TDNN Time Delay Neural Network.

VOIP Voice Over Internet Protocol.
VTLP Vocal Tract Length Perturbation.

W2L Facebook Wav2letter.
WER Word Error Rate.

1

INTRODUCTION

1.1 MOTIVATION

Since the emergence of computers, researchers have been looking for ways to make more intelligent computer systems and improving human-machine communication, one of these steps involves the process of computers understanding Speech. Not long ago, this type of technology was only present in science-fiction stories, but today, Automatic Speech Recognition (ASR) is a reality.

ASR is the process where a machine is able to recognize human speech and produce a 'string output' of the sentences in its written form. The data can then be used not only to perform actions but also to obtain information from the user. This process has evolved a long way from the early days where it was only possible to recognize a small vocabulary of words in a paused manner and speaker-dependent context, to today's state-of-the-art recognition systems that are able to achieve comparable results with human perception. For humans to be able to communicate with devices in a natural way, it is necessary for the Speech Recognition models to have good accuracy, low processing time, and depending on the context it may be necessary to have the ability to recognize different speakers.

In today's age, many ASR services are available, and these are generally able to achieve fast and somewhat accurate results for general use cases, but better results can be achieved when using models specifically designed for a defined task. Many aspects can prove as hindrances to a Speech Recognition model, that can be, e.g., background noise interference, differences in accents, dialects, and physiology affecting our speech, differences in vocabulary size and content, low audio quality and others. All of these can lead to miss recognition of utterances. Local inference ASR, or also known as offline ASR systems, have also recently gained interest due to increasing importance on both subjects of cyber-security and online privacy, and for these reasons a lot of Speech-to-Text engines are becoming open-source. Furthermore, these open-source projects are also showing significantly good performance and accuracy rates compared to current Speech-to-Text commercial engines [1].

What we have been watching recently is that the development of these ASR Systems have been benefited by the quick evolution of techniques like Deep Learning, of the advance

in computing power and the availability of data and information [2]. However, what the literature shows us is that the focus of these new developments is mainly focused on English and Chinese languages, possibly largely motivated by being in the top ranking of the most spoken languages in the world [3]. In the case of other languages with a lower amount of representation in world communication, like Portuguese, there has not been nearly as much investment in the development of these types of systems.

Although some efforts have been done to develop Brazilian Portuguese (BP) ASR, in the literature review developed around this theme, to the best of our knowledge, there is no model for the European Portuguese (EP) dialect that is freely available for general use.

1.2 OBJECTIVES

The automatic speech recognition is the task of transcribing a speech audio signal into a written language. The process of ASR opens the doors to a broad set of possible customer experience enhancements, especially associated with the dynamics of communication with customers.

While the history of statistical approaches for automatic speech recognition is long, only a few very spontaneous projects have used such development on the Portuguese Language. Focused on this problem, the research question to which this study intends to answer is as follows: ***What steps and procedures are involved in creating an ASR system for EP?*** Based on this question, which constituted the starting line for the work we propose to develop, the main objectives of this work are:

- Research and describe ASR systems and how do they work;
- Collect as much possible transcribed speech and develop an ASR model for EP;
- Test multiple methods to improve the accuracy of the system, and compare the obtained results with available commercial alternatives.

1.3 DOCUMENT STRUCTURE

This document is organized into eight chapters. Chapter 2 describes, in its essence, what is speech, and how it works. In addition, it also contains a brief introduction to the current foundations of modern ASR systems. Chapter 3 is dedicated to describing the application of speech recognition, an evolutionary perspective of how ASR technologies evolved over the years. Furthermore, application areas are also characterized in that chapter, as well as the main parameters involved in this type of system and how their performance can be evaluated.

Chapter 4 deals with the structure and paradigms of models used to recognize discourse. A reflection is done comparing two of the most prevalent paths in this field (i.e., traditional models and the recent end-to-end models). At the end of the chapter, a general introduction of the technologies used in the development of our system is present. In Chapter 5, various methods used to improve ASR models are introduced, which were described in previous works to be able to both improve accuracy, training time, and to increase the robustness of the model. These methods were given special attention for their ability to possibly limit the effects of small amount of collected data.

In Chapter 6, a description of all data collected (and their source) to train the ASR model is presented. The transcribed speech in English used to train the back-model, the EP transcribed speech to train our new model, and the EP text data used to create the Language Models are detailed. Chapter 7 contains a description of the specification of both the Acoustic's and Language's models created, the methodology used during development and the results of such methods. In addition, the final results are compared with other ASR services.

Finally, in Chapter 8, the conclusions and contributions are summarized and we present a brief discussion about future work.

2

THE HUMAN SPEECH

2.1 HUMAN VOICE

ASR models assume, in their construction, the knowledge of what the human voice is and how it is produced. This prior knowledge allows us to better understand this speech production process, which despite being intuitive for us humans, is a quite complex process.

The joint work of three different systems is necessary for our body to produce the sounds that compose our voice [4], as we can see in Figure 1. These systems can be referred to as the respiratory system, the phonetic system, and the resonance system, whose distinct roles are necessary for voice production. It should be noted that these systems' primary functions are not the articulation of speech, but breathing, chewing, and smell, among others. It is necessary to describe these three systems in order to understand their functions and the organs that constitute them:

RESPIRATORY SYSTEM: Responsible for our breathing, includes our lungs, rib-cage, chest muscles, diaphragm, and windpipe;

PHONATORY SYSTEM: Responsible for the production of sound, includes our larynx and vocal cords primarily;

RESONANCE: Responsible for shaping and amplifying the sound produced in the Phonatory System into vowel and consonant sounds we use to communicate. This includes a variety of different parts in and around the mouth.

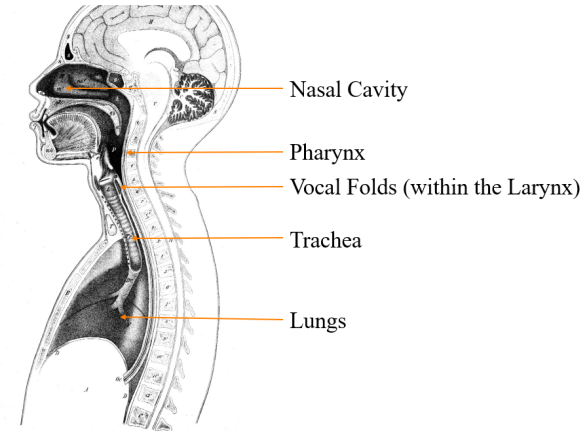
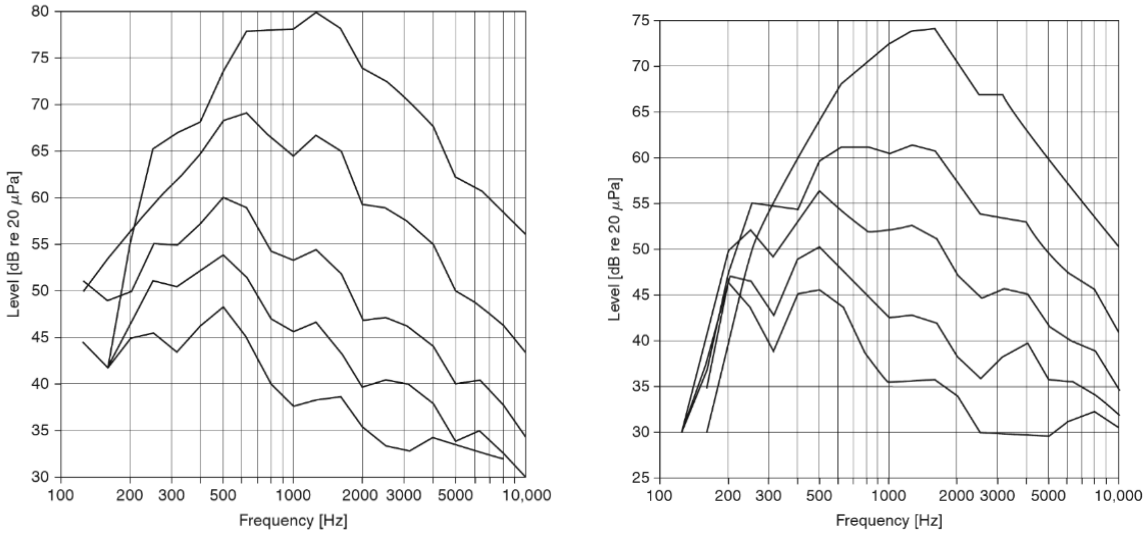


Figure 1: Sagittal Section of the Vocal Tract, extracted from [5]

These three systems described above are the physiologically responsible for producing the sounds in speech, which can be divided into two different types, voiced and voiceless sounds. When we breathe, our vocal folds are open to allow air to flow from your upper airway into your trachea and lungs. For voiced sounds (/b/, /j/, /r/, etc.), when we speak, we close our vocal folds and begin to exhale, causing an increase in pressure that pushes the vocal folds open. This opening releases the tension and lets the vocal folds close up again. This continuous opening and closing cycle is what produces the vibrations that are the source of our voices, these are a series of sound wave frequencies with an average fundamental frequency of Fo. Alternatively, when producing voiceless sounds (/f/, /p/, /s/, etc.), we keep our vocal folds relaxed even when exhaling, and air passes through unobstructed. If the air moves very quickly, the turbulence causes a different kind of phonation: whispering [4]. Both voiced, and voiceless sounds are then amplified and shaped by our Resonance system, creating articulation and the different sounds we use during communication.

There is a substantial amount of data on the voice fundamental frequency (Fo) in speech [6]. The typical reported values obtained for the frequency of the voice fundamental in speech vary around 120Hz for men, 210 Hz for women [6], and around 270Hz for children [7]. However, a higher deviation from the base frequency is typically noticeable in tonal languages, such as Mandarin [8]. As stated, this value varies from person to person but can be influenced by multiple factors, most notably studied are age and sex. However, aspects like type of discourse, mood, and language, as well as the presence of a smoking habit, can also have deep effects on the value of Fo [9]. But the human voice is not a simple sin wave. It is a composite waveform containing overtones and harmonic frequencies far above these ranges. These are responsible for the fact that the spectrum of speech is generally found in the frequency range from 100-8000Hz.



(a) Male Voice (b) Female Voice

Figure 2: Effect of Various Levels of Vocal Effort, extracted from [10]

One of the critical aspects that affect how we sound is the vocal effort, meaning the effort made by the speaker to compensate for the distance between him and the listener('s) or surrounding noise and several other factors. Is is evident in the Figure 2 that a higher effort is associated with a energy moved towards higher frequencies associated with a higher pitch. Inherently "screamed" speech sounds different from talking with a calm voice [10]. While language can be something that groups many people, the voice's character and sound are different from person to person.

2.2 PHONETICS

During pronunciation, we split each word into syllable(s). These are units of pronunciation having one vowel sound, with or without surrounding consonants, forming the whole or a part of a word. In simplified terms [11]:

CONSONANTS: Consonants are sounds that are articulated with a complete or partial closure of the vocal tract. They can be voiced or voiceless and they break up the stream of vowels and require more precise articulation;

VOWELS: Vowels are voiced sounds that are pronounced without any obstruction in the vocal tract.

This classification is not perfect since the syllabic function needs to be considered [4]. In electronics, acoustics, and related fields, the waveform of a signal is the shape of its graph as

a function of time, independent of its time and magnitude scales and of any displacement in time. By analyzing these types of representations of a speech waveform (Figure 3), we can detect some interesting aspects of EP pronunciation:

- Distinguishable gaps of silence between words are not a mandatory occurrence;
- There are regular peaks in volume amplitude;
- Vowels are generally voiced more loudly and have increased duration;
- There is a visible reduction in volume during pronunciation when changing from one vowel to another.

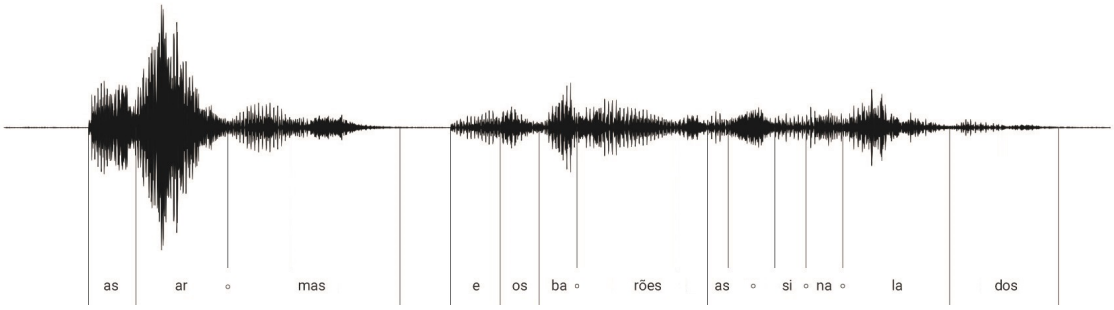


Figure 3: Waveform Audio of the first line of the epic poem 'Os Lusíadas', said by António Fonseca

The Portuguese alphabet is composed of 23 different letters (5 Vowels and 18 Consonants), and these letters can have different pronunciations depending on their surrounding letters. Portuguese also makes use of five different diacritics: the 'cedilla' [ç], acute accent [á, é, í, ó, ú], grave accent [à], circumflex accent [â, ê, ô] and tilde [ã, õ]. Because all these different letters can have different pronunciations, it is common when studying speech to divide it into phonemes. These are distinct abstract units of sound used to distinguish one word from another in a particular language.

To better understand what these are, a Phone is an atomic unit of sound and describe the physical sound produced when a person speaks or any other sound. In contrast, a phoneme is an abstract concept, not meant to describe directly the physical essence of the sound. In sum, many phones can be mapped to single phoneme, represented by a single letter or a specific sequence of letters in a particular language. Additionally a phoneme is a speech sound in a given language that, if swapped with another phoneme, could change one word to another, for example in Portuguese if we replace the /r/ sound in 'rato' with the /f/ sound, we reproduce the word 'fato'. There are around 37 different phonemes in EP [12], although it was found that this number varies between sources. These different sounds

can be split into distinct groups: 18 Vowel Sounds (9 Oral Vowels, 5 Nasal Vowels, and 4 Semi-Vowels) and 19 Consonant Sounds [12].

There are multiple representation of Phonemes but 2 of the most commonly used are IPA and SAMPA. IPA stands for ”International Phonetic Alphabet” and SAMPA stands for ”Speech Assessment Methods Phonetic Alphabet”. Both consist of phonetic alphabets but SAMPA is a partial encoding of the IPA, with the benefit of being readable in ASCII format. But as Unicode support for IPA symbols becomes more and more widespread, the necessity for SAMPA has decreased. The IPA and SAMPA list of the phonemes present in EP is as described in the following Tables 1, 2, 3, 4.

IPA	SAMPA	Word Example
[i]	[i]	v**i**
[e]	[e]	v**ê**
[ɛ]	[E]	p**é**
[a]	[a]	p**á**
[ɐ]	[6]	p**ar**a
[ɨ]	[1]	d**e**
[ɔ]	[O]	s**o**l
[o]	[o]	p**ô**r, s**o**u
[u]	[u]	t**u**

Table 1: Oral Vowels Sounds

IPA	SAMPA	Word Example
[ĩ]	[ĩ]	**s**im
[e]	[e]	p**e**nte
[ɐ̃]	[6~]	rom**ã**, b**a**nco
[õ]	[õ]	p**õ**e, p**o**nte
[ũ]	[ũ]	at**u**m

Table 2: Nasal Vowels Sounds

IPA	SAMPA	Word Example
[j]	[j]	**p**ai
[j̃ / j]	[j]	mã**e**
[w]	[w]	**p**au
[w̃/w]	[w]	c**ã**o

Table 3: Semi-Vowels Sounds

IPA	SAMPA	Word Example
[p]	[p]	**p**á
[b]	[b]	**b**em
[t]	[t]	**t**u
[d]	[d]	**d**ou
[k]	[k]	**c**acto
[g]	[g]	**g**ato
[f]	[f]	**f**é
[v]	[v]	**v**ê
[s]	[s]	**s**abe, **p**asso, **c**aça
[z]	[z]	ca**s**a, a**z**ar
[ʃ]	[S]	**ch**ave
[ʒ]	[Z]	**j**á
[m]	[m]	**m**ão
[n]	[n]	**n**ão
[ɲ]	[J]	ve**nh**o
[l]	[l]	**l**á
[ʎ]	[L]	va**lh**a
[ɾ]	[4]	ca**r**o
[ʀ]	[R]	ca**rr**o

Table 4: Consonant Sounds

Portuguese is a pluricentric language, meaning it is a language with several interacting codified standard forms, often corresponding to different countries. There are notable differences between EP, BP, and Angolan Portuguese (AP) and all other variants spoken in different parts of the world. Furthermore, it is essential to note that Portuguese phonology varies highly between its dialects, even located in the same country, wherein some cases, speakers of different dialects have difficulties comprehending each other. This can have severe effects on speech recognition accuracy [13] (the same happens to humans [14]), and for this reason, this work's focus, is limited to the generally regarded standard EP pronunciation.

2.3 SOUND REPRESENTATION

There are multiple ways to represent sound, and these can be more or less useful on the state of the art Speech Recognition systems.

When looking at the previously shown wave-plots, we have seen that there are apparent differences between different sounds, but these differences are not as noticeable in some other cases. These types of visualizations only show us the loudness (represented by the amplitude) of sound waves changing through time. The bigger the amplitude of the wave, the more volume at that specific time frame. An amplitude = 0 represents complete silence.

As stated before, the audio signal of a voice is a complex composition of multiple single-frequency sound waves. But when sound is captured and represented in these wave-plots, only the resultant amplitudes of adding those multiple waves together is shown, as shown in Figure 4. These diagrams are not very informative since they only represent loudness over time. For machine learning purposes, a denser representation differentiating the strength of the frequencies present in the sound is needed.

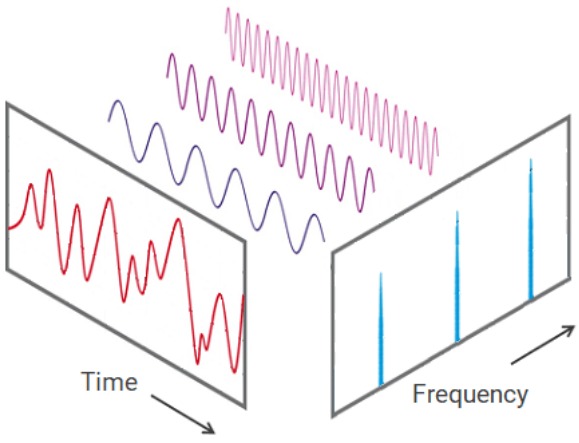


Figure 4: Resultant Amplitude of Multiple Waves

Using Fourier Transform, we can decompose a wave into its constituent signals. Providing information about what frequencies are present and the magnitude of each one of them in the signal. Using a Spectrogram, we can have a visual representation of the signal strength at various frequencies present over a period of time in a particular waveform. Spectrograms are two-dimensional graphs, where the X-axis represents the time, the Y-axis represents frequency, and where variations on a color gradient represent the signal strength.

To form a spectrogram, the audio signal is split into small frames with a fixed duration to which the Discrete Fourier transform is calculated, in this case, a value of 20ms was used as it is a common value for speech recognition purposes. Although having windows as short makes us not lose any phoneme (since humans can not speak more than one phoneme in such a short period of time), it is frequent to have these windows overlap to prevent loss of frequencies [15]. An overlap of 50% is commonly used for speech recognition so a stride value (step size) of 10ms was used.

Humans do not perceive frequencies on a linear scale. We are better at detecting differences in lower frequencies rather than higher frequencies. For example, the difference between 500 and 1000 Hz is much more noticeable to our ears than the difference between 10,000 and 10,500 Hz, even though the distance between the two pairs is equal. In [16], a scale was proposed where lengths in pitch sounded equally distant to the listener and called it the mel scale.

Over the years multiple formulas have been suggested for the calculation of this scale but today $y = 2595 * log_{10}(1 + \frac{x}{700})$ is commonly used [17], and can be seen plotted on Figure 5.

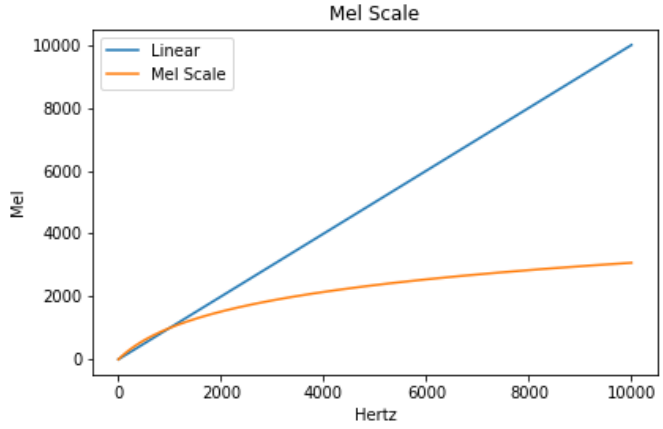


Figure 5: Mel Scale

Because of these differences in frequency perception, it is common in Speech Recognition to use Mel Spectrograms instead of the standard frequency variety. To do this, we have to simply convert the values on the y-axis of the spectrogram to the respective values on the mel scale, and we obtain a Spectogram like the one seen on Figure 6.

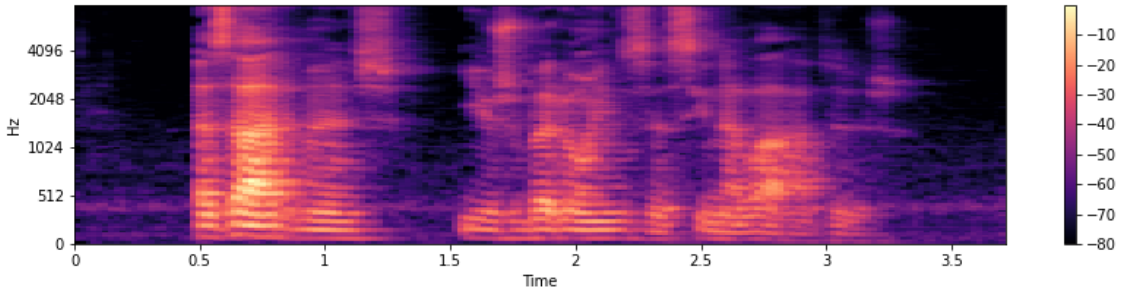


Figure 6: Mel Spectrogram of the first line of the epic poem 'Os Lusíadas', said by António Fonseca

When identifying different sounds such as human voice, our ears are more sensitive to peaks in the signal spectrum. These distinctive, resonant frequency of the acoustic signal produced by speech or singing are called formants [18] and can be seen on Figure 7. The information required to distinguish between different vowels can be simply represented by specifying 2-3 specific frequencies in the spectrum.

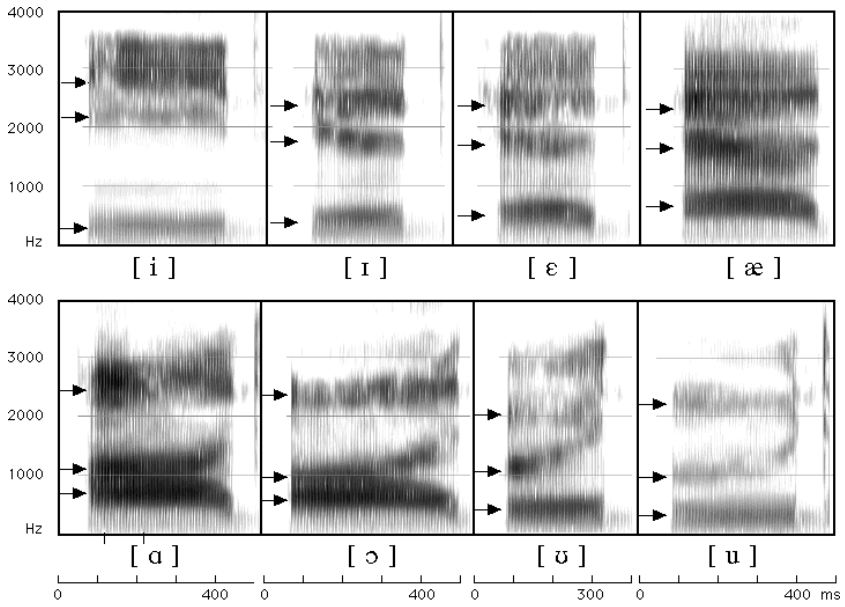


Figure 7: Formant present in different Vowel Sounds

These specific frequencies are generally called formants and are usually referred to as F1-3, with the lowest frequency called F1, the second F2, and the third F3, although the first two formants are sufficient to identify the sound in most cases. As stated before, the Fo is affected by several factors, including age and gender, this is also true for the remaining frequencies, and the effect of age and gender. As it is possible to notice on Table 6, there

is a distinctive difference between the values for the formant frequencies of several vowel sounds for male, female, and child, respectively.

vowel		F1	F2	F3
/ee/	male	270	2290	3010
	female	310	2790	3310
	child	370	3200	3730
/e/	male	530	1840	2480
	female	610	2330	2990
	child	690	2610	3570
/ae/	male	660	1720	2410
	female	850	2050	2850
	child	1030	2320	3320
/ah/	male	730	1090	2440
	female	590	1220	2810
	child	680	1370	3170
/oo/	male	300	870	2240
	female	370	950	2670
	child	430	1170	3260

Table 5: Average Formats for different Vowels

Table 6: Average Formats for different Vowels, extracted from [19]

Since we can convert one audio file into an image, we can transform Speech Recognition into an image classification problem where each image represents a spoken utterance from left to right as time progresses. We can train a Speech Recognition System to analyze these images and identify different aspects, like formants, and recognize the different sounds present in the audio.

2.4 WORDS AND LETERS

For many years, many of these linguistics notions and terms were used when developing ASR. Most of these concepts, like syllables, vowels, and consonants are ignored by most successful and state of the art speech recognizers, although these are still useful concepts to better understand ASR systems [11].

In its core principle, the key focus is finding the most probable word sequence given the observed audio signals. We want to match the audio signals to words, but due to the

possible variants in phrases, we want to explore all promising possibilities, that generate a coherent speech.

In the EP Language, there are thousands of different words. All of them are combinations of the phonemes that were referred to in the previous tables. At the time of writing, the Portuguese Dictionary 'Priberam' has around 133 000 unique entries (and growing number) that include general vocabulary commonly used in conversations and the most common terms of the main scientific and technical areas [20]. In Portuguese, there also special cases where word pairs are homonyms, this happens when 2 or more words are spelled and sound the same way but have different meanings. But as expected, not all words (as can be seen on Figure 8), phonemes, or even letters are found as commonly in the Portuguese Language as others.

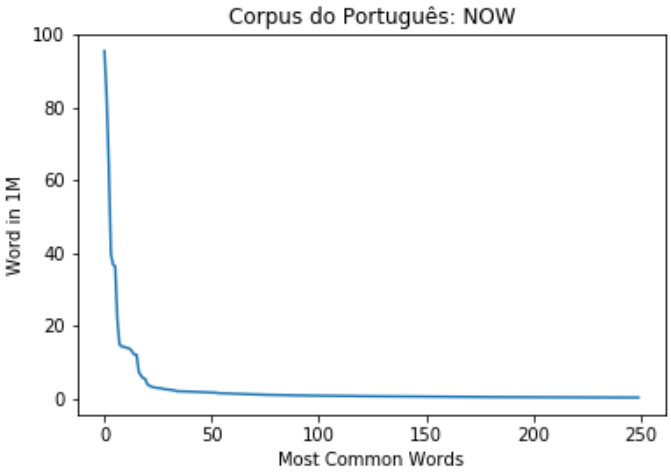


Figure 8: Most Common Words of 'Corpus do Português: NOW'

For this reason, the accuracy of certain words can have a much bigger impact on the end result of the model developed. Previous work on recognition of spontaneous monologues and dialogues has shown that infrequent words are more likely to be misrecognized [21]. Another important aspect is that the frequency of most words is highly dependant on both the context and their surrounding words. Because of this, the task of finding the most probable word sequence given the audio is usually divided into two different tasks:

1. Transform the audio signal into phonemes or other linguistic units that represent speech;

2. Transform these previous units into the most likely word sequence while making the phrase grammatically and semantically correct.

2.5 SUMMARY

Speech is a complex process that results from the joint work of three different systems (respiratory, phonatory, and resonance system) necessary to produce the sounds that constitute the voice. While language can be something that groups of people have in common, the sound and character of the voice is individual from person to person.

The Portuguese alphabet consists of 23 different letters, which can have different pronunciations so it is customary to divide speech into phonemes in order to be able to analyze it. The fact that the phonology of Portuguese is varied, can have serious effects on the accuracy of speech recognition led us to consider in this work only the standard European pronunciation.

There are several ways to represent sound and they can be useful in the latest generation of speech recognition systems. Since we can convert an audio file into an image, we can transform speech recognition into an image classification process. The main focus is to find the most likely word sequence given by the observed audio signals, exploring all the promising possibilities that generate a coherent speech.

3

SPEECH RECOGNITION

3.1 APPLICATION AREAS OF SPEECH RECOGNITION

Speech Recognition is changing the way people interact with technology. For many years, this type of technology was only imaginable in Sci-Fi movies. Characters like Space Odyssey's HAL 9000 Computer, the Star Trek Ship system, and even the iconic R2D2 and C3PO of the Star Wars Universe, used Speech Recognition to interact with their non-robot counterparts. In these movies, the use of this type of technology seems to be a thing from a far, far away future, but it is, in fact, an approaching reality.

In recent years, we have seen an ever-growing number of applications for Speech Recognition, many of these, surround facilitating interaction for human-machine communication. Tools like a mouse, keyboard, trackball, and the touch-screen, have been used for years to fill this purpose, but these methods are not always the most suitable for all occasions.

Oral speech is the natural form of communication between humans and can be, in certain situations, a more practical method for accessing information and interacting with devices. We saw the rise of voice recognition technology in our phones. In recent years, we are bringing it into our homes. Today, businesses in a wide array of sectors are investigating different ways to integrate it into their technological catalogs.

Digital Personal Assistants

One of the most commonly known uses of this technology is Voice Assistants. A number of 'Tech Giants' have invested in the development of voice assistants in the last decade, of these, are examples: Amazon's Alexa, Apple's Siri, Microsoft's Cortana, and Google Assistant. These are primarily present in our phones and smart speakers and are designed to hear and interpret our voice and subsequently respond to our questions or execute a specific action.

Microsoft's 2019 Voice Report, stated that 69% of the respondents have used digital assistants and expected that 75% of household would have at least one smart speaker by 2020 [22].

These are primarily used for playing music, radio, podcasts, and audio-books, as well as receiving weather/traffic information, setting alarms and calendar appointments, receiving news updates, home automation purposes, and making to-do lists. Furthermore, with the Internet of Things and the ability to connect home appliances, smart lights, thermostats, and other technologies in cohesive systems, the ability to control these with voice commands can be very convenient and improve their usability, especially interesting case when providing support for elderly and/or disabled people.

Office / Industries

As Speech Recognition evolved, the industries are starting to take advantage of this type of technology. In the workplace, it can, for example, be used to access reports and information, schedule and record meetings and video conferences, as well as many office applications also include the ability for dictation. Systems like Nuance's Dragon [23] are common in this let users integrate Speech Recognition on their enterprise workflows. Another interesting approach is the possibility of measuring stress levels in the workplace, a variety of projects have found success on using machine learning for stress detection [24, 25, 26].

For the business client, having access to a voice assistant that eases their purchase of products and services, provides them with information, and assist them with simple issues can both boost their customer loyalty and satisfaction. The use of voice applications by clients can also create new ways to make a business better understand their clients.

Telecommunications

In today's day and age, it is common for mobile phones to support voice dialing by name or number. It is also increasingly common for companies to include automation of operator-assisted services (for example: call processing systems to automate operator service routing). TalkDesk [27] is at the time a big example of a provider of these types of services.

Medical

Healthcare is also one of the areas that in recent years has gotten much investment and success on the use of voice-controlled systems [28, 29, 30, 31]. Similarly to the Office Environment, dictation systems (with medical vocabulary) can be bring increased productivity and better patient care. It can both reduce the time a physician spends writing during the appointment, allowing for shorter appointments, as well as ensures that all the essential data is digitally stored and easily accessible to other relevant specialists that are or will be in the future, concerned with a patient's health.

In environments where seconds are crucial, and where hands and eyes are busy like during medical surgery, immediate access to information via other means can have a significant impact on both the safety and efficiency of the procedures.

Automotive Market

The automotive industry has been in recent years started to utilize speech recognition as a core part of the car systems. For years the use of mobile phones while driving posed as one of the most common causes of car accidents. With the help of SR, texting while on the road or using a GPS to find their destination can no longer be an obstacle for the drivers.

Apart from Apple's CarPlay [32] and Google's Android Auto [33] universal systems, multiple other car manufacturers have experimented with the integration of speech recognition in their car systems.

Voice Biometrics

Another interesting recent development that stems from voice recognition technology is voice biometry. Voice Biometrics systems, allows us to create a digital profile of someone's voice and are able to perform authentications (identify a person's individuality) through natural voice pattern instead of passwords or other authentication methods [34].

Language Learning and Translation

Learning a different language can be a complex and difficult process from a wide array of viewpoints. A person needs to understand word order, pronunciation, lexicology, grammar, along with a host of other linguistic domains.

In recent years, many different language-learning apps/software, like DuoLingo [35], have started using voice recognition to evaluate their users to help teaching users how to properly pronounce words in a foreign language. This can be done by comparing a person's speech on a sentence to a series of native speaker variants of the same sentence and establishing whether the two are similar enough to be recognized, or if not, informing the user the particular aspects of their syntax or pronunciation that need to be revised.

On the other side of the spectrum, Automatic Translation has seen recent success in the ability to bring down language barriers. Voice recognition-powered translations can provide us with immediately translated and subtitled video and audio content.

3.2 History of Speech Recognition

ASR has been an active research area for over 65 years and it has come a long way from its humble start. In recent years, there has been a heavy investment in this area providing very

impressive results. ”We’ve seen more progress in this technology in the last 30 months than we saw in the last 30 years” said Shawn DuBravac in April 2017 [36].

3.2.1 *1950s - 1960s*

The earliest attempts to devise ASR systems were made in the 1950s and 1960s. While signal processing and computer technologies were still very primitive, researchers started experimenting with the ideas of acoustic phonetics.

- 1952 - The first known and documented speech recognition system is released, with the name ”Audrey”. This system designed by Davis, Biddulph, and Balashek at Bell Laboratories, was fully analog and able to recognized strings of digits (in English) with pauses in between them spoken by a single voice. This system had 97-99% accuracy after being adapter to speaker [37];
- 1956 - Independently, H. F. Olsen and H. Belar from RCA Laboratories developed a Speech Recognition system capable of recognizing 10 isolated monosyllabic words;
- 1959 - Fry and Denes at the University College in England, built a phoneme recognizer of four vowels and nine consonants. [38] They achived better phoneme recognition accuracy for words consisting of two or more phonemes by incorporating statistical information concerning phoneme sequences in English, marking the first use of statistical syntax at the phoneme level in ASR;
- 1959 - Forgie and Forgie at MIT Lincoln Laboratories developed a system capable of recognizing ten vowels embedded in a /b/-vowel-/t/ format in a speaker-independent manner [39];
- 1960 - Japanese vowel recognition was performed by J.Suzuki and K. Nakata from the Radio Research Lab in Japan [40];
- 1962 - At the World’s Fair, IBM demonstrated their ”Shoebox” machine. This device recognized and responded to 16 spoken words, including the ten digits from ”0” through ”9”. Beyond the digits, this machine was able to recognize words such as ”plus”, ”minus” and ”total” as they were spoken into a microphone, and calculate and print answers to simple arithmetic problems [41];
- 1962 - Sakai and Foshita from Kioto University built an hardware phoneme recognizer using a speech segmenter and zero-crossing analysis of different regions of the input utterance [42];
- 1963 - NEC Laboratories in Japan built a hardware digit recognizer;

- Martin at RCA Laboratories developed solutions to problems associated with the non-uniformity of time scales in speech events, improving accuracy by reliably detecting speech start and end;
- 1968 - Vintsyuk, in the Soviet Union, proposed the use of the now known as Dynamic Time Warping method for aligning speech utterances, unfortunately his efforts were largely unknown in other countries until the 1980s [43].

3.2.2 *1970s*

During this decade, there were significant advances on Speech Recognition and there was achieved a number of significant milestones.

- 1971-1976: During this years, the Speech Understanding Research (SUR) was created from the investment of the US Department of Defense and DARPA on the topic of Speech Recognition. This program developed many similar systems (Hearsay-II [44] and HWIM - Hear What I Mean [45]) and technologies including the 'Harpy' Speech System, capable of recognizing around 1000 words [46];
- 1978 - Sakoe and Chiba at NEC Laboratories in Japan, started to use dynamic programming techniques to solve speech alignment problems. [47]
- Bell Laboratories: Improved on the developments made on the previous decades, aiming at making speaker-independent ASR systems.

3.2.3 *1980s*

During the 80s, speech recognition vocabulary went from a few hundred words to several thousand words and the recognition of connected words started being possible. One of the breakthroughs came from the use of a statistical method known as the Hidden Markov Model (HMM) in Speech Recognition. [48]

- 1982 - Time-varying parametric modeling of speech was introduced [49];
- 1986 - IBM released 'Tangora', the first real-time PC-based large vocabulary isolated word dictation system [50];
- 1989 - The people at Carnegie Mellon University developed the Sphinx Speech Recognition System, a large-vocabulary speaker independent continuous speech recognition system. [51] Updated versions of this systems are still in use today.

3.2.4 *1990s*

During the 90s, the use of personal computer became more and more common, this propelled forward Speech Recognition by making more widely used.

- 1990 - Dragon releases DragonDictate, the first commercial speech recognition program for general-purpose dictation for 30000$. It was a connected word type system, meaning a pause was need in between (at least a quarter-second), and had a general vocabulary of 30.000 words plus a more field specific 80.000-word dictionary built in [52];
- 1990 - Perceptual Linear Predictive (PLP) was introduced. This technique uses three concepts from the psychophysics of hearing to derive an estimate of the auditory spectrum: (1) the critical-band spectral resolution, (2) the equal-loudness curve, and (3) the intensity-loudness power law.
- 1994 - PLP technique was further improved with the introduction of relative spectra (RASTA) [53];
- 1997 - Dragon releases Dragon "NaturallySpeaking", another commercial speech recognition program this time for continuous speech.

3.2.5 *2000s*

Apart from the constant improvements and the continuous trivialization of the use of ASR, during the 2000s many further applications were developed, this include: Speaker Recognition, Human-Machine Interaction, non-English Speech Recognition and non-native speech recognition. Moreover, large tech companies started integrating speech recognition into their products.

- 2002 - Effective Affordable Reusable Speech-to-Text (EARS). The goal of this five-year project is to significantly advance the state of the art in multi-lingual speech recognition of both broadcast news and human-to-human conversational speech [54];
- 2002 - Microsoft integrates speech recognition in their Office products;
- 2005 - Global Autonomous Language Exploitation (GALE) that as the objective to produce a system that is able to automatically take multilingual newscasts, text documents, and other forms of communication, and make their information available to human queries [55];

- 2007 - Google Released GOOG-411, a telephone service that provided a speech-recognition-based business directory search [56];
- 2008 - Google launches Voice Search app.

3.2.6 *Systems - State of the Art*

In the last decade, speech recognition became more and more common, as it was quickly adopted by a vast number of users. Companies like Google, Apple, Microsoft, and Amazon are strongly investing in SR research and many improvements have been done to speech processing technologies, not only in improving on the accuracy of previous systems but recently we have seen on device recognition (ASR done locally on smartphone or IoT device, without the need to be connected to the Internet).

Although for many years, this area of investigation was dominated by close-source and proprietary software. There has been a surge of open source projects with growing communities and large amounts of support around them. Noticeable ones are Kaldi, Mozilla DeepSpeech, PaddlePaddle DeepSpeech, and Facebook Wav2letter.

These engines are able to achieve accuracy comparable to the current status of commercial products of ASR systems and personal assistants, are evaluated in terms of accuracy rate, and performance-based on evaluating runs on multiple machines, of different based architectures, some of which applying hardware acceleration.

It is also important to refer the move towards leaving Traditional Speech Recognition Methods to an emerging paradigm in the field of Deep Neural Network (DNN)-based Speech Recognition.

Kaldi

Kaldi is a open-source ASR toolkit, originally released in 2009 [57]. The code of this toolkit is written in C++ language and is freely released by the Apache License v2.0. Over the years it has suffered numerous updates provided by a large community of researchers and professionals that use and support it. The main goal of Kaldi is to build a software that is modern, easy to use, flexible enough to extend and update, and powerful at training the acoustic model [58].

In its beginnings, Kaldi followed a traditional approach of ASR, following the Gaussian Mixture Model (GMM)-HMM methods, which were, at the time, one of the more conventional ways to build speech recognition models. But over the years, constant updates changed the way it is structured, adding multiple options of models to its users, including a DNN model to be used instead of GMM. The later approach using DNNs is found to achieve a better performance in terms of Word Error Rate (WER) [59].

Facebook Wav2Leter

Facebook AI Labs, released in 2016 a paper announcing a new model ASR model called "Wav2letter". This approach was to be solely based on end-to-end on CNNs, lighter in terms of computational power and model size, while at the same time, was able to achieve comparable results to all current state of art ASR models [60], and its architecture is represented in Figure 9.

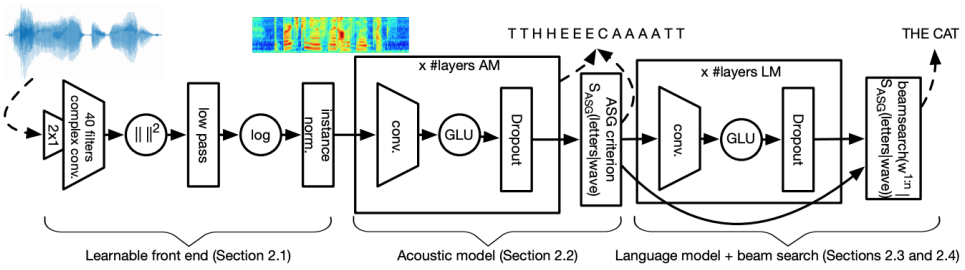


Figure 9: Architecture of the wav2letter Model, extracted from [61]

This paper also introduced Auto Segmentation Criterion (ASG), an alternative algorithm to the more commonly used CTC, that improves upon in 2 ways:

- There is no special indicative blank label;
- Gives the ability to learn the likelihood of transitions between letters.

To do this first step, for repeated letters, ASG includes a '2' for repeated letters, instead of the blank token. For example in the case of 'hello' would become 'hel2o'. By removing this special token, ASG significantly simplifies the graph that the algorithm must search when generating alignments, and subsequently improving performance. For the second point, instead of in each frame, the probability of each letter being normalized with the probability of the other letters in that frame, ASG contains its own weight matrix that models possible transitions between each letter. In real language, certain sequences are much more likely than others. For example in Portuguese, the sequence 'çã' is more common than 'ço'. This likelihood of certain combinations of letters, called 'transitions', could improve model accuracy.

Despite Wav2letter effort to run the model with almost any sort of audio input representation, their experiments also showed that MFCCs slightly improved the performance when comparing it with its counterparts, spectrum, however, this gap in performance should vanish with enough data [60].

Deep Speech

In 2014, a ASR end-to-end model, by the name of Deep Speech, was announced by Baidu AI Labs [62]. This state-of-the-art speech recognition system was developed using end-to-end deep learning and is significantly simpler than traditional ASR systems, relieving the user of laboriously engineered processing pipelines. Furthermore, it simplified the design by not needing specific components to model background noise, reverberation, or speaker variation, but instead directly learns a function that is robust to such effects, producing improvements over the results presented by traditional models on noisy environments. It does not make use of a Lexicon/Pronunciation Model (phoneme dictionary), but instead does recognition based on character and relies on a well-optimized RNN training system that has the ability to use multiple GPUs, as well the need to obtain a large amount of varied data for training.

Deep Speech acoustic model, which its core is based, on RNN, consists of five layers, first of which is a TCN, followed by 2 regular feed forwards layers like ANNs, then comes the Bidirectional Recurrent Layer (BRNN), followed finally by one more non-recurrent output layer, but this takes account of both forward and backward units coming of the BRNN, trained finally with CTC, coupled with a simple beam search decoder.

CTC loss function is used to map the variable length audio input to variable length output; however, in contrast with past attempts to create end-to-end speech recognition models involving CTC-RNN models that in need for a lexicon, such as a large list of vocab, and some pre-training using GMM-HMM system, Deep Speech trains this CTCRNN network from scratch.

3.3 Speech Recognition Parameters

Speech Recognition Systems can be categorized into different groups depending on the constraints imposed and the nature of the input speech [63].

3.3.1 *Structures of Speech*

ASR systems can be classified based on the type of utterances it is able to recognize. These utterances can be composed of a single word, up to a continuous large number of sentences.

Isolated words - These types of recognizers require that utterances are very short (1-2 words long) and surrounded by a quiet interval (single utterance at a time). These are comparatively simpler to implement and tend to have good results since the words are usually clearly pronounced, although the number of use cases is very limited. This type of system is mostly used on a 'command and control' type applications, where the utterance is recognized from a small pool of commands, and appropriate action is

done. These systems have a kind of "On/Off" states, requiring that the speaker waits between utterances;

CONNECTED WORDS - Connected word recognizers are very similar to Isolated Words, the difference is that these allow separate utterances to be chained together, with a minimal interval in between them. This allows users user to speak fluent speech consisting of words from a specified vocabulary, for example, telephone numbers;

CONTINUOUS SPEECH - This type of speech is derived from reading texts (books, newspapers, news broadcasts) or rehearsed speech. These type of systems must utilize special methods to determine utterance boundaries allowing users to speak in a semi-natural way;

SPONTANEOUS SPEECH - In general, Spontaneous Speech is considered the most natural-sounding and different and more complex than what happens when a text is rehearsed or simply read. These type of systems need to be able to handle slight stutters, sounds like "ums" and "ahs" while the speaker is thinking, slight mispronunciations, false-starts and even moments where adjacent words are pronounced together (with no distinct division in between them).

3.3.2 *Speaker Dependence*

Every individual has unique voices, due to their distinguishing characteristics such as pitch and timbre caused by their specific physical body and personality. Speech Recognition systems can be split into 2 main different groups, speaker-dependent, and speaker-independent. In the first case, the system is trained to recognize the voice of a single speaker. These are generally more precise when recognizing utterances made by a specific speaker, but to a detriment, these are much less accurate for any other speaker. These systems are generally easier to develop and have better accuracy on there specific purpose but are less tolerant to changes.

On the other hand, speaker-independent systems are more capable of recognizing utterances made by different users, even if these have not participated in the training part of the system. For this to happen, speaker-independent systems usually need a much larger training corpus as well a big variety of speaker in it, making them harder to be created and in the end usually less accurate although more flexible and tolerant.

It is possible to devise a third group, where the system is developed to adapt its operation using the best-suited speaker data to recognize the speech and increase accuracy [64, 65].

3.3.3 *Types of Speech Data*

The structure of the speech data can vary greatly depending on the ASR system's purpose. These structures can differ from simple isolated words to more 'natural' and spontaneous multi-user conversations. The less control over the speech material can oppose a challenge for its own recognition. Some common speech structures in corpus are [66]:

READ ALOUD ISOLATED WORDS - The speaker is usually asked to read aloud a list of words. This pronunciation can vary greatly from how it would be pronounced in a spontaneous conversation for a number of reasons: The user is usually much more careful in his pronunciation, the phenomenon is known as "spelling pronunciation" is a normal occurrence, and furthermore, surrounding words can influence pronunciation by introducing a context (homologous words) or anomalies;

READ ALOUD TEXT FRAGMENTS - More natural speech material can be obtained when asking the speaker to read entire text fragments. This can vary in length between short sentences to multiple sentences taken from, for instance, a book;

SEMI-SPONTANEOUS SPEECH - The speakers are tasked to read the number or alpha-numerical expressions and have freedom on how to pronounce these expressions, telephone numbers are a prime example for having multiple ways to be read (a string of digits, strings of numbers containing multiple digits);

SPONTANEOUS SPEECH - Speakers are allowed to freely choose their own words and their own subject of conversation, this is especially important in a dialogue situation. In order to keep some control over the speech material, the experimenter can predetermine a subject for the speaker(s) to talk about; Although a speech situation with two or more people is more natural than a monologue, overlapping acoustic material may result in several people speaking simultaneously.

3.3.4 *Vocabulary*

The size of the vocabulary of a Speech Recognition system largely influences the complexity, as well as the accuracy of the system. A smaller vocabulary usually means a more accurate and more easily created system. Vocabulary sizes can vary from Small (e.g., digit only) to very-large vocabulary up to tens of thousands of words.

Depending on the system, is possible to correctly recognize words that are Out of Vocabulary. These are unknown words that appear in the testing speech but do not appear in the recognition vocabulary usually being crucial such as names and locations, however, many speech recognition systems are closed-vocabulary recognizers meaning that they only recognize words in a fixed finite vocabulary, missing recognizing them as in-vocabulary words

and affecting the recognition accuracy of their surrounding words. This is an interesting problem because simply increasing the vocabulary size in a recognizer cannot resolve the problem, several alternative approaches have been proposed [67].

3.4 AUDIO CONFIGURATION

Audio Encoding and File Format

An audio encoding refers to the way in which audio data is stored and transmitted. It is important to note that an audio format is not always equivalent to audio encoding. In the case of the file format .WAV, defines the header format of an audio file, but not the audio encoding by itself. WAV will often use linear PCM encoding, but we can not assume it until the header is inspected. In other cases like FLAC, it represents both a file format as well as an encoding, which can lead to confusion.

In general terms, audio files consist of two parts, a header and the audio data itself. In the header or metadata section, as the name suggests, "data that provides information about other data" is saved. The data stored can vary from the file format, but generally contains the information about Sample Rate, Bit Depth, Encoding, as well as track information like Artist, Album, and Genre that is important when dealing with music files.

The second section of the file contains the data of the audio itself, and it can either be uncompressed or compressed. A compressed audio stream has the benefit of requiring less space for it to be stored but generally require additional processing to reduce the file size, and this compression can be either lossless or lossy. Lossless compression can be 'undone' to restore the digital data to its original format, but in lossy compression, some data is removed, meaning that such information is lost during compression and affect negatively the accuracy of the speech recognition.

Mono/Stereo Audio

Mono uses a single audio channel, to represent sound as heard as if it was coming from only one position. Stereo is the representation of sound using more than one independent audio channels, usually only used two. This is used to create the impression of sound heard from various directions, as in natural hearing. Stereo sound has replaced mono almost completely in most aspects of our life, from the movies and television we watch to the music we hear. In contrary to this, in speech recognition mono data is the most commonly used, although there have been studies on the use of stereo training data [68].

It is also possible to use different audio channels for each speaker present on the recording, this is a common technique used in interviews and telephone data. Because we did not have

any data in this specific format, all of the stereo data was down-mixed, where both channels of the stereo stream were mixed into one only stream.

Sample Rates and Bandwidth

In the real world, the sound is characterized as an analog waveform. In digital audio, we approximate this analog wave by extracting its amplitude at a rate. In digital audio, the sample rate specifies the number of samples to take from an audio source material per second. The higher the sample rate, the better we can faithfully represent higher and higher frequencies, but at the same time increasing the amount of storage needed to store the audio file.

From the Nyquist-Shannon sampling theorem, the general approach is to sample at least twice the highest frequency of the sound wave you want to store digitally. The range of human hearing is around 20-20000 Hz, which means, to digitally represent audio in this range, a sample rate of at least 40000 times per second is needed, which is part of the reason why CD audio uses a sample rate of 44100 Hz.

Luckily for us, human speech exists only in a much smaller range from around 85-8000 Hz, for this reason for Speech Recognition it is common to use a sample rate of 16000 Hz, values lower may reduce the speech recognition accuracy, and higher levels have no discernible effect on speech recognition quality.

The model developed was trained with audio with a sample rate of 16000 Hz, any audio in a different sample rate was re-sampled.

Bit Depth

As previously stated, the sound is characterized as an analog waveform. Bit depth affects the dynamic range of an audio sample. A higher bit depth allows a better representation of different audio amplitudes and reduces the signal to noise ratio within audio samples. Music is commonly provided using 16-bit depth, but a higher value of 24 bits is used in DVD Audio. For speech recognition, it is common to use 16-bit depth (although some telephone data usually used 8-bit depth), and any audio sample on a different format, were converted.

3.5 System Performance Evaluation

Over the past few years, an increasing amount of speech recognition systems have been developed creating constant competition to decide which one is the best one on the market. One important aspect is that there are no 'one-size-fits-all' best models. A model can have incredible results in clean environments but have relatively bad results in noisy environments. Also, different models can have different features that are required for a certain project, so

it is important to identify the use case and requirements for the task in question. One of the invariable important aspects of the evaluation of Speech Recognition performance is its accuracy, normally represented by WER. The formula for the calculation of this metric is as shown in Figure 10.

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

S : *Number of Substitutions*
D : *Number of Deletions*
I : *Number of Insertions*
C : *Number of Correct Words*
N : *Number of Words in the Reference (N=S+D+C)*

Figure 10: WER Formula

The WER is derived from the Levenshtein Distance (capitalization is commonly ignored) working at the word level, it is both valuable when comparing different systems as well as for evaluating improvements within one system, although it provides no details on the nature of translation errors and further work is therefore required to identify the source(s) of error.

Alternatively, we can also apply Levenshtein Distance at a Character and Sentence Level, producing Character Error Rate (CER) and Sentence Error Rate (SER) respectively. CER can provide additional information of the error nature and SER can be an important factor especially when dealing with data input (IDs, Dates, Numbers) where is majorly important to have the whole sentence transcribed correctly. Examples of WER and CER values achievable can be seen on the following examples:

$$Ref : \textit{A casa que os Maias vieram habitar em Lisboa}$$

$$Hyp : \textit{A casa que os Maias vieram habitar em Lisboa}$$

$$WER : 0\% \, CER : 0\%$$

$$Ref : \textit{A casa que os Maias vieram habitar em Lisboa}$$

$$Hyp : \textit{A casa que os Saias vieram tentar em Lisboa}$$

$$WER : 22\% \, CER : 11\%$$

A number of other characteristics may have an impact on what systems fit better in our case, aspects like:

TRANSCRIPTION SPEED - The amount of time it takes for a sentence/command to be transcribed can have a big impact on the usability of the system, depending on our

application. For example, transcription speed is commonly a major factor when SR is used to control robots in real-time, such as in [69, 70];

ONLINE DECODING - Online Decoding or Streaming, defines the ability for the system to accept streams of signals in real-time, in contrast with an offline decoding model which needs the complete audio signal put together for it to be able to process it. For example, this is a "must-have" feature when using ASR to help people who are Deaf or who have hearing loss better comprehend a live show/program.

3.6 SUMMARY

Speech recognition, also known as ASR, is a capability that enables a program to process human speech into a written format. For more than 60 years, researchers have been developing systems to enable human-machine communication over voice. From the first machines with only the ability to recognize a reduced number of different words, to modern deep learning with performances reaching the human levels of recognition, in various languages, significant milestones have been achieved.

While speech technology had a limited vocabulary in the early days, it is utilized in a wide number of industries today, such as automotive, technology, and healthcare. Its adoption has only continued to accelerate in recent years due to advancements in deep learning and big data [71].

Although all models have the same general purpose, different parameters like Structures of Speech, Speaker Dependence, Types of Speech, Vocabulary, and Audio configuration differentiate their specific objective. Depending on our goals, different models can be better suited for our needs, but generally, model performance is evaluated in both CER and WER.

4

SPEECH RECOGNITION MODELS

4.1 ASR SYSTEM STRUCTURES

Over more than 60 years of ASR development, many iterations and variations of ASR systems have been created. In such a large spectrum of time, we were bound to have different approaches to solve ASR. In this section, we will present the base structure of ASR models, but, because of all these existing model alternatives, some of these concepts may not apply completely to every model. The ASR systems architectures are commonly divided into different sections, as seen on Figure 11.

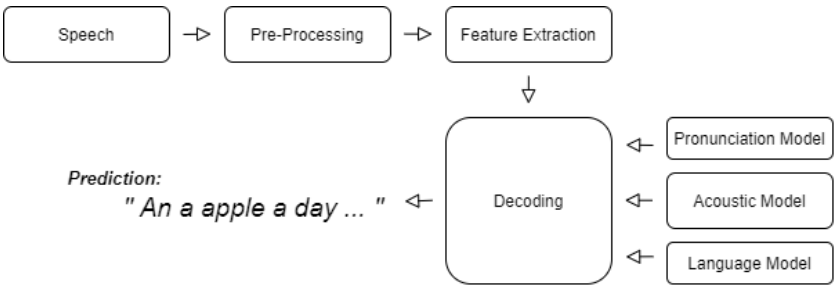


Figure 11: Traditional Hybrid Models Diagram

PRE-PROCESSING: Take the original audio and use a variety of methods to try to increase its overall quality, and subsequently increasing the accuracy of the results of the model. It is also common in modern systems to use a variety of data augmentation techniques, which will be discussed further on Chapter 6;

FEATURE EXTRACTION: Convert speech (waveform) to some representation. This representation needs to be reliable for several circumstances of the speech signal, potential environmental conditions, and speaker variations while retaining the portion that characterizes the information in the speech signal;

ACOUSTIC MODEL: Used to model the mapping between speech input and feature sequence (typically a phonemes or letters);

PRONUNCIATION MODEL: Achieve a mapping between the previously achieved features on the acoustic model to graphemes;

LANGUAGE MODEL: Maps this graphemes into a sequence of fluent words as the final transcription.

4.1.1 *Pre-Processing*

Pre-processing of speech signals can be considered a crucial step in the development of a robust and efficient speech or speaker recognition system. [72] Under optimal conditions, state of the art ASR systems have very high levels of accuracy, but these conditions are not possible in every situation. A noisy background, a considerable distance between the talker and microphone, or a echo intense environment are factors that can oppose a challenge during recognition. In this first step, many techniques can be applied to seek better and more reliable results, by reducing the effects of some of those not so optimal conditions:

AMBIENT/BACKGROUND NOISE REDUCTION/REMOVAL: In real-world scenarios, the audio data collected contains not only speech but background present in the recording. ASR models need to be able to distinguish between speech and ambient noise (wind, cars, ...) to achieve good accuracy. One of the most straightforward methods to reduce the effects of background noise is to simply remove/reduce it before it is passed on to the remaining system [73];

SPEECH DETECTION: It is common in real-world scenarios that not all audio data present is speech. To get better accuracy and improve recognition speed, we want to be able to detect the presence or absence of speech and differentiates speech from non-speech, and remove these last sections [64];

ECHO CANCELLATION/DEREVERBERATION: A sound source located a distance from a microphone will have its direct path signal arrive at the microphone and then reflections of the original signal will be captured. As the distance from the source to the microphone increases, the signal to reverberation ratio does too, and as a result, the effect of reverberation will also increase. Various methods can be used to mitigate these effects on the audio [74];

AUDIO CUT/PACKET LOSS CONCEALMENT : When a voice signal is sent over as VoIP packets over the internet, the packets may (and likely will) travel different routes. A packet therefore might arrive very late, be corrupted, or simply can not arrive at all. In VoIP connections, error-control techniques such as automatic repeat request (ARQ) are not feasible. PLC is a group of algorithms to reduce the effect of the loss of voice packets [75];

AUTOMATIC GAIN CONTROL: In environments where the input speech level is expected to vary significantly, AGC can be used to keep a constant recording volume regardless of the input signal level. The gain by which to achieve this constant recording level is determined by monitoring the voice activity state and the peak levels of the input signal [76].

Additionally in this section, it is common in the modern system to apply a number of Data Augmentation Techniques during training, this will be further discussed on Chapter 5. It is also important to note that, depending on the original dataset and purpose of the system, this pre-processing step can in some cases be skipped.

4.1.2 *Feature Extraction*

In ASR, Feature Extraction is done by transforming the speech from its waveform to a relatively and denser representation for later processing and analysis. Doing this we can illustrate a speech signal by a predetermined number of components of the signal, in a much smaller format [77]. To extract audio features, it is common to use sliding windows of width 20ms and 10ms apart to parse the audio waveform. For each of these sliding windows, we extract a frame of audio signals and apply Fourier Transform. These exact first steps were done to generate the previously shown spectrograms in Chapter 2. Such values are chosen so we can capture the dynamic frames and the proper context. Then depending on the method a number of features are extracted. As stated before, the way people sound and their voice varies between person to person. Because of this it is of special importance to extract features that will be robust enough to be independent of who the speaker is and to possible noises present in the environment. Also, like it is commonly done in other Machine Learning problems, we want extracted features to be independent of others.

There is a number of speech feature extracting methods including: Mel Frequency Cepstral Coefficients (MFCC), Linear Prediction Coefficients (LPC), Linear Prediction Cepstral Coefficients (LPCC), Line Spectral Frequencies (LSF), Discrete Wavelet Transform (DWT) and Perceptual Linear Prediction (PLP) [77]. Of these, from our research, we found that the MFCC is possibly the most commonly used, generating a total of 39 different features. These previously mentioned approaches have been used and tested in a variety of use cases, and many variations of them have been developed with the intention of creating Feature Extracting mechanisms that are less susceptible to noise and are faster [77].

A newer approach that appeared with the use of DNN in ASR, and the approach used in our case, is to not use any of this predefined extraction methods and simply feed the short-time Fourier transform (STFT) of the signal to the model into a deep convolutional network at the beginning of the model. These technique used in State of the Art Systems is able to achieve a better representation of the audio for the purpose of recognition. Alternatively to

feature extraction, there have also been several attempts to train ASR systems with raw wave signals [78, 79], and perhaps most notability Facebook's W2L approach, but the results show that we are still in a stage where this method produces worst results than what is commonly used alternatives [61].

4.1.3 *Acoustic Model*

Acoustic Models can be considered the main part of ASR systems. These models contain statistical representations of each of the distinct sounds that makes up a word and are used to represent the relations between audio signal and phonemes or other linguistic units that make up speech. This model is created from a large dataset of audio recordings and their corresponding transcripts by using training algorithms to create statistical representations for each sound in a language. The speech decoder listens for the each distinct sounds spoken by the user and then returns the closest matching representations of the speech.

Classical Models

As shown in the previous chapter, ASR as been for many years a subject of interest for many researches. In such a large spectrum of time, we were bound to have different approaches to solve ASR. At the moment we are living in a "Deep Learning Era", where models using DNNs are achieving unprecedented accuracy but for a many years, GMM and HMM were the two must-learn technology for speech recognition. Although we now have more and newer options, where some newer systems are completely free of these approaches, this type of technologies is not rendered useless and there are multiple hybrid systems that combine both HMM and Deep Learning.

ASR decoding is mainly composed of two major steps: the mapping and the searching. In the mapping, we map the acoustic information of an audio frame to a phone. A GMM is responsible for emitting the probabilities of a phone given the acoustic vectors to an HMM model, which is responsible for modeling the possible transitions at each time step and allows the occurrence of multiple pronunciations. Then, we search for the phone sequence for the optimal word sequence. But this is a complicated process since we have to take into account alignments of acoustic information to a phonemes and account for multiple paths that produce the same results.

For over 30 years, the speech recognition area has not seen much improvement [80]. But in 2012, a major breakthrough is credited to Geoffrey Hinton [81], where it is proposed to replace the GMM by a DNN, and in doing this, they improved the state-of-the-art model performance by over 30%. Following recent trends in Deep Learning Applications for computer vision, where deep models have outperformed traditional machine learning algorithms, the end-to-end area for speech recognition has been rising in popularity over

the years. An end-to-end model is a system where much of the ASR problem pipeline is replaced by a DNN architecture. With a single model, the parameters and features learned are solely tuned by the training algorithm to increase the accuracy rate of the system [80].

End-to-End Models

Deep Learning, opens the door for an alignment-free one-to-one mapping which maps an audio frame to a relatively high-level component, and as before, we can then search for it. This gives us a head start and bypasses the complicated alignment process. This is the core concept of the popular paper **[CTC]**, released in 2006, proposing Connectionist Temporal Classification (CTC). While in early research, the deep network works with phonemes, now the deep network works with character sequences. This frees us further from the Lexicon/Pronunciation Model that follows. Since then, the CTC method has been extensively used in end-to-end speech recognition systems.

The deep network generates a probability distribution for all characters. We no longer just greedily pick the most likely character in each time step. There are multiple paths that represent the same word, and we need to sum over them to find the optimal paths. Since there are possible pauses during the speech, within or between words, we also need to introduce empty tokens (lets use 'ε' as an example) to model such behavior.

Given the output "gaaaεεtεεo", we then apply the CTC compression rule, and any repeated characters will be merged into one, and a final step to remove all 'ε' occurrences from the final word. This 'ε' character is also useful in the case of words that contain repeated letters one after the other, for example the word 'voo', but the model needs to be able to predict the 'ε' between the two 'o' characters. Because we have multiple ways of representing the same word, for instance ["gεatεo", "gaaεaεεtεεo", "gaaεεtεεooo", "ggεaεεto",...] all represent "gato", to find the most likely word sequence, we need to search and sum the probability of all different path combinations.

More recently, attention-based recurrent networks have been successfully applied to speech recognition [82]. Those RNNs are based on the Encoder-Decoder/Sequence to Sequence (Seq2Seq) architecture, that has been used on a variety of Deep Learning problems. The encoder is typically an RNN that transforms the input into an intermediate representation. The decoder, also typically an RNN, uses this representation to output the desired sequences. The attention mechanism acts by selecting relevant content from the input at every time step, thus helping the decoder. Our focus on describing the CTC process, is because it is the method used on our ASR system.

4.1.4 *Lexicon/Pronunciation Model*

Pronunciation Models represents the likelihood of a sequence of phonemes (or other linguistic units that make up speech) given the existing words in a language. These provide the link between the statistical representations in the Acoustic Model and the words present in the Language Model. This data is typically written by humans (or at least supervised) and it contains the representation of the pronunciation(s) for all words contained in the training corpus. The OOV rate represents the percentage of word tokens in the test data that are not contained in the pronunciation model, this value should be minimal, since each OOV word may result in errors. Different languages can have a lot of variations in their morphological richness, this has a significant effect on the size of the constructed model. The same word in a language may have multiple pronunciations, this can be cause by a number of different reasons:

- The same word can have multiple pronunciation in different dialects (EP and BP). The same can happen when dealing with non-native speakers;
- In some languages the surrounding words can highly affect the pronunciation of the start and end of certain words (in Portuguese: "... da Água");
- Linguistic Deviations can be common and how you pronounce certain words can be common (in Portuguese: "dezoito").

This could signify that Pronunciation Model should have multiple pronunciation for each word, but from our research, large vocabulary systems usually contained only one pronunciations per word. Adding more pronunciation variants can give more flexibility to the model, but at the same time, increasing the number of ambiguities and in end, reducing accuracy. As stated before, in state of the art End-to-End systems, it is possible to skip this step by outputting character sequences instead of phonemes, and the inferences provided by this model that helps output words that are well formatted (correctly written), are integrated in the next step, the Language Model.

4.1.5 *Language Model*

Language Models represent the likelihood of a sequence of words. By assuming that the utterance is both grammatically correct and that it "makes sense", language models can provide context and help distinguish between words that sound similar to the previous steps. This if more easily understood with an example: Lets imagine that our previous models gave these 2 phrases very similar probability of being the correct utterance spoken in a certain audio clip:

"Vou à padaria comprar pão" and *"Vou à padaria comprar cão"*

If you know Portuguese, your instincts will tell you that the first phrase is much more likely the correct one, even if you have not listened to the audio in question, for two reasons:

1. The first phrase is grammatically corrected as opposed to the second one;
2. Given the context, the first phrase is much more likely to exist than the second one. We usually go to a bakery to buy bread, not dogs.

This simple example shows the type of the decision made by the Language Models, this being usually built around n-grams. This types of models are commonly generated by tremendous amounts of text data (in comparison with the amount of audio data used on the Acoustic Model) and are especially fruitful at improving the system accuracy if they are trained to deal with narrow subjects (contexts with smaller vocabularies). Although we can have giant amounts of training audio data for our LM, sequences of words (that make sense and are grammatically correct) will still not be observed in the training set, and it is important for this sequences of words to not appear as impossible (0% probability of existing) just because they were not present in training.

N-gram Language Models

In a lot of natural language processing tasks, a Language Model can be defined as a probability of a sequence of words, or characters. In the case of words, a LM can be viewed as a way to observe language grammar.

A n-gram language model is modeled based on specific 'n' number of words. By parsing a larger amount of text, we are able to create approximations on the probabilities of the adjacency of existing words in that text file. In its simplest form, a 2-gram, the probability of a word is calculated based solely on the probably of it occurring right after the previous word (this probability is created by observing text data during training).

In this simple case the probability of a phrase is as follows:

$$P(\text{"}O\ poeta\ é\ um\ fingidor\text{"}) =$$
$$P(\text{"}O\text{"}) * P(\text{"}poeta\text{"}, \text{"}O\text{"}) * P(\text{"}é\text{"}, \text{"}poeta\text{"}) * P(\text{"}um\text{"}, \text{"}é\text{"}) * P(\text{"}fingidor\text{"}, \text{"}um\text{"})$$

$$Where\ P(\text{"}A\text{"}, \text{"}B\text{"}) = count(\text{"}BA\text{"})/count(\text{"}B\text{"})$$

By increasing the 'n' value, model takes in consideration the 'n-1' words that came previously on the sentence of the word we want to find out. This can provide a significant increase in accuracy but at the cost of also increasing the size of the model and consequently the time it takes for us to find the most probable sequence.

One downside of this basic form of a n-gram LM is that there is a possibility that the training corpus does not contain every legitimate word combination (this gets worse when the 'n' value and the vocabulary is increased). If a valid sentence does not show up in corpus, it has a probability of zero; however, there are multiple techniques smooth this results, but these will not be discussed in this work.

Beam Search

In the context of ASR, decoding the most likely output sequence based on a LM would likely increase the accuracy of the model; however, this involves searching through all possible output sequences (even the ones that did not appear during training). This creates an exponential problem with the length of the sequence we observing during search (n-1). This is perfectly doable with smaller 'n' values and reduced vocabulary, but quickly turns impracticable when increasing these numbers, hence there exists less accurate solutions to this problem, but at the same time much quicker, such as decoding using Greedy or Beam Search.

In Greedy Search decoding, we only take in consideration the highest scoring element at each stage, meaning selecting only the highest likely word at each step in the sequence. As expected this greatly increases speed but can take a toll in model accuracy, possibly reducing the overall accuracy of the model in general. A more accurate, although slightly less efficient, alternative algorithm, is called Beam Searching, as seen on Figure 12.

In Beam Search decoding, we take in consideration an expanding k most possible next steps. When 'k' = 1, beam search decoder acts in the exact same way as the greedy counterpart; however when increasing 'k', at the cost of a small increase in time, accuracy can be largely improved.

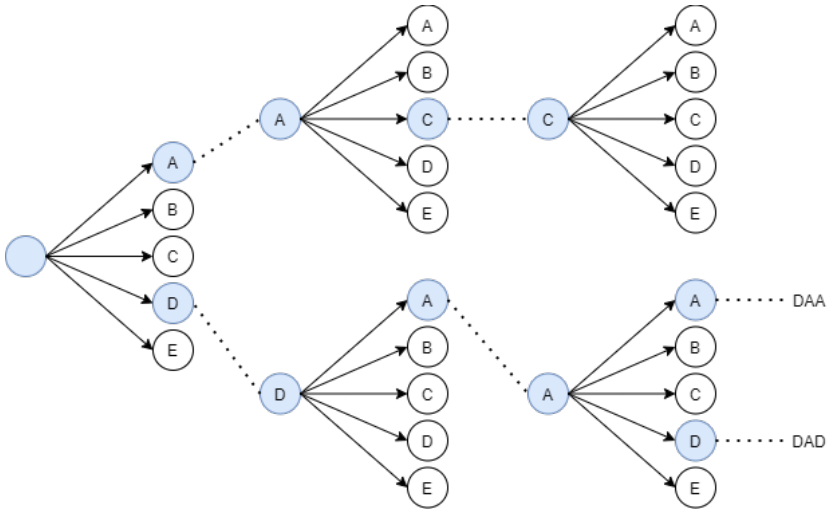


Figure 12: Beam Search Using K=2 and N=3

In this last example, each node corresponds to a word, and we are able to find that the 2 most likely sequences are 'DAA' and 'DAD'. With these, we are then able choose of the two, the one with the most probability of occurring.

4.2 Classical Models vs End-to-end Models

As stated before, during a long period of time, the classical SR approach HMM-GMM was the best method for the development of ASR systems. However, each of these models that created the system, are trained independently with different objectives. For this reason, errors that occur in each component may have very negative impacts on another subsequent component. But researchers started evaluating that each of the previous described components, could possibly work more effectively with the use of one single neural network, and that is the principal motivation for creating a process where train is done in the entire model as one component itself. These are called end-to-end models, and envelop multiple components in the pipeline discussed above.

The HMM-GMM previously used in acoustic models, are replaced by a single neural network that output characters instead of phonemes, removing the need for a pronunciation model, of this are common the CTC and Seq2Seq methods. Feature Extraction can also be done by this unique model, by having convolutional neural networks at the beginning of the model, creating a good representation of the audio signal provided to the system. One example of this type of system is the one used in this work, namely the deepspeech.pytorch [83].

5

MODEL PREPARATION

5.1 DATA AUGMENTATION

Deep Learning has in recent years gotten remarkably good results in a multitude of tasks, from Computer Vision [84] to Natural Language Processing [85]. But these recent advances have been largely attributed to the quantity and diversity of data gathered [86].

When training machine learning models, we are in reality tuning different parameters such that the model can map an input (in our case an audio file) to an output (in our case a string of the transcription). If we are using a Deep Learning Model to perform a complex task, a lot of parameters need to be tuned and a proportionally big amount of examples need to be shown to the model to achieve good performance. But unfortunately, many application domains where large data-sets are crucial for good performance, do not have access to substantial amounts of data, like it is in our case.

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data [86]. One of the key subjects where these types of techniques have been studied and developed for many years, and one of the easiest to help us understand these concepts, is Computer Vision.

Figure 13: Original Image
extracted from [87]

From the original image in Figure 13, different augmentation techniques can be applied such as Noise Injection, Hue/Saturation Changes, Cropping + Padding, Flipping, Rotating,

Blurring, Dropout of Regions, and Change of Perspective. By applying one or more of these techniques, we can generate a multitude of new unique, and distinct images. These new data with minor alterations of the original can vastly increase the dataset size and diversity, as can be seen on Figure 14.

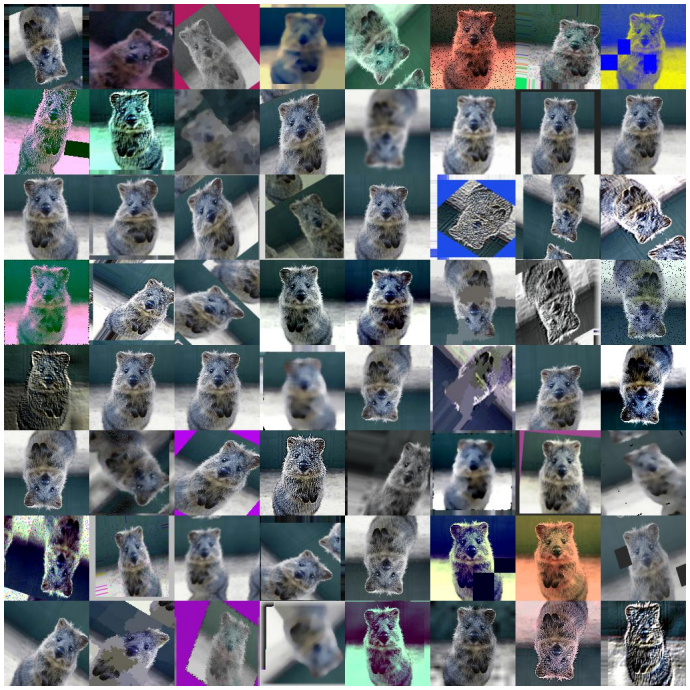


Figure 14: Examples Results of Different Augmentations applied to the Original Image, extracted from [87]

In the case of speech recognition, augmentation traditionally involves deforming the audio waveform used for training in some fashion or adding background noise [88]. The use of this new enhanced data can improve the robustness of the model by making it more 'invariant' to certain conditions like illumination or rotation. Using these type of techniques is especially useful when our training data has a limited set of different conditions and the application data can have a variety of other conditions. Data Augmentation techniques can also be useful if we have large amounts of data. Generalizing our training data can help reduce the number of irrelevant patterns, resolving the class imbalance problems in classification tasks, and increasing the overall performance.

Data augmentation is performed before the data is fed into the model, but two different options of when to apply these techniques are available:

PRE-PROCESSED AUGMENTATION: In this method, all necessary transformations are done before the 'training' of the model commences, meaning the dataset size is increased by a factor equal to the number of transformations applied to the data. This type of

augmentation is normally done on small datasets, where storing such amounts of augmented data leads are possible;

REAL-TIME AUGMENTATION: In this method, the original training data stays the same, but for every mini-batch during training, the data is augmented using the different techniques. This keeps the dataset at the same size but increases the model training time since time is needed to apply all the different techniques to the data. This type of augmentation is normally done on a large dataset where it would be impossible to store all augmented data.

Due to the large amount of space needed to store our dataset and the impossibility to store more large amounts of data in our machine, all augmentation techniques used were done using the second method. Because of this choice, it is important to take into consideration the impact of time and resources of each augmentation technique when applied to our data. For this work, a number of 'simple' and 'computationally cheap' methods were tested, and more expensive techniques such as Audio Restoration were deemed unreasonable for the conditions of this project.

5.1.1 *Noise Injection*

In the case of Speech Recognition, it is common for the publicly available dataset to contain only data that is clear of noise. Furthermore, the task of recognizing speech in a noisy environment is considerably more difficult than in a clean environment. If we combined these two facts together when applying these models using real data, the accuracy of the results is greatly reduced.

To help mitigate this issue, in noise injection, **background noise is intentionally and randomly added into the training data**, as can be seen on Figure 15, with the purpose of increase the overall robustness to interference.

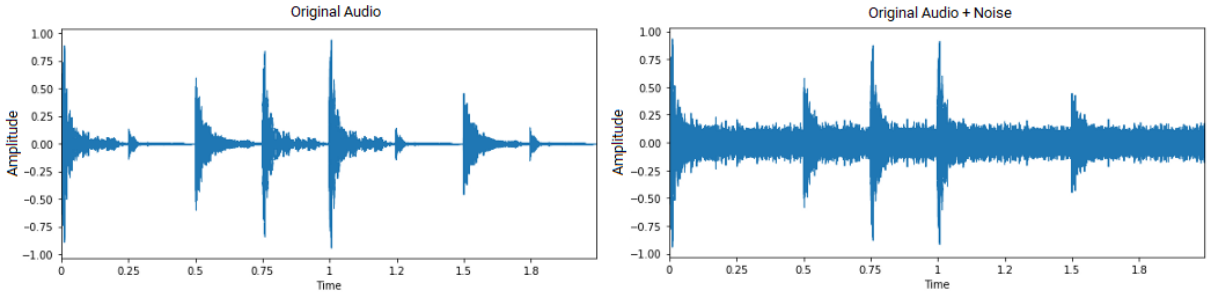


Figure 15: Noise Injection Example

The first step is to create a pool of different noises that can later be used for augmentation. In our case 3 different subgroups were created. The first pool of six different samples of noise was extracted from [89], corresponding to a set of different 'Colors of Noise':

WHITE NOISE : White noise is a signal, named by analogy to white light, with a flat frequency spectrum when plotted as a linear function of frequency (e.g., in Hz). In other words, the signal has equal power in any band of a given bandwidth (power spectral density) when the bandwidth is measured in Hz;

PINK NOISE : The frequency spectrum of pink noise is linear in logarithmic scale; it has equal power in bands that are proportionally wide;

BROWN NOISE : Noise with a power density which decreases 6 dB per octave with increasing frequency (frequency density proportional to $1/f^2$ over a frequency range excluding zero;

BLUE NOISE : Blue noise's power density increases 3 dB per octave with increasing frequency (density proportional to f) over a finite frequency range;

VIOLET NOISE : Violet noise's power density increases 6 dB per octave with increasing frequency (density proportional to f^2) over a finite frequency range;

GREY NOISE : Grey noise is random white noise subjected to a psycho-acoustic equal loudness curve (such as an inverted A-weighting curve) over a given range of frequencies, giving the listener the perception that it is equally loud at all frequencies. This is in contrast to standard white noise which has equal strength over a linear scale of frequencies but is not perceived as being equally loud due to biases in the human equal-loudness contour.

The second pool of noises was extracted from multiple YouTube videos of ambient noise. These videos ranged from simulated rain noises, coffee/store noises, street/city noises, etc., in total this pool totaled 6 hours of audio. A third and last pool containing four different samples of noise collected from real telecommunications at NOS was also created.

Noise Injection was implemented by randomly inserting noise iteratively in utterances on each epoch of training. Three parameters affect the operation of said method, 'noise_prob' (probability of noise to be inserted on a certain audio clip), 'noise_min' (minimum amount volume of noise), and 'noise_max' (maximum amount volume of noise). These last 2 values are relative to the volume of the original clip. This was done dynamically because of the restrictions on our corpus size on our machine as opposed to the more common method of creating different copies of the corpus with different variations of noise and joining them together.

Multiple experiments were done to confirm that the introduction of noise during the training approach works well for the model and can provide substantial performance improvement for speech recognition [90, 91, 92, 93, 94].

5.1.2 *Speed Perturbation*

Speed Perturbation is a method that consists of changing the speed of the audio signal, producing different versions of the original audio with slightly different tempo and pitch. This technique has a low implementation cost, making it very easy to adopt and experiment with.

The application of this method creates an audio clip that can sound unnatural compared to the original clip where it was applied. When the speed of the signal is reduced ($\alpha<1$), there is a shift in the signal energy from the high frequencies towards the lower frequencies. The opposite can be said when the speed of the signal is increased ($\alpha>1$). It is important to refer that this method is a combination of two other methods, these being Tempo Perturbation (TP) and the other Vocal Tract Length Perturbation (VTLP).

In TP, we simply shift the tempo of said audio. This changes the duration of the signal which also affects the number of frames in the utterances while maintaining the original pitch of the sound. VTLP changes the pitch of the clip and tries to emulate how different humans would say the same utterance while maintaining the speed of the clip.

The unnatural sounding utterances of SP could be resolved by using either of the other methods described, however in practice, this does not seem to cause a problem as shown in [95]. Furthermore, although speed perturbation emulates VTLP + TP, following the same paper it was found that the addition of TP to VTLP to the data was actually detrimental to the results and that individually each of these two perturbances had worst results than SP.

Following these results, it was decided to simply use TP and it was implemented by randomly choosing a warping factor from the range [0.85, 1.15], it was then applied iteratively to every utterance in each epoch of training using the speed function of the Sox audio manipulation tool. This was done because of the restrictions on our corpus size on our machine as opposed to creating multiple copies of the corpus with different warping factors and joining them together. A test using a clean subset of our corpus was done to confirm the improvements provided by this method.

5.1.3 *Volume Perturbation*

It is common that in a well-curated audio corpus, the variance in audio volume is minimal, but when dealing with real data, this is not always the case.

Volume Perturbation is a method that consists of changing the amplitude of the audio signal, producing different versions of the original audio with slightly different volumes. This technique has a low implementation cost, making it very easy to adopt and experiment with. Volume Perturbation was implemented by randomly choosing a scaling factor from the range [0.5, 1.5], it was then applied iteratively to each utterance for each epoch of training using the speed function of the Sox audio manipulation tool. This was done because of the restrictions on our corpus size on our machine as opposed to creating different copies of the corpus with different scaling factors and joining them together. A test using a clean subset of our corpus was done to confirm the improvements provided by this method.

5.1.4 *Mixed Bandwith Training*

The quality of the audio fed to the recognition models can have serious effects on their accuracy, where higher quality data usually equates to better results. For this reason, it is common for audio corpus used in speech recognition to contain better quality audio than what would be commonly encountered with real-life data.

One of the important factors to evaluate the audio quality is the bandwidth of the audio clip. These are usually split into 2 different categories, narrow-band and wide-band. Currently, there are many devices and equipment that receive both narrow-band and wide-band speech for ASR-based applications. In conventional approaches, different acoustic models are built to handle narrow and wide-band speech separately since their sampling frequencies are different (8 kHz vs 16 kHz). However, it is not very economical or efficient to collect large amounts of training data for each of the tasks. A simple solution is to down-sample the wide-band speech and treat it similar to that of the narrow-band [96]. However, wide-band has information that is useful to detect certain phonemes and is lost with down-sampling [97, 98]. Moreover, models built on narrow-band tends to perform worse on the wide-band speech [97, 96].

Since all of our training data was wide-band speech, Mixed Bandwidth Training was implemented by randomly down-sampling utterance with a change of 33%, during training the model. The files were down-sampled during each epoch of training and to do this, the Sox audio manipulation tool was used. The model is then trained with the mixture of both narrow-band and wide-band audio. A test using a clean subset of our corpus was done to confirm the improvements provided by this method.

5.1.5 *SpecAugment*

In the modern ASR models, the audio waveform is encoded as a dense visual representation, such as a spectrogram, before being input as training data for the network.

Augmentation of training data is normally applied to the waveform audio before it is converted into the spectrogram, this is the case of all the methods previously listed. SpecAugment was a method introduced by Google Brain [88], that investigated the approach of augmenting the spectrogram itself, rather than the waveform data 'by treating it as a visual problem rather than an audio one' [99]. This method does not require additional data and although simple, it has shown to be surprisingly effective in improving the performance of ASR networks.

SpecAugment reshapes the spectrogram by warping it in the time direction, masking blocks of consecutive frequency channels, and masking blocks of utterances in time. These augmentations have been shown to help the network to be robust against deformations in the time direction, partial loss of frequency information, and partial loss of small segments of speech of the input, these last ones is particularly interesting when dealing with phone/VOIP conversation data.

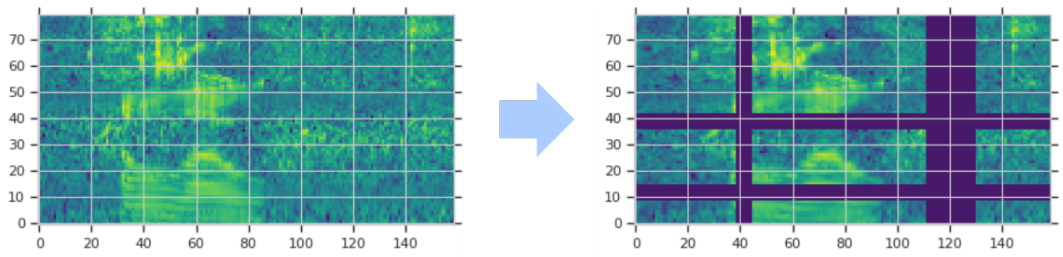


Figure 16: Spec Augment Process, extracted from [88]

The log mel spectrogram is augmented by warping in the time direction, and masking (multiple) blocks of consecutive time steps (vertical masks) and mel frequency channels (horizontal masks), like it can be seen on Figure 16. The masked portion of the spectrogram is displayed in purple for emphasis [88].

Seannaren's [83] implementation of SpecAugment algorithm was used do this project, and a test using a clean subset of our corpus was done to confirm the improvements provided by this method.

5.1.6 *Audio Encoding and Codecs Simulation*

Audio encoding and file formats both refer to the manner in which audio data is stored and transmitted, although these refer to different aspects of configuration.

The audio file format refers to the file format used to store digital audio data on a computer, commonly used formats are for example '.mp3' or '.wav'. These files can be split The data can be a raw bit stream in an audio coding format, but it is usually embedded in a

container format or an audio data format with defined storage layer. This audio file formats can be divided into 3 different types:

UNCOMPRESSED AUDIO FORMAT: In this type of audio formats, the audio is completely unaltered from its original state. It goes without saying that uncompressed files are quite large taking large amounts of disk space. One of the most common uncompressed audio format is WAV;

LOSSLESS COMPRESSED AUDIO FORMAT: A lossless compressed audio format stores data in less space without losing any information and the original uncompressed data can be recreated from the compressed version; In this type of format, many methods are used to be able to store the data in less space, most notably the ability for these formats to store silence using almost no space. The FLAC format is probably the best-known example of lossless compression. Development in lossless compression formats aims to reduce processing time while maintaining a good compression ratio;

LOSSY COMPRESSED AUDIO FORMAT: Lossy audio format enables even greater reductions in file size by removing some of the audio information and simplifying the data. This, of course, results in a reduction in audio quality, but a variety of techniques are used, mainly by exploiting psychoacoustics, to remove the parts of the sound that have the least effect on perceived quality, and to minimize the amount of audible noise added during the process; The popular MP3 format is probably the best-known example of lossy compression. Most formats offer a range of degrees of compression, generally measured in bit rate. The lower the rate, the smaller the file and the more significant the quality loss.

The conversion from a lossy audio format to a lossless does not miraculously recover data loss because of the choice of format. Additionally, when using audio for ASR, we are not always sure of what formats the audio had been stores previously, and if any previous conversion of formats caused any loss or alteration in information stored. Distorting audio during training to simulate effects of format distortion has proven to improve results provided by such models [100]. Additionally, we can also convert the commonly .WAV files stored with "16-Bit PCM" used in ASR models, to other codecs, most notably G.711 and G.722 that are commonly used in Voip and Telephone Calls. This can be used to simulate this type of data, by simply transforming any other type of files, into these formats.

5.2 TRANSFER LEARNING

Humans have an inherent ability to transfer knowledge across tasks. What we acquire as knowledge while learning about one task, and instead of learning everything from scratch,

when we attempt to learn new aspects or topics, we transfer and leverage our knowledge from what we have learned in the past [101].

In conventional machine learning, models are developed to solve a specific task, and it would be necessary to completely rebuild them from scratch when the task changes even slightly. This aspect is especially important to consider in the context of deep learning when most models need lots of data to solve complex problems, and getting vast amounts of labeled data for supervised models can be difficult, considering the time and effort it takes to both gather and label all the data points. Furthermore, it is common for deep learning models to be very specialized in a particular domain or even a specific task. While these might be lead to really high accuracy and beating all benchmarks, it would be only on very specific data-sets and end up suffering a significant loss in performance when used in a new task that would still be similar to the one it was trained for.

This forms the motivation to investigate transfer learning, which tries to leverage the knowledge (features, weights, etc.), gained from previously trained models and use it to solve new problems and even mitigate problems like having low amounts of data for the newer task, as shown on Figure 17.

Transfer learning should enable us to utilize knowledge from previously learned tasks and apply them to newer, related ones. If we have significantly more data for task T1, we may utilize its learning, and generalize this knowledge (features, weights) for task T2 (which has significantly less data). In our specific case of ASR, certain low-level concepts, what is voice vs noise and pauses, possibly some common sounds/phonemes and letters between languages, can be shared across tasks, and thus enable us to achieve better and faster results.

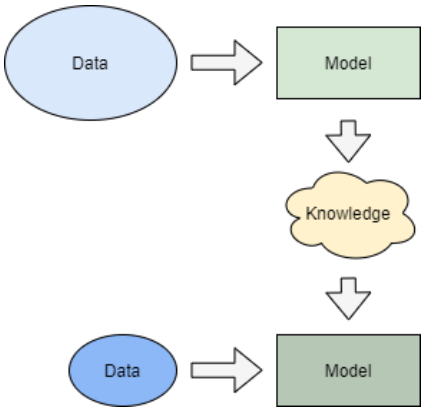


Figure 17: Transfer Learning Diagram

There are different transfer learning strategies and techniques, which can be applied based on the domain, the model type used, the task at hand, and the availability of data. Deep learning models are representative of what is also known as inductive learning. The objective of inductive-learning algorithms is to infer a mapping from a set of training examples. For

instance, in cases of classification, the model learns a mapping between input features and class labels. Deep learning systems and models are layered architectures that learn different features at different layers (hierarchical representations of layered features). These layers are then connected to the last layer to achieve the final output. The key idea is to leverage the pre-trained model's weighted layers to extract features during training with new data for the new task.

The more straightforward method of using this knowledge is called freezing. The idea is to just simply copy the pre-trained model's weighted layers to extract features but not to update the weights of the model's layers during training, and attach it to a newer final layer built for the new task, as seen in Figure 18.

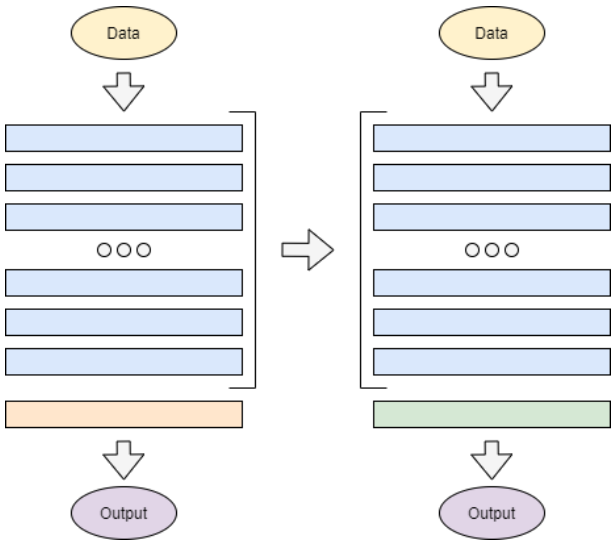


Figure 18: Freeze Transfer Learning Diagram

A more involved technique, called fine-tuning, where we do not just simply replace the final layer, but we also selectively retrain some or even possibly all the previous layers inherited by the base model, shown in Figure 19. As discussed earlier, the initial layers have been seen to capture generic features, while the later ones focus more on the specific task at hand. Using this insight, we may freeze (fix weights) certain layers while retraining, or fine-tune the rest of them to suit our needs. In this case, we utilize the knowledge in terms of the overall architecture of the network and use its states as the starting point for our retraining step. This, in turn, helps us achieve better performance with less training time.

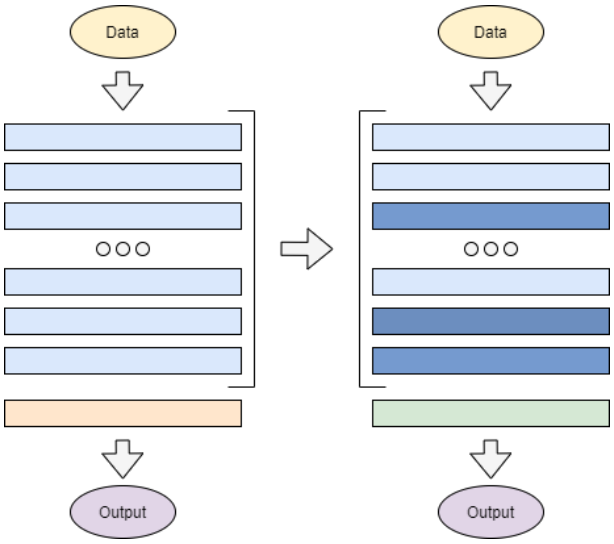


Figure 19: Fine Tuning Transfer Learning Diagram

Luckily for us, many of the state-of-the-art deep learning architectures used in ASR are open source, and we additionally are able to download pre-trained models that are usually shared in the form of the millions of parameters/weights the model achieved while being trained to a stable state. These can help solve complex real-world problems with several constraints, tackle problems like having little or almost no labeled data availability and improve the obtained results by transferring knowledge from one model to another based on domains and tasks. It is important to note that, although TL can help us create better models, there are also cases when transfer learning can lead to a drop in performance. For example, in cases where the source task is not sufficiently related to the target task, or if the transfer method could not leverage the relationship between the source and target tasks. For this reason, we will test the difference in performance in models with and without transfer learning in Chapter 7.

6

DATA AND METHODOLOGY

6.1 DATA

This section describes all data used during the realization of this project, as well as the projects from they were retrieved and the steps done to transform said data into the format used for training. The majority of the data is publicly available online, and is split between two languages, English and Portuguese.

A larger and publicly available English dataset was used on the pre-trained back-model and a smaller and newly created EP corpus was used in the development of the new model for the Portuguese Language.

6.1.1 *English Voice Datasets*

A total of 4 different English Datasets were used in the training of the back-model. The chosen are all well known and generally considered benchmarks for speech recognition tasks, these are: Librispeech [102], Tedlium v2 [103], VoxForge [104] and Common Voice [105]. As stated, all these distinct datasets were concatenated to form a much bigger corpus and then used to develop the base English model for our project.

LibriSpeech

The LibriSpeech is a corpus of approximately 1000 hours of 16kHz read English speech, prepared by Vassil Panayotov with the assistance of Daniel Povey, and it was publicly released in 2015 [102]. The data is derived from recordings of audio books from the LibriVox project [106] that have been segmented and time-aligned. At the time of writing, this corpus is considered a very significant public available data-sets for the English language.

The LibriSpeech dataset is divided into 3 main parts. A 'train' set used for training model, a 'dev' set to be used for validation, and a 'test' for testing the accuracy of the model. The 'training' dataset is divided into 3 different volumes, with approximate size 100, 360, and 500 hours respectively. All 3 subsets were used in training, totaling approximately 960 hours

of transcribed speech. The 'development' set is divided into 2 volumes, a 'clean' and 'other' set, containing respectively the lowest and the highest WER evaluated by the authors in a state-of-the-art model at the time. Both were volumes were used for validation of the model. Finally, the 'test' set is also divided into 2 volumes, a 'clean' and 'other' set, divided in the same matter as the 'dev' set, and both were used for testing the model accuracy. The different sub-set of data present in this corpus can be seen in Table 7.

Subset	Hours	Per-Speaker Minutes	Num. Female Spks	Num. Male Spks
train-clean-100	100.6	25	125	126
train-clean-360	363.6	25	439	482
train-other-500	496.7	30	564	602
dev-clean	5.4	8	20	20
dev-other	5.3	10	16	17
test-clean	5.4	8	20	20
test-other	5.1	10	17	16

Table 7: LibriSpeech Data Split

TED-LIUM v2

The 'TED-LIUM Release 2' dataset consists of a enhanced version of the original TedLium dataset [107].

The corpus consists of 207 hours of english talks (141h male / 66h female) and their transcriptions, all available on the TED website. It contains data from 1242 talks, from a total of 1495 diferent speakers, containing a total of 2.6M words. It is divided into 3 smaller sets, train, dev, and test, containing 1514, 8, and 11 talks respectively.

Voxforge

VoxForge is an open multi-language speech project initiated in 2006. It was set up to collect transcribed speech in multiple languages (17 at the moment) for use with Free and Open Source Speech Recognition Engines, and it uses crowdsourcing for data collection. All the files are submitted by users and made available under the GPL License and then 'compiled' into acoustic models.

It contains around 120 hours of speech submitted from close to 3000 different speakers, and it is available in 8 kHz and 16 kHz sample rate. All data is available in project website grouped by submission and no validation set or test set is defined at the time of writing.

Common Voice

The Common Voice is an open, multi-language dataset of transcribed speech intended for speech technology research and development [105]. Similarly to Voxforge, this project employs crowdsourcing for data collection as well as for data validation.

As of the time of writing, it contains a total of 7,226 recorded hours (where 5,671h have been validated) split between 54 different languages, making it one of the largest audio corpus in the public domain for Speech Recognition. In the English Language, it contains around 1500h of validated transcribed speech from around 61,000 different users, split into 3 smaller sets, train, dev and test.

6.1.2 *Portuguese Voice Datasets*

In the creation of an ASR, many hours of transcribed speech are needed to achieve good results. Unfortunately, there is not nearly as much available open data for the Portuguese Language compared to English. It is also common for these corpus to not distinguish between different Portuguese dialects. Because of these two reasons, a corpus of transcribed audio was created by combining multiple online resources with attention to filtering out any other dialect that is not EP.

VoxForge

As previously said, VoxForge is a multi-language speech project that aims to collecting transcribed speech in a variety of different languages with the use of crowd-sourcing. In Portuguese, there are a total of 407 submissions from users, but unfortunately VoxForge makes no distinction between the different Portuguese Accents. For this reason, audio clips from all different user's submission were carefully heard and classified in one of two groups, either EP or not. In the set a total of 384 submissions in BP and 23 in EP are present adding up to a total of around 250 minutes of speech, 236 minutes in Brazilian/Other Portuguese Dialects and only 14 minutes in EP.

Because of the aim of this project, we will be focusing only on the EP partition being the dialect we are working for. The distribution of the submissions by Age and Gender can be seen on Table 8.

Age / Gender	Male	Female
Teen	5	0
Adult	14	4

Table 8: LibiSpeech Data Distribution

Each submission file manly consists of small collection of audio files (utterances spoken by the same user) and a file containing all transcriptions and the corresponding path for the audio recording. Each submission was extracted and the format was simply adapted to fit the one used by ASR tools. Because of the reduced size of this corpus, no validation and test split were done, and all data was used for training the models.

Librivox + Gutenberg

A corpus created from both Librivox and Gutenberg projects, was at the beginning of being created, in a reproduction of what was previously done in the creation of the LibriSpeech corpus, but this time for EP.

The creation of this corpus represents a lot of hours since after listening for each segment, downloading each separate file (audio and text), pre-processing is done on the audio segments and text files to remove all metadata present. After having both files prepared, forced alignments are used to split them into segments and syncing each text file to the corresponding audio file. The last step, and most time consuming, is validating and fine-tuning each sentence end and start time stamps since the automated syncing does not always produce 'perfect' results. Because of this long and time-consuming process, the creation of this corpus was postponed to a later date and unfortunately was never finished for this project.

Common Voice

As said before, the Common Voice is an open-source, multi-language dataset of voices, and at the time of writing it containing at the time of writing 48 hours of validated speech in Portuguese.

Similarly to the VoxForge project, the corpus makes no differentiation between European, Brazilian, or any other dialects of Portuguese Language. The same initial treatment was done to this corpus: segments of all 715 different users with verified audio were listened to, and only submissions from users with seemingly EP accent were used. In the end, only audio from 64 users were used for a total of 2 hours of validated transcribed speech were produced. The corpus pre-defined 3 smaller sets 'train', 'dev' and 'test' and this partitions were kept defining 3 sets with 535, 611 and 376 utterances, respectively.

Fala Bracarense

The 'Fala Bracarense' is a corpus of approximately 80 hours (80 interviews with about 1 hour each) at 16kHz, where both the interviewer and interviewee talk in EP, and as the same states, with more of a Braga's accent.

The subjects of the interviews were varied that allowed the production of diverse forms/structures and the use of pragmatic and discursive mechanisms. The project had the aim to achieve the necessary representativeness of individuals belonging to different parameters of Gender, Age, and Education Level (in years) in the city of Braga. The interviews were transcribed according to semi orthographic rules. The Age, Education Level and Gender of the interviewees are described on Table 9.

Education Level / Age / (M/F)	25	26-59	60-75	75+
0-3	0/0	1/1	1/3	2/3
4-9	3/3	3/3	3/3	3/3
10-12	3/3	3/3	3/3	0/3
Degree	3/3	3/3	3/3	3/3

Table 9: Fala Bracarense's Age, Education Level and Gender of Interviewees

There is no information about the Age or Education Level of any of the interviewers, although it was possible to extract from the metadata that all interviews were female and did a certain amount of interviews as presented on Table 10.

	'IC'	'JV'	'CS'	'MA'	Unknown
Num. Of Interviews	19	25	20	14	2

Table 10: FalaBracarense Interviewers

The corpus is publicly available at [108], where for each interview there is a .wav audio file containing the complete interview and a .exb file with metadata and the timestamped transcription of the interview. Using these timestamps, the complete interviews were split into audio and text files representing only one sentence each, complying with the formatted used by the ASR tools. Since the transcription of this corpus did not conform with the transcription of the others, some alterations had to be done to the text. These alterations mostly concern the removing of the '•' symbols that represent silence and '((risos))' that represented laughter as well as removing all punctuation and converting all lowercase characters into their corresponding uppercase character.

RTP Play Corpus

By accessing the RTP Play platform [109], we were able to gather a very sizable amount of subtitled audio from different television shows from the Portuguese public broadcasting corporation. A script was created to fulfill the following steps:

1. We firstly gathered all IDs and URLs corresponding to TV Shows available on the platform, by parsing the 'https://www.rtp.pt/play/programas' web-page, using Beautiful Soup [110];
2. For each of the available series, we gathered a complete list of all existent seasons and corresponding episodes, using Selenium [111];
3. By consulting every episode ID, we were able to make a list of episodes in each season that contained subtitles (via a .vtt file);
4. Every episode of every TV show available that contained subtitles was download and their audio was promptly extracted and converted to .wav files and the latter were paired with the corresponding .vtt files;
5. The list of series was filtered, since some of the TV Shows were not in EP (majority in English), these were deleted from the set;
6. Because some of the .vtt files extracted were empty or contained formatting errors, these were deleted (with their corresponding audio file pair).

After this process, we achieved a large corpus audio data and their transcription. A final step was done, using the timestamps and transcriptions present on the subtitle file, to split the audio and text files into smaller ones, each corresponding to one utterance present in the audio.

```
8
00:00:21.400 --> 00:00:24.319
Uma camareira
que acha que sabe mais do que eu.

9
00:00:24.560 --> 00:00:26.599
E um cozinheiro
que já nasceu queimado.
```

Listing 6.1: Excerpt of one .vtt file used

This whole process, initially created a very big corpus containing around 1387 hours of raw audio, split between 150 shows and a total of 2883 episodes. But unfortunately for us, not all data could be used for 2 main reasons:

1. A very sizable part of the data, contained transcription errors and/or miss-alignment captions;

2. Since we are dealing with subtitled audio from TV shows, and not carefully transcribed audio for the purpose of ASR, many alterations to what was really said in the audio files, were done in the transcription. It is common throughout the corpus the use of synonyms and omitting words for improved clarity on the subtitles, but at the same time, contaminating the quality of this data for use in ASR.

As stated before, the episodes were split into smaller clips based on their subtitle timestamps, and for every available episode a number of clips were listened to. The series were then classified, and split into 3 different sets with processed data, based on their transcription accuracy: A 'perfect set' with around 15 hours of audio with accurate transcriptions; A 'good set' with around 55 hours of audio containing some small transcription errors or small misalignment problems; And lastly, a 'bad set' with all remaining audio containing very noticeable miss-alignment issues and deep transcription errors.

Unfortunately, it was not possible, in the time frame of this project, to guarantee the complete accuracy of each transcription in either set, and for this reason, no validation or test set were created from this corpus.

NOS Calls

The NOS Calls corpus, is a small set of hand transcribed audio totaling 2 hours of customer service calls from NOS. A total of 10 hours of audio were provided by NOS with the purpose of testing the model accuracy for telephone data.

One majorly important aspect that is distinctive from all previously described corpus, is that this data from NOS Calls was originally .mp3 files limited at 8000 Hz and 64.0 kb/s, making it of considerably worse quality than any other corpus used during this project, and as discussed before, this usually reflects on lower performance results.

The original 10 hours of raw audio were carefully listened to, all audio parts that could be used in identifying any person talking were removed (by silencing the respective audio frames) and frames only containing noise or music were removed. Finally, the files were split into smaller utterances using pydubs's silence detection [112]. This processing pipe produced a total of 2 hours of transformed audio clips (ranging from 1 to 5 seconds) that were listened to and carefully transcribed.

Unfortunately, even after anonymizing all the data, we were not authorized to use this data on the Google's Colab Platform and, because of time restrictions and not having much hope in getting good results with such low quality audio, this data ended up not being used for training and testing the models.

6.1.3 *Portuguese Text Datasets*

CETEM Público

The CETEMPúblico (Corpus de Extractos de Textos Electrónicos MCT/Público) is a corpus of approximately 180 million words in EP, created by the Project 'Processamento computacional do português', in a protocol between the Portuguese Ministry of Science and Technology (MCT) and the Portuguese journal 'PÚBLICO' in April of 2000. The corpus is divided in different extracts that correspond to different articles, and each of these is then split in a number of tags. The number of these different tags present on the whole document is described Table 11.

This corpus is intended primarily for all those who develop programs that process the Portuguese language, and who consequently need raw material for their work. The overwhelming majority of CETEM Publico's text is in EP, although there are some texts by Brazilian and African authors [113].

Structure	Number
Extracts <ext>	1504258
Paragraphs <p>	2571735
Phrase <s>	7082094
Titles <t>	655059
Authors <a>	247392
Elements of list	80060

Table 11: CETEM Público 1.7 Structure, extracted from [113]

The latest version (v1.7 from 18 of September of 2001) was used to train language models. All tags were removed from the corpus, and each of this elements and phrases were split into different lines to comply with the format used to train the language model.

NOS Verbatium

The NOS Verbatium corpus is a small set of text data extracted from 15 different data-sets with a variety of subjects. This different sets contain information with a variety of subjects, from user complaints to service status reports, that are used to measure user happiness with the services provided.

The columns of each set were individually analyzed and the ones useful for our project, were extracted and concatenated into a single text file, producing a total of 4400 lines, each representing a phrase. This data provided by NOS is specially interesting because it was full of examples where telecommunications specific terms were present.

6.2 METHODOLOGY

SeanNarens's deepspeech.pytorch [83] implementation of Deep Speech 2, as well as most notorious modern ASR frameworks let us use GPU's power to greatly reduce the training time of acoustic models. Since we did not have access to local machines equipped with powerfull GPUs for machine learning purposes, the possibility of renting machines on services like AWS or Azure was studied, but, after investigating, we concluded that these types of services were too expensive for the development of this project. We decided to use Google's Colab Platform to do all model creation steps. This platform gives users free access to modern GPUs, with some restriction. This choice exhibited some advantages and disadvantages:

- With the use of GPUs, we greatly increase the model's training speed, to what could be considered as reasonable time;
- Although each user is limited to one only instance with GPU access, and each of this sessions is limited to one GPU, by using multiple accounts we were able to train multiple models at the same time;
- Each of these sessions are limited to roughly 10 hours for each day, and since for every new session, the framework needed to be installed, and all training data transferred to be locally stored on the machine, this equated to less training time per day that was preferable;
- Sessions with access to GPUs, are severely limited in storage space (75GB in total). For this reason several steps were taken to ensure the least use storage space possible, and more importantly we were not able to use all training data available and every augmentation step was done in real-time.

Because of this choice to use Google's Colab Platform, we were given machines with a variety of configurations, but on all instances where training time was evaluated, it was assured that a machine with the following specifications was used:

- 2 Cores of Intel(R) Xeon(R) CPU @ 2.20GHz [Family 6, Model 79];
- 15GB RAM;
- Nvidia Tesla P100 GPU;
- 75GB of Storage.

The files containing the full configuration that was used to created these models (with exception to the first test where batch size was changed) are presented in appendix A.1 and the file containing the test configuration is presented in appendix A.2.

6.2.1 *Acoustic Model*

The model created is based on Seannaren's implementation of Deep Speech 2 [83] and Figure 20 illustrates the model's architecture.

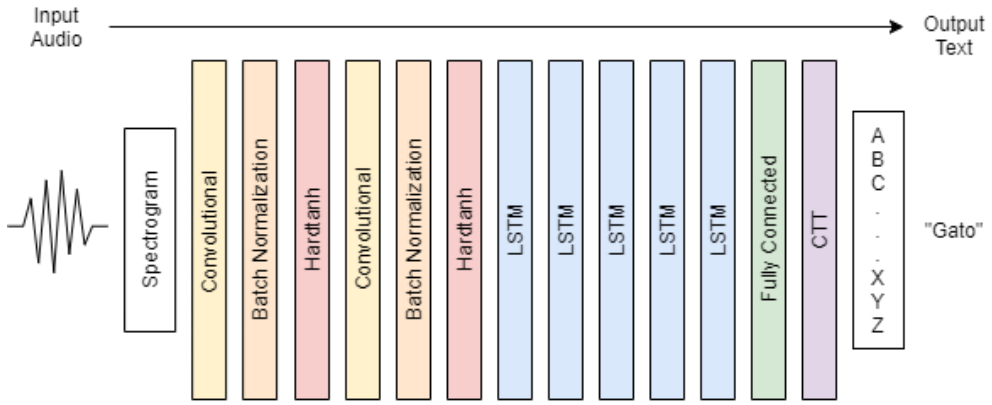


Figure 20: Architecture of Models Used

In the beginning, audio stored in a '.wav' format is possibly augmented during training and fed into the system that creates a visual representation (spectrogram) of the data stored. This representation is then fed into the first layer of the neural network, and for each instance of time, a total of 161 of features are used. To get the complete value of features used for a audio file, we need to multiply this number by the number of windows used to represent the audio in the spectrogram.

The first 6 layers of the network are 2 sets of a Convolutional Layer, Batch Normalization Layer and a Hardtanh Layer, and are part of what is referred in deepspeech.pytorch as a MaskConv. These first layers are used by the model to store the important features of the audio, in the same way we could extract them in other models using, for example, MFCC.

In each of these sets, the first layer is a spatial convolution, commonly found in image-related Deep Learning tasks, used for features extraction. These, over the training, are tuned to release redundant information found in the spectrogram and take into account the important information. Each convolution layer is followed by a batch normalization layer used for a faster convergence during training and a final Hardtanh activation layer.

After these 2 sets of layers, the output is then fed to a stack of 5 LSTM bidirectional RNNs, where batch normalization is also applied. After the bidirectional recurrent layers, one fully connected layer is employed to generate the scores over the label set, and in the last CTC layer we calculate the probability distributions over the labels with a softmax layer. At each time-step, the softmax layer predicts a label, with this label being either a label from the set of character used or the blank token.

Finally, the predicted transcription is decoded from the sequences given according to the probability distributions. One important aspect is that the created model does not contain any digits or punctuation marks in the label set. To accommodate the used data to not contain these types of characters, all numbers where transformed into the their word notation, and all forms of punctuation where removed from the text.

Transfer Learning

For transfer learning, a pre-trained model provided by github's user 'altavatarindia' [114], trained on the previously described English data, was used as the back-model, and it was able to achieve the results presented on Table 12.

Dataset	WER	CER
Librispeech clean	11.0 %	3.6 %
Librispeech other	28.0 %	11.7 %
Ted	26.8 %	9.1 %
Commonvoice	16.0 %	7.0 %

Table 12: Pre-Trained Model Results, extracted from [114]

Following the results found on [115] that demonstrated that fine-tuning a pre-trained English ASR model with new BP data, provided better results than freezing or training the model from scratch. We chose to use the existing implementation of fine-tuning present on deepspeech.pytorch [83] to train a model, and compare its performance to a model trained from scratch. A modification to the code was done, since the already existing implementation did not take into account our need to change the model's output label set to a broader group that contained all characters used in EP (the previously used labels for english + [ç, á, é, í, ó, ú, à, â, ê, ô, ã, õ]). To do this, the final layer, that originally contained 29 features, was replaced by a layer containing all the 41 outputs needed, where the already known labels kept their weights and the newer introduced characters started with a random weight distribution.

6.2.2 *Language Model*

Two different language models were created using kenlm language model [116]. One bigger model using the data present in CETEMPúblico, and another smaller model using data only present in NOS Verbatium Corpus. Both of these models were created at a max order value of 3 (meaning these models contain a 1-Gram, 2-Gram and 3-Gram representation of the data), and as a final step, to improve performance, both models were filtered to

remove entries that would never be queried (based on our data) and transformed in their binary representation. Unfortunately, the smaller model was never used during the project, because it was created to improve the performance of the NOS's test data. The performance improvements of using CETEMPúblico Language Model is going to be tested in Chapter 7.

CETEMPúblico Language Model

Using the data from CETEMPúblico a language model was created. To reduce the size of the model, we restricted the saved n-grams by filtering instances that appeared a small amount of times, in this case, all 2-grams and 3-grams that were found in the data less than 3 and 10 times, respectively. The CETEMPúblico Language Model dimensions is presented in the following Table 13.

N-Gram	Number of Values
n = 1	759385
n = 2	2809791
n = 3	1526187

Table 13: Dimensions of CETEMPúblico Language Model

NOS Verbatium Language Model

Using the data from NOS Verbatim a language model was created. To reduce the size of the model, we restricted the saved n-grams by filtering instances that appeared a small amount of times, in this case, all 2-grams and 3-grams that were found in the data less than 3 and 5 times, respectively. The NOS Verbatim Language Model dimensions is presented in the following Table 14.

N-Gram	Number of Values
n = 1	11030
n = 2	27971
n = 3	26785

Table 14: Dimensions of NOS Verbatim Language Model

6.3 VALIDATION STRUCTURE

In the next chapter, we will present the results of the models created. The evaluation of said models will be presented in terms of WER and CER, as these are very common metrics for evaluating ASR systems [117]. These metrics were already presented in Chapter 3. We

will additionally refer to the model's training time when comparing models using different augmentation techniques. The validation will be split into two different sections, 'Initial Validation' and 'Final Validation':

INITIAL VALIDATION: Initial testing models to test the effect of batch size and data augmentations techniques previously described on Chapter 5. These models were created using only the "Fala Bracarense" Dataset;

FINAL VALIDATION: Final models created to test the effect of all data augmentations techniques and the language model created. These models were created using all data available and the tests include results of the two 'Test Sets' created, one with data from 'Fala Bracarense' Dataset and another with 'Common Voice' Dataset.

A final step is done comparing the results of the final model created with the performance of two Speech Recognition Services using the test sets.

7

VALIDATION AND PERFORMANCE EXPERIMENTS

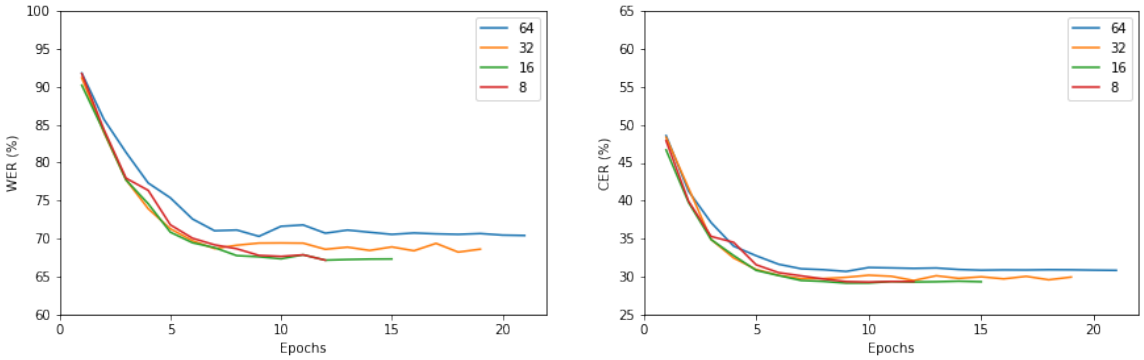
7.1 INITIAL VALIDATION

In this section, we will present preliminary results done as 'proof of concept' for evaluating training techniques described in the previous chapter. All of these preliminary experiments were done using only the 'Fala Bracarense' dataset which, as stated before, contains around 80 hours of data. This corpus was split in 2 parts, 80% for training and 20% for validation, and no test corpus was used to determine system performance on 'unseen' data.

7.1.1 *Batch Size*

When talking about machine learning, the batch size defines the number of samples that will be propagated through the network. For instance, when we have a total of 550 training samples, and set the batch size variable equal to 100. The algorithm takes the first 100 samples (from 1st to 100th) from the training dataset and trains the network, changing the weights once completed. Next, it takes the second 100 samples (from 101st to 200th) and repeats the process. This procedure is repeated until we have propagated all samples through of the network, and at the end of this procedure, a epoch of training is concluded.

A higher batch size usually equates to a faster training speed but can also significantly reduce the final model accuracy. A validation to verify such impacts was done, and produced the results presented in Figure 21. The experiment consisted of 4 models with different batch size values, trained in equivalent machines for roughly the same amount of time. The values used during training were 64, 32, 16, 8. The WER and CER achieved in this validation are described in Table 15.



(a) WER (b) CER

Figure 21: Batch Size Effect on Accuracy and Training Speed

Batch Size	Epoch Reached	WER	CER
64	21	70.4%	30.9%
32	19	68.5%	29.6%
16	15	67.3%	29.4%
8	12	67.2%	29.4%

Table 15: Batch Size Variation Results

Using batch size of 32 proved to be a good compromise between speed and the resulting accuracy, and subsequently was the value used for all remaining validation tests. To better improve our changes of having good results in the creation of the model, the final sections of experiments were done using batch size of 16.

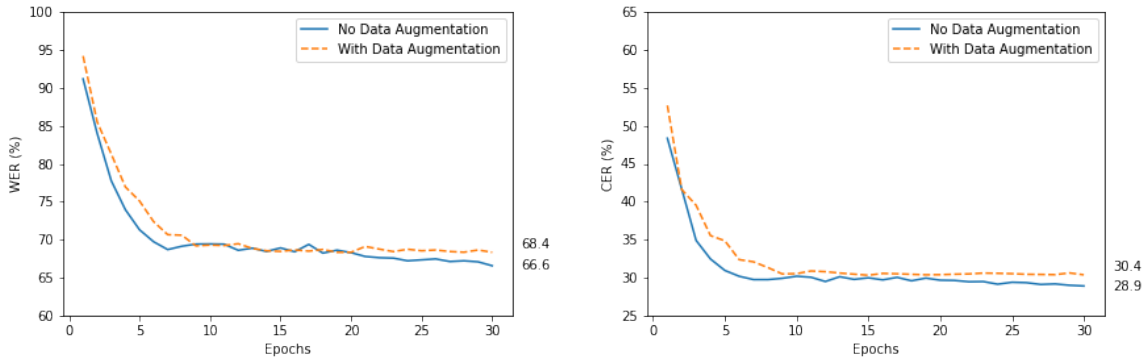
7.1.2 *Noise Injection*

For Noise Injection, snippets of audio containing noise were mixed with audio during training. To inject noise in the data, noise snippets were mixed at a probability of 0.4 and with a volume of 0.0-0.4 relative to the volume of the original audio file. The noise snippets used for simulation come from a collection of sample noises and YouTube of simulated and real-life noises from videos. Since all noise audio files were of greater length than any of the data present during training, as previously described, smaller clips were audio were extracted from noise files with the same duration as the training clip and with a random start position.

It is important to note that the grand majority of the training data used in this section had no considerable noise. The small part that had noise consisted only of small interjections

('hmm', 'mm-hmm', etc.) by either member of the interview, while the other person was talking.

Training with Noise Injection increased the training time of each epoch by an average of 8%. Our experiments show that Noise Injection provided no improvements over the model with no augmentation, achieving 68.4% WER and 30.4% CER, an increase of 3.4% and 5.5% respectively, as seen in Figure 22.



(a) WER (b) CER

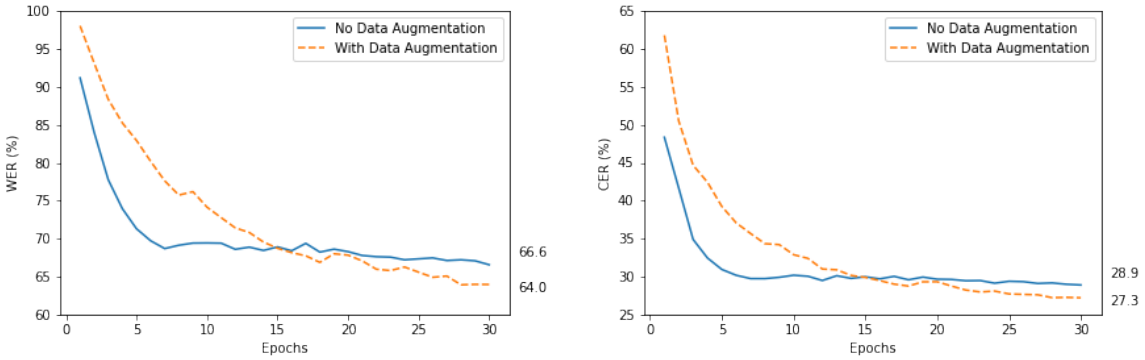
Figure 22: Noise Injection

This was a disappointing result as we expected that, although only a very small amount of noise was present in the data, the use of Noise Injection would provide improvements in accuracy, but this was not the case.

7.1.3 *SpecAugmentation*

As per definition of SpecAugment, the augmentation policy consists of warping the features, masking blocks of frequency channels, and masking blocks of time steps. A implementation developed by [83] was used based on the method developed by Google [99].

Training with Spec Augmentation increased the training time of each epoch by an average of 0.7%. Our experiment shows that Spec Augmentation provided improvements over the model with no augmentation, achieving 64.0% WER and 27.3% CER, an increase of 4.0% and 5.5% respectively, as seen in Figure 23.



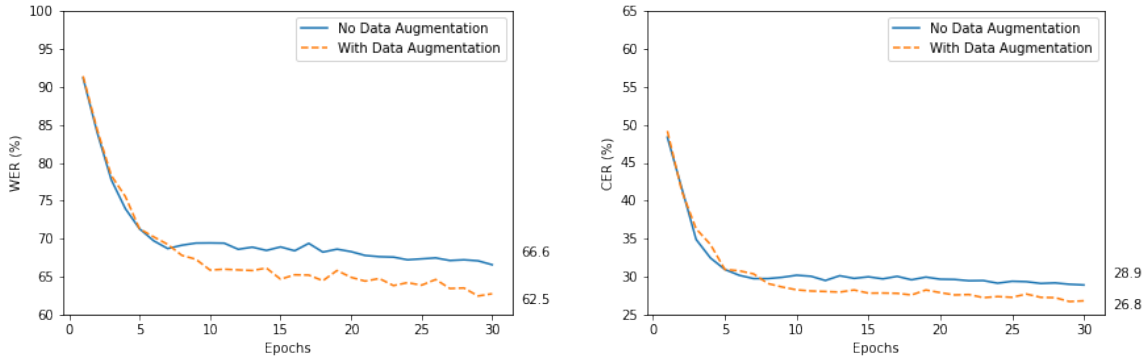
(a) WER (b) CER

Figure 23: Spec Augmentation Results

As expected, Spec Augmentation provided better results, but also showed an interesting phenomenon. In the first few epochs the base model had better performance, but after epoch 15 the roles were reversed. We think this was probably caused by the increase in difficulty created by Spec Augmentation. By 'hiding' certain information to the model, it can make the task of Speech Recognition relatively harder, creating worst performance of the model in the first few epochs. After a few epochs, the model 'learns' how to work better with fewer data providing better accuracy.

7.1.4 *Speed Perturbation*

In Speed Perturbation, small changes, either slight increases or decreases, are applied to the pitch of the audio signal during training. The training with Speed Perturbation increased the training time of each epoch by an average of 4%. Our experiment shows that Tempo Perturbation provided improvements over the model with no augmentation, achieving 62.5% WER and 26.8% CER, an decrease of 6.1% and 7.2%, respectively, as seen in Figure 24.

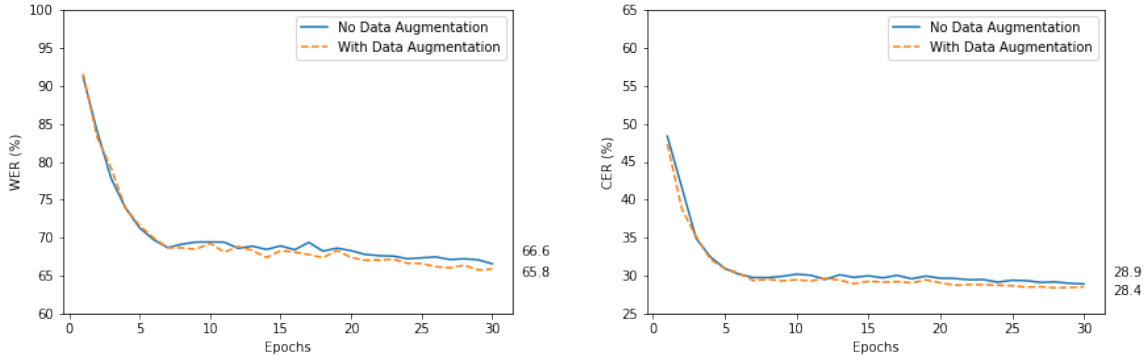


(a) WER (b) CER

Figure 24: Tempo Perturbation

7.1.5 *Volume Perturbation*

In Volume Perturbation, small changes, either slight increases or decreases, are applied to the strength of the audio signal during training. Training with Volume Perturbation increased the training time of each epoch by an average of 1.5%. Our experiment shows that Tempo Perturbation provided improvements over the model with no augmentation, achieving 65.8% WER and 28.4% CER, an improvement of 1.2% and 1.7% respectively, as seen in Figure 25.



(a) WER (b) CER

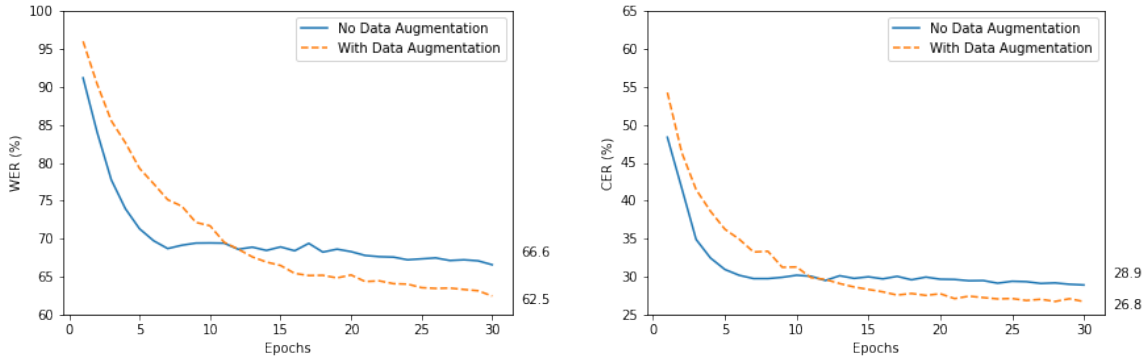
Figure 25: Volume Perturbation

7.1.6 *Mixed-Bandwidth Training*

In Mixed-Bandwidth Training, files of multiple bandwidth are used during training. As stated before all our data was recorded in either 16 kHz (grand majority) or 8 kHz and, as so, were the only considered during training. All data present in the 'Fala Bracarense' set is

sampled at 16 kHz, with a probability of 0.3, training utterances were down-sampled to 8 kHz. Because our Feature Extraction is configured to operate at 16 kHz, the waveform was then up-sampled back to 16 kHz before fed into the model.

Training with Mixed-Bandwidth increased the training time of each epoch by an average of 2%. Our experiments show that the effectiveness of Mixed-Bandwidth Training achieved 62.5% WER and 26.8% CER reduction over a system trained with no augmentation, an increase of 6.1% and 7.2%, respectively, as seen in Figure 26.



(a) WER (b) CER

Figure 26: Mixed-Bandwidth Training

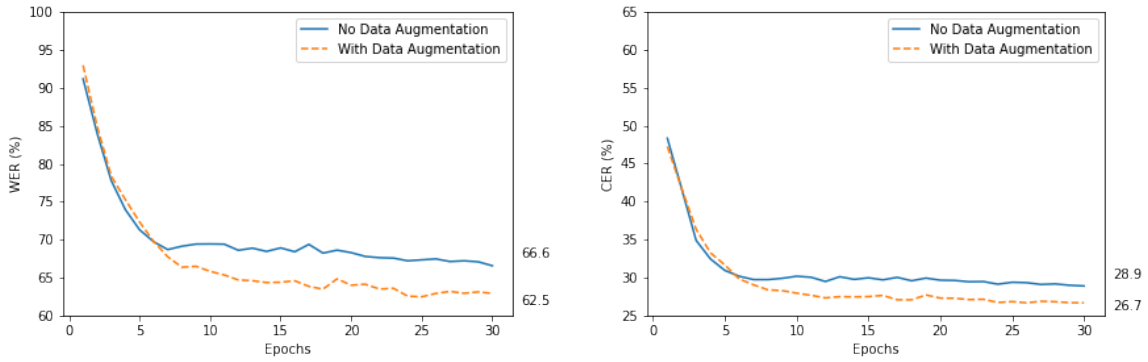
Like the previous augmentation technique, using narrow-band audio (in this case, down-sampled wide-band speech) in speech recognition removes part of the information provided to the model maintaining the task of Speech Recognition. For this, the task can be considerably harder, initially achieving worst results that the base model where only wide-band audio was used during training. But making it so that with more training, the model achieves better results when recognizing wide-band speech.

7.1.7 *Audio Encoding and Codecs Simulation*

To simulate a variety of audio encodings and codecs before transmission to the recognition system, we encode and decode each waveform with a randomly selected encoding or codecs, focusing on commonly used in telephone speech. For this validation the following encodings for wav files were used: Signed-Integer, U-Law and A-Law; as well as the mp3, mp4 codecs. Note that the training data we use may already be encoded in arbitrary ways before we obtain it, in which case it is decoded and then re-encoded with the selected codec.

The training with Audio Encoding and Codecs Simulation increased the training time of each epoch by an average of 4%. Our experiments show that the effectiveness of Audio Encoding and Codecs Simulation achieved 62.5% WER and 26.7% CER reduction over a

system trained with no augmentation, an improvement of 5.4% and 7.2% respectively, as seen in Figure 27.



(a) WER (b) CER

Figure 27: Audio Encoding Simulation

7.1.8 *Conclusions*

These initial preliminary validations ended up being extremely useful, both in finding a good set of values for batch size to use during training of the remaining models, and provided some interesting results of the use of data augmentation techniques, as seen on Table 16.

Technique	WER	CER
Base Model	66.1%	28.8%
Noise Injection Model	68.4% (+3.4%)	30.4% (+5.5%)
SpecAugmentation Model	64.0% (-4.0%)	27.3% (-5.5%)
Tempo Perturbation Model	61.8% (-6.5%)	26.6% (-7.6%)
Volume Perturbation Model	65.8% (-1.2%)	28.4% (-1.7%)
Mixed-Bandwidth Training Model	62.5% (-6.1%)	26.8% (-7.8%)
Audio Encoding and Codecs Simulation Model	62.5% (-5.4%)	26.7% (-7.2%)

Table 16: Data Augmentation Results

While not increasing the training time by a significant amount, all of the augmentation techniques, with the exception of Noise Injection, provided in the end a model with greater accuracy than our base model. With these results in mind, we will continue to our final validation.

7.2 FINAL VALIDATION

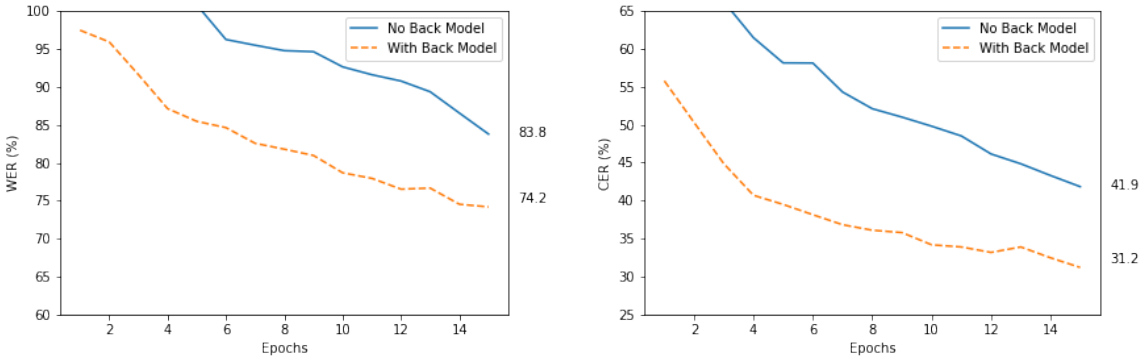
In this section, we present the results from training the final models, which were trained using all available data described in Chapter 6.

The corpus was split into training, validation, and testing subsets, as follows:

1. 'Fala Bracarense' corpus was randomly split in 3 parts, 70% for training, 20% for validation and 10% for testing;
2. 'Comon Voice' corpus was filtered to only contain EP data, and the original partitions of the data were kept;
3. 'Voxforge' corpus was filtered to only contain EP data, and, because of the reduced size, it was decided to split it in 2 parts, 80% for training, 20% for validation.
4. 'RTP Play' corpus was used solely for training since there were slight inaccuracies in the translated speech;
5. 'Nos Calls' corpus was not used in this model.

7.2.1 *Transfer Learning*

As stated before, we experimented with Transfer Learning techniques by using a pre-trained English model as our 'base' neural network. Due to the increased size of the Portuguese character set, we added the remaining needed nodes to the last layer of the base-model (with random weights) and tuned the entire model for the Portuguese dataset using all previously discussed augmentation techniques. Our experiments show that the effectiveness of the Transfer Learning achieves 74.2% WER and 31.2% CER after training for 15 epochs in comparison with 83.8% WER and 41.9% for the model trained from scratch, which represents an improvement of 11.4% and 25.4%, respectively, while bearing no noticeable difference in training time, as seen on Figure 28.

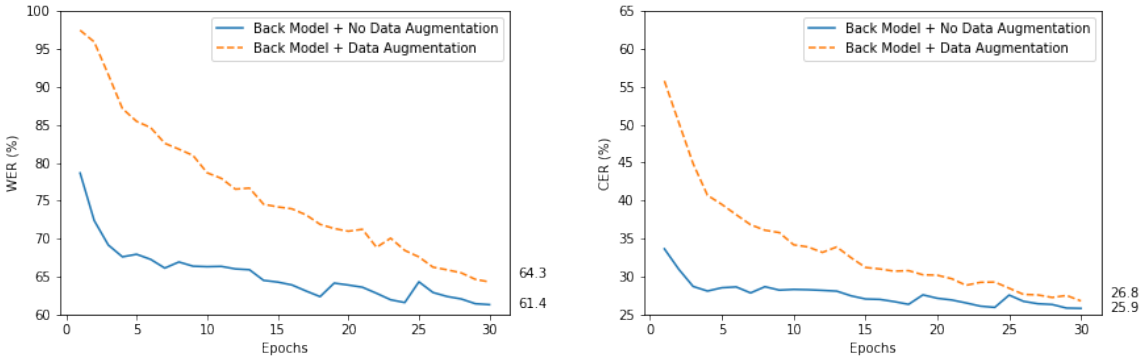


(a) WER (b) CER

Figure 28: Results from Transfer Learning from English Back Model

7.2.2 *Data Augmentation*

For the last two models, we decided to keep using Transfer Learning, since the results on the previous evaluation were very promising, and decided to compare the accuracy resulting from a model trained using all augmentation techniques during training versus none. Our experiments show that both options resulted in very similar results, with the "No augmentation" model achieving 61.4% WER and 25.9% CER after training for 30 epochs in comparison with 64.3% WER and 26.8% achieved by the model trained using all previously described augmentation, as seen in Figure 29.



(a) WER (b) CER

Figure 29: Complete Data Augmentation Results

The results of these models when using the test subsets for Fala Bracarense and Common Voice Corpus are presented in Table 17.

	No Augmentation	All Augmentations
Fala Bracarese Test	61.0%	64.7%
Common Voice Test	72.4%	74.9%

Table 17: WER Results of Test Set with Data Augmentation

Since the 'No Augmentation' model provided slightly better results, both in terms of CER and WER, than the 'All Augmentation' variant, all remaining tests were done only on the prior model.

7.2.3 *Language Model*

As referred before, a language model can greatly improve the results provided by an ASR system, and to test this hypothesis we present the results of using the created "CETEMPublico" language model paired with the 'No Augmentation' model of the previous section. Our experiments show that the vast effectiveness of Language Models made an improvement of 24.7% and 40.7% on the 'Fala Bracarense Test' and the 'Common Voice Test', respectively, as seen in Table 18.

	Base Model	With Language Model
Fala Bracarense Test	61.0%	40.5%
Common Voice Test	72.4%	56.3%

Table 18: WER Results of Test Set with Language Model

7.3 COMPARISON WITH OTHER SERVICES

To further evaluate our developments, we decide to make a comparison of our WER results with the output of two different online ASR Services, namely Google's Speech Recognition and Facebook's Wit.Ai. The results of the said comparison are presented in Table 19.

	Base Model	Google	Facebook
Fala Bracarense Test	40.5%	45.6%	67.7%
Common Voice Test	56.3%	12.3%	30.3%

Table 19: Comparison of our model with Commercial Alternatives

This comparison provided interesting insights:

1. Our model performed slightly better than Google's and significantly better in the 'Fala Bracarense Test', but performed significantly worse in the 'Common Voice Test';
2. Google's ASR achieved better performance than Facebook's offer in both test datasets;
3. Both commercial alternatives achieved better performance on the 'Common Voice Test' than in the 'Fala Bracarense Test'.

8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

ASR opens the door to a wide range of possible improvements in the customer experience. The use of speech recognition is increasing every day and this is currently one of the main methods of interactions between users and their electronic devices. The use of this type of technology has increased significantly in recent years, and this change being the result of the recent evolution in ASR systems. In this dissertation we firstly covered what is phonetics and speech, and how we can represent them. Although very intuitive for us, speech is in reality a very interesting and complex process, that involves different parts of the human body.

The main objective of this dissertation was the construction of an ASR system for the EP language. To do this we based our approach on deepspeech.pytorch, which is a derivation from Deep Speech 2 [118], a state-of-the-art end-to-end system. We introduced ASR systems and how they are generally structured with the recent split between Traditional (HMM-GMM based) and new End-to-End Models (DNN-based) models. Neural networks with a large number of layers, or simply Deep Learning, have in recent years become one of the most studied way of creating ASR systems, this trend can be traced by the evolution of DNN algorithms, as well as the recent increase in power of recent hardware (primarily GPUs) and the availability of data.

Although there is a recent improvement in ASR accuracy, there are many factors that make implementing these systems a challenge. One simple factor is that ASR is not a simple binary classification problem, as we are not simply expecting a 'yes' or 'no', 1 or 0 response from the model. The correct output is a series of words that precisely represent what is said in a certain utterance, from thousands of words presents in the ever increasing vocabulary that exist in the EP language. A second aspect is the low amounts of data readily available for the EP. Any type of Neural Network problem, especially DNN implementations, are deeply restricted in performance by the data used in training, both in terms of the size and the quality. Over the length of project, we did our best to gather large amounts of

data from a variety of different sources/projects, but all data combined is still a small fraction compared to what is commonly available in the English Language, and extremely small when comparing to the amount of transcribed speech hours used on state-of-the-art commercial ASR systems. To try to minimize this problem we also introduced transfer learning and showed that training these types of models largely benefited from a properly initialization of the weights (provided by a back-model trained in large amounts of data). Nonetheless the data collected can be used in a variety future projects regarding human voice, including a future iteration of this project.

We also tested a variety of batch sizes to find a good compromise in terms of training speech and the accuracy achieved by our models and a total of 6 different data augmentation techniques were tested during the creation of our model, 5 of them when implemented individually provided better accuracy on our validation data. Also, we showed the vast improvements that a Language Model provided to our ASR acoustic model's accuracy, reducing the WER in our primary test set by approximately 25%.

Given a relative short training time, our best model achieved an WER of 40.05% in our most prominent test set, with this result being slightly better when comparing to the values achieved using commercial services; however, we have a long way until we achieve results similar to those of commercial systems in other corpus.

8.2 Future Work

After everything that was done and described in this dissertation, there is still a lot of work to be done. Since we based the model in the base transfer learning methods, we have not stressed our topology by adding convolutional or hidden layers. Additionally, our parameter optimization was very limited to different batch size values. We also didn't explore other kinds of RNN structures (e.g., GRU) or different kinds of input features (e.g., Raw Audio, MFCC, PLP, LPC).

Increasing our training data is also highly recommended as state-of-the-art models are trained with thousands of hours of transcribed speech. An increase in the amount of data could lead to significant improvements in the model accuracy. Furthermore, we did not investigate adding punctuation, accents, and hyphens to our softmax layer, which could disambiguate some Portuguese words, and output a more accurate representation of what is said in utterances. One of the more interesting points that we regret not having done is creating a base BP dataset and training a back model using this data and later applying transfer-learning techniques to a EP model variant. In addition, it would be interesting to investigate what are the more error-prone phonemes, characters and words.

Furthermore, the created models could be trained for longer periods of time, and with it possibly reaching slightly improved accuracy results. It would be also interesting to compare

the accuracy achieved in this model with a different classical ASR approach. Moreover, as encoder-decoder based models with attention mechanisms are raising in popularity in the area for end-to-end solutions, and this includes speech recognition, it would be very interesting to compare performance with these type of models.

BIBLIOGRAPHY

[1] Basem Rizk. "Evaluation of State Of Art Open-source ASR Engines with Local Inferencing". PhD thesis. Aug. 2019. DOI: 10.13140/RG.2.2.34901.37603.

[2] Li Deng. "Industrial Technology Advances: Deep Learning - From Speech Recognition to Language and Multimodal Processing". In: *APSIPA Transactions on Signal and Information Processing (Cambridge University Press)*. Feb. 2016.

[3] Ethnologue. *What is the most spoken language?* https://www.ethnologue.com/guides/most-spoken-languages. Accessed: 2020-11-13.

[4] Dan Jurafsky Andrew Maas. *Stanford CS224S / LINGUIST285 Spoken Language Processing Lecture 2: Phonetics*. 2017.

[5] Friedrich Heinrich Hermann. Techmer. *Phonetik: Zur vergleichenden Physiologie der Stimme und Sprache*. 1880.

[6] Hartmut Traunmüller and Anders Eriksson. "The frequency range of the voice fundamental in the speech of male and female adults". In: 2 (Jan. 1995).

[7] David N. Sorenson. "A fundamental frequency investigation of children ages 6–10 years old". In: *Journal of Communication Disorders* 22.2 (1989), pp. 115–123. ISSN: 0021-9924. DOI: 10.1016/0021-9924(89)90028-2.

[8] Patricia Keating and Grace and Kuo. "Comparison of speaking fundamental frequency in English and Mandarin". In: *The Journal of the Acoustical Society of America* 132.2 (2012), pp. 1050–1060. DOI: 10.1121/1.4730893.

[9] David Sorensen and Yoshiyuki Horii. "Cigarette smoking and voice fundamental frequency". In: *Journal of Communication Disorders* 15.2 (1982), pp. 135–144. ISSN: 0021-9924. DOI: 10.1016/0021-9924(82)90027-2.

[10] Eddy Bøgh Brixen. "Audio Metering. Measurements, Standards and Practice". In: Focal Press, 2014.

[11] Jonathan Hui. "Speech Recognition — Phonetics". In: (Aug. 2019). URL: https://jonathan-hui.medium.com/speech-recognition-phonetics-d761ea1710c0.

[12] Instituto Camões. *A pronúncia do português europeu Convenções e Transcrição Fonética*. Accessed: 2020-06-10. 2006. URL: http://cvc.instituto-camoes.pt/cpp/acessibilidade/capitulo2_1.html.

[13] Dimitra Vergyri, Lori Lamel, and Jean-Luc Gauvain. "Automatic speech recognition of multiple accented English data." In: Jan. 2010, pp. 1652–1655.

[14] Sarah Stevenage, G. Clarke, and A. McNeill. "The effect of regional accent on voice recognition". In: *Journal of Cognitive Psychology* (Jan. 2012).

[15] Ing-Jr Ding. "Speech recognition using variable-length frame overlaps by intelligent fuzzy control". In: *Journal of Intelligent Fuzzy Systems: Applications in Engineering and Technology* 25 (Jan. 2013), pp. 49–56. DOI: 10.3233/IFS-2012-0613.

[16] S. S. Stevens, J. Volkmann, and E. B. Newman. "A Scale for the Measurement of the Psychological Magnitude Pitch". In: *Journal of the Acoustical Society of America* 8.3 (1937), pp. 185–190. URL: %5Curl%7Bhttps://academic.microsoft.com/paper/2103869314%7D.

[17] D. O'Shaughnessy. *Speech Communications: Human and Machine*. Wiley, 2000. ISBN: 9780780334496. URL: %5Curl%7Bhttps://books.google.pt/books?id=yHJQAAAAMAAJ%7D.

[18] Perry Cook. "Identification of Control Parameters in an Articulatory Vocal Tract Model, with Applications to the Synthesis of Singing". MA thesis. Stanford, California: Stanford University, Dec. 1990. URL: https://ccrma.stanford.edu/files/papers/stanm68.pdf.

[19] Perry Cook. "Identification of control parameters in an articulatory vocal tract model, with applications to the synthesis of singing". PhD thesis. Jan. 1991.

[20] Priberam. *Dicionário Priberam da Língua Portuguesa*. https://dicionario.priberam.org/. Accessed: 2010-06-01.

[21] E. Fosler-Lussier and N. Morgan. "Effects of speaking rate and word frequency on pronunciations in convertional speech". In: *Speech Commun.* 29 (1999), pp. 137–158.

[22] Microsoft. "Voice Report 2019". In: Accessed on 18/11/2020. 2019.

[23] *Dragon*. Accessed on 20/08/2020. Nuance. URL: https://www.nuance.com/dragon.html.

[24] Kevin Tomba et al. "Stress Detection Through Speech Analysis". In: Jan. 2018, pp. 394–398. DOI: 10.5220/0006855803940398.

[25] D. Carneiro et al. "New Methods for Stress Assessment and Monitoring at the Workplace". In: *IEEE Transactions on Affective Computing* 10.2 (2019), pp. 237–254. DOI: 10.1109/TAFFC.2017.2699633.

[26] D. Carneiro et al. "New Methods for Stress Assessment and Monitoring at the Workplace". In: *IEEE Transactions on Affective Computing* 10.2 (2019), pp. 237–254. DOI: 10.1109/TAFFC.2017.2699633.

[27] Accessed: 2020-08-20. Talkdesk. URL: https://www.talkdesk.com/.

[28] Chung-Cheng Chiu et al. "Speech recognition for medical conversations". In: *INTERSPEECH*. 2017.

[29] Albert Haque and Corinna Fukushima. "Automatic Documentation of ICD Codes with Far-Field Speech Recognition". In: *ArXiv* abs/1804.11046 (2018).

[30] Wael Salloum et al. "Crowdsourced Continuous Improvement of Medical Speech Recognition". In: *AAAI Workshops*. 2017.

[31] Feifan Liu et al. "Towards spoken clinical-question answering: evaluating and adapting automatic speech-recognition systems for spoken clinical questions". In: *Journal of the American Medical Informatics Association : JAMIA* 18 5 (2011), pp. 625–30.

[32] *Car Play*. Accessed: 2020-08-20. Apple. URL: https://www.apple.com/ios/carplay/.

[33] *Android Auto*. Accessed: 2020-08-20. Google. URL: https://play.google.com/store/apps/details?id=com.google.android.projection.gearhead.

[34] Nilu Singh, Alka Agrawal, and Prof. Raees Khan. "Voice Biometric: A Technology for Voice Based Authentication". In: *Advanced Science, Engineering and Medicine* 10 (July 2018). DOI: 10.1166/asem.2018.2219.

[35] Accessed: 2020-08-19. Duolingo. URL: https://www.duolingo.com/.

[36] Amy Nordrum. *CES 2017: The Year of Voice Recognition*. https://spectrum.ieee.org/tech-talk/consumer-electronics/gadgets/ces-2017-the-year-of-voice-recognition. Accessed: 2020-12-19.

[37] Wayne A. Lea. *Trends in Speech Recognition*. USA: Prentice Hall PTR, 1980. ISBN: 0139307680.

[38] Pablo Denes. "The design and operation of the mechanical speech recognizer at University College London". In: 1959.

[39] James W. Forgie and Carma D. Forgie. "Results Obtained from a Vowel Recognition Computer Program". In: 1959.

[40] Suzuki J. and Nakata K. "Terminal Analog Speech Synthesizer". In: 1959.

[41] IBM. *IBM Archives: IBM Shoebox*. https://www.ibm.com/ibm/history/exhibits/specialprod1/specialprod1_7.html. Accessed: 2019-12-15.

[42] Shuji Sakai Toshiyuki; Doshita. "The Automatic Speech Recognition System for Conversational Sound". In: 1964.

[43] Sadaoki Furui. "Fifty years of progress in speech and speaker recognition". In: *Journal of The Acoustical Society of America - J ACOUST SOC AMER* 116 (Oct. 2004), pp. 2497–2498. DOI: 10.1121/1.4784967.

[44] Lee D. Erman and Victor R. Lesser. "Hearsay-II. Tutorial Introduction and Retrospective View". In: 1978.

[45] Jared J. Wolf and William A. Woods. "The HWIM speech understanding system". In: 1977.

[46] Bruce T. Lowerre. "The HARPY speech recognition system". In: 1976.

[47] Hiroaki Sakoe and Seibi Chiba. "Dynamic programming algorithm optimization for spoken word recognition". In: 1978.

[48] L. Bahl et al. "Maximum mutual information estimation of hidden Markov model parameters for speech recognition". In: *ICASSP '86. IEEE International Conference on Acoustics, Speech, and Signal Processing* 11 (1986), pp. 49–52.

[49] Alan V. Oppenheim and Alan S. Willsky. "Signals and Systems". In: 1982.

[50] S. Das and M. Picheny. "Issues in Practical Large Vocabulary Isolated Word Recognition: The IBM Tangora System". In: 1996.

[51] Kai-Fu Lee et al. "The SPHINX speech recognition system". In: *International Conference on Acoustics, Speech, and Signal Processing,* (1989), 445–448 vol.1.

[52] "Dragon Introduces Voice 'Typewriter'". In: *New York Times, March 20* (1990).

[53] Hynek Hermansky and Nelson Morgan. "RASTA processing of speech". In: *IEEE Trans. Speech and Audio Processing* 2 (1994), pp. 578–589.

[54] *HTK Rich Audio Transcription Project Summary and Aims*. Accessed: 2020-01-25. 2002. URL: http://mi.eng.cam.ac.uk/research/projects/EARS/ears_summary.html.

[55] SRI International. *GALE - Global Autonomous Language Exploitation*. Accessed: 2020-01-25. 2010. URL: http://www.speech.sri.com/projects/GALE.

[56] M. Bacchiani et al. "Deploying GOOG-411: Early lessons in data, measurement, and testing". In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. Mar. 2008, pp. 5260–5263. DOI: 10.1109/ICASSP.2008.4518846.

[57] Daniel Povey et al. "The Kaldi speech recognition toolkit". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding* (Jan. 2011).

[58] Kaldi. *Acoustic modeling code*. http://www.kaldi-asr.org/doc/model.html. Accessed on 12/03/2020. 2002.

[59] Maryam Najafian et al. "Improving speech recognition using limited accent diverse British English training data with deep neural networks". In: *Machine Learning For Signal Processing (MLSP)* (Jan. 2016), pp. 13–16.

[60] Vineel Pratap et al. "Wav2Letter++: A Fast Open-source Speech Recognition System". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (May 2019). DOI: 10.1109/icassp.2019.8683535.

[61] *Better, Faster Speech Recognition with Wav2Letter's Auto Segmentation Criterion*. Accessed: 2020-08-13. 2019. URL: https://towardsdatascience.com/better-faster-speech-recognition-with-wav2letters-auto-segmentation-criterion-765efd55449.

[62] Awni Hannun et al. *Deep Speech: Scaling up end-to-end speech recognition*. 2014. arXiv: 1412.5567 [cs.CL].

[63] D. Gibbon, R. Moore, and R. Winski. *Handbook of Standards and Resources for Spoken Language Systems*. Mouton de Gruyter, 1997. ISBN: 9783110153668. URL: %5Curl%7Bhttps://books.google.pt/books?id=0ElXJGBO79YC%7D.

[64] Ondrej Klejch et al. "Speaker Adaptive Training Using Model Agnostic Meta-Learning". In: Dec. 2019, pp. 881–888. DOI: 10.1109/ASRU46091.2019.9003751.

[65] X. D. Huang. "A Study on Speaker-Adaptive Speech Recognition". In: *Proceedings of the Workshop on Speech and Natural Language*. HLT '91. Pacific Grove, California: Association for Computational Linguistics, 1991, pp. 278–283. DOI: 10.3115/112405.112458.

[66] Mouton de Gruyter. "HANDBOOK of Standards and Resources for Spoken Language Systems". In: 1997. ISBN: 3-11-015366-1.

[67] Long Qin. "Learning Out-of-Vocabulary Words in Automatic Speech Recognition". In: 2013.

[68] L. Jiang et al. "High-Performance Robust Speech Recognition Using Stereo Training Data". In: *Proc. ICASSP*. Institute of Electrical and Electronics Engineers, Inc., May 2001. URL: https://www.microsoft.com/en-us/research/publication/high-performance-robust-speech-recognition-using-stereo-training-data/.

[69] Kateryna Zinchenko, Chien-Yu Wu, and Kai-Tai Song. "A Study on Speech Recognition Control for a Surgical Robot". In: *IEEE Transactions on Industrial Informatics* PP (Nov. 2016), pp. 1–1. DOI: 10.1109/TII.2016.2625818.

[70] M. Stenman. "Automatic speech recognition An evaluation of Google Speech". In: 2015.

[71] IBM. *Speech Recognition*. Accessed: 2020-12-03. Sept. 2020. URL: http://www.speech.sri.com/projects/GALE.

[72] Ayaz Keerio et al. "On preprocessing of speech signals". In: 2008.

[73] Urmila Shrawankar. "Noise Estimation and Noise Removal Techniques for Speech Recognition in Adverse Environment". In: (Jan. 2010), pp. 336–342.

[74] J. Picone, M. A. Johnson, and W. T. Hartwell. "Enhancing the performance of speech recognition with echo cancellation". In: *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*. 1988, 529–532 vol.1. DOI: 10.1109/ICASSP.1988.196636.

[75] Adil Bakri et al. "Automatic Speech Recognition for VoIP with Packet Loss Concealment". In: Oct. 2015.

[76] R. Prabhavalkar et al. "Automatic gain control and multi-style training for robust small-footprint keyword spotting with deep neural networks". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 4704–4708. DOI: 10.1109/ICASSP.2015.7178863.

[77] Sabur Alim and Nahrul Khair Alang Md Rashid. "Some Commonly Used Speech Feature Extraction Algorithms". In: (Dec. 2018). DOI: 10.5772/intechopen.80419.

[78] Neil Zeghidour et al. *Learning Filterbanks from Raw Speech for Phone Recognition*. 2018. arXiv: 1711.01161 [cs.CL].

[79] Khe Chai Sim, Petr Zadrazil, and Françoise Beaufays. *An Investigation Into On-device Personalization of End-to-end Automatic Speech Recognition Models*. 2019. arXiv: 1909.06678 [eess.AS].

[80] Igor Macedo Quintanilha. "End-to-End Speech Recognition Applied to Brazilian Portuguese Using Deep Learning". PhD thesis. Mar. 2017. DOI: 10.13140/RG.2.2.11437.59367.

[81] Geoffrey Hinton et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". In: *Signal Processing Magazine, IEEE* 29 (Nov. 2012), pp. 82–97. DOI: 10.1109/MSP.2012.2205597.

[82] Jan Chorowski et al. *Attention-Based Models for Speech Recognition*. 2015. arXiv: 1506.07503 [cs.CL].

[83] Sean Naren. *deepspeech.pytorch*. https://https://github.com/SeanNaren/deepspeech.pytorch. Accessed: 2020-11-03. 2017.

[84] Joseph Walsh et al. "Deep Learning vs. Traditional Computer Vision". In: Apr. 2019. ISBN: 978-981-13-6209-5. DOI: 10.1007/978-3-030-17795-9_10.

[85] Tom Young et al. "Recent Trends in Deep Learning Based Natural Language Processing [Review Article]". In: *IEEE Computational Intelligence Magazine* 13 (Aug. 2018), pp. 55–75. DOI: 10.1109/MCI.2018.2840738.

[86] *1000x Faster Data Augmentation*. https://bair.berkeley.edu/blog/2019/06/07/data_aug/. Accessed: 2020-07-19.

[87] Alexander B. Jung et al. *imgaug*. https://github.com/aleju/imgaug. Accessed: 2020-02-01. 2020.

[88] *SpecAugment: A New Data Augmentation Method for Automatic Speech Recognition*. https://ai.googleblog.com/2019/04/specaugment-new-data-augmentation.html. Accessed: 2020-07-19.

[89] Audio Check. *Noise*. Accessed: 2020-10-23. 2009. URL: https://www.audiocheck.net/testtones_index.php.

[90] G. Saon et al. "Sequence Noise Injected Training for End-to-end Speech Recognition". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 6261–6265. DOI: 10.1109/ICASSP.2019.8683706.

[91] Shi Yin et al. "Noisy training for deep neural networks in speech recognition". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2015 (Dec. 2015). DOI: 10.1186/s13636-014-0047-0.

[92] Nelson Yalta et al. *CNN-based MultiChannel End-to-End Speech Recognition for everyday home environments*. 2019. arXiv: 1811.02735 [eess.AS].

[93] Chaim Baskin et al. *NICE: Noise Injection and Clamping Estimation for Neural Network Quantization*. 2018. arXiv: 1810.00162 [cs.CV].

[94] Adnan Siraj Rakin, Zhezhi He, and Deliang Fan. *Parametric Noise Injection: Trainable Randomness to Improve Deep Neural Network Robustness against Adversarial Attack*. 2018. arXiv: 1811.09310 [cs.LG].

[95] T. Ko et al. "Audio augmentation for speech recognition". In: *INTERSPEECH*. 2015.

[96] Michael Seltzer and Alex Acero. "Training Wideband Acoustic Models Using Mixed-Bandwidth Training Data for Speech Recognition". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 15 (Feb. 2007), pp. 235–245. DOI: 10.1109/TASL.2006.876774.

[97] Pedro Moreno and Richard Stern. "Source of Degradation of Speech Recognition in the Telephone Network". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on* 1 (Apr. 1994), pp. 109–112. DOI: 10.1109/ICASSP.1994.389343.

[98] Jinyu Li et al. "Improving wideband speech recognition using mixed-bandwidth training data in CD-DNN-HMM". In: Dec. 2012, pp. 131–136. ISBN: 978-1-4673-5125-6. DOI: 10.1109/SLT.2012.6424210.

[99] Daniel Park et al. "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition". In: Sept. 2019, pp. 2613–2617. DOI: 10.21437/Interspeech.2019-2680.

[100] Arun Narayanan et al. *Toward domain-invariant speech recognition via large scale training*. 2018. arXiv: 1808.05312 [cs.CL].

[101] Dipanjan Sarkar. "A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning". In: Accessed: 2020-10-20. 2018.

[102] Vassil Panayotov et al. "Librispeech: an ASR corpus based on public domain audio books". In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 5206–5210.

[103] Anthony Rousseau, Paul Deléglise, and Yannick Estève. "Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks". In: *Conference on Language Resources and Evaluation (LREC)*. 2014.

[104] VoxForge. *VoxForge*. http://www.voxforge.org/. Accessed on 18/11/2020. 2006.

[105] Mozilla. *Common Voice*. http://commonvoice.mozilla.org/. Accessed on 18/11/2020. 2019.

[106] Livribox. *Livribox*. http://www.librivox.org/. Accessed on 18/11/2020. 2005.

[107] Anthony Rousseau, Paul Deléglise, and Yannick Estève. "TED-LIUM: an Automatic Speech Recognition dedicated corpus". In: *Conference on Language Resources and Evaluation (LREC)*. 2012, pp. 125–129.

[108] Paulo Martins. *Fala Bracarense*. http://cehum.ilch.uminho.pt/fala_bracarense. Accessed on 18/11/2020. 2014.

[109] RTP. *RTP Play*. https://www.rtp.pt/play/. Accessed on 12/10/2020.

[110] Leonard Richardson. "Beautiful soup documentation". In: *April* (2007).

[111] SeleniumHQ. *Selenium*. https://github.com/SeleniumHQ/selenium. Accessed: 2020-05-12. 2013.

[112] James Robert. *pydub*. https://github.com/jiaaro/pydub. Accessed: 2020-11-23. 2011.

[113] Linguateca. *CetemPublico*. https://www.linguateca.pt/cetempublico/. Accessed on 18/11/2020. 2000.

[114] altavatarindia. *deepspeech-releases*. https://github.com/altavatarindia/deepspeech-releases/releases/tag/1-0-0-stable. Accessed: 2020-11-03. 2020.

[115] Igor M. Quintanilha, Luiz W. P. Biscainho, and Sergio L. Netto. "A new automatic speech recognizer for Brazilian Portuguese based on deep neural networks and transfer learning". In: *Congreso Latinoamericano de la AES* (2018).

[116] Kenneth Heafield. "KenLM: Faster and Smaller Language Model Queries". In: *Proceedings of the Sixth Workshop on Statistical Machine Translation*. WMT '11. Edinburgh, Scotland: Association for Computational Linguistics, 2011, pp. 187–197. ISBN: 9781937284121.

[117] Peilu Wang et al. "A New Word Language Model Evaluation Metric for Character Based Languages". In: 8202 (Jan. 2013), pp. 315–324. DOI: 10.1007/978-3-642-41491-6_29.

[118] Dario Amodei et al. "Deep Speech 2: End-to-End Speech Recognition in English and Mandarin". In: Dec. 2015.

A

APPENDICES

A.1 BASE TRAINING CONFIGURATION (TRAIN_CONFIG.PY)

```
from dataclasses import dataclass, field
from typing import Any, List

from deepspeech.pytorch.enums import DistributedBackend, SpectrogramWindow,
    RNNType
from omegaconf import MISSING

defaults = [
    -"optim": "sgd"",
    -"model": "bidirectional"",
    -"checkpointing": "file""
]

@dataclass
class TrainingConfig:
    no_cuda: bool = False  # Enable CPU only training
    finetune: bool = False  # Fine-tune the model from checkpoint "
        continue_from"
    seed: int = 123456  # Seed for generators
    dist_backend: DistributedBackend = DistributedBackend.nccl  # If using
        distribution, the backend to be used
    epochs: int = 50  # Number of Training Epochs

@dataclass
class SpectConfig:
    sample_rate: int = 16000  # The sample rate for the data/model features
    window_size: float = .02  # Window size for spectrogram generation (seconds
        )
    window_stride: float = .01  # Window stride for spectrogram generation (
        seconds)
    window: SpectrogramWindow = SpectrogramWindow.hamming  # Window type for
        spectrogram generation
```

```
@dataclass
class AugmentationConfig:
    speed_volume_perturb: bool = False  # Use random tempo and gain
        perturbations.
    spec_augment: bool = False  # Use simple spectral augmentation on mel
        spectograms.
    noise_dir: str = './noise'  # Directory to inject noise into audio. If
        default, noise Inject not added
    noise_prob: float = 0.4  # Probability of noise being added per sample
    noise_min: float = 0.0  # Minimum noise level to sample from. (1.0 means
        all noise, not original signal)
    noise_max: float = 0.5  # Maximum noise levels to sample from. Maximum 1.0


@dataclass
class DataConfig:
    train_manifest: str = 'data/train.csv'
    val_manifest: str = 'data/validate.csv'
    batch_size: int = 16  # Batch size for training
    num_workers: int = 4  # Number of workers used in data-loading
    labels_path: str = 'labels.json'  # Contains tokens for model output
    spect: SpectConfig = SpectConfig()
    augmentation: AugmentationConfig = AugmentationConfig()

@dataclass
class BiDirectionalConfig:
    rnn_type: RNNType = RNNType.lstm  # Type of RNN to use in model
    hidden_size: int = 1024  # Hidden size of RNN Layer
    hidden_layers: int = 5  # Number of RNN layers

@dataclass
class UniDirectionalConfig(BiDirectionalConfig):
    lookahead_context: int = 20  # The lookahead context for convolution after
        RNN layers

@dataclass
class OptimConfig:
    learning_rate: float = 3e-4  # Initial Learning Rate
    learning_anneal: float = 1.1  # Annealing applied to learning rate after
        each epoch
    weight_decay: float = 1e-5  # Initial Weight Decay
    max_norm: float = 400  # Norm cutoff to prevent explosion of gradients

@dataclass
class SGDConfig(OptimConfig):
    momentum: float = 0.9
```

```
@dataclass
class AdamConfig(OptimConfig):
    eps: float = 1e-8  # Adam eps
    betas: tuple = (0.9, 0.999)  # Adam betas

@dataclass
class CheckpointConfig:
    continue_from: str = ''  # Continue training from checkpoint model
    checkpoint: bool = True  # Enables epoch checkpoint saving of model
    checkpoint_per_iteration: int = 0  # Save checkpoint per N number of
        iterations. Default is disabled
    save_n_recent_models: int = 10  # Max number of checkpoints to save, delete
         older checkpoints
    best_val_model_name: str = 'deepspeech_final.pth'  # Name to save best
        validated model within the save folder
    load_auto_checkpoint: bool = False  # Automatically load the latest
        checkpoint from save folder

@dataclass
class FileCheckpointConfig(CheckpointConfig):
    save_folder: str = 'models/'  # Location to save checkpoint models

@dataclass
class GCSCheckpointConfig(CheckpointConfig):
    gcs_bucket: str = MISSING  # Bucket to store model checkpoints e.g bucket-
        name
    gcs_save_folder: str = MISSING  # Folder to store model checkpoints in
        bucket e.g models/
    local_save_file: str = './local_checkpoint.pth'  # Place to store temp file
         on disk

@dataclass
class VisualizationConfig:
    id: str = 'DeepSpeech training'  # Name to use when visualizing/storing the
         run
    visdom: bool = False  # Turn on visdom graphing
    tensorboard: bool = False  # Turn on Tensorboard graphing
    log_dir: str = 'visualize/deepspeech_final'  # Location of Tensorboard log
    log_params: bool = False  # Log parameter values and gradients

@dataclass
class ApexConfig:
    opt_level: str = 'O1'  # Apex optimization level, check https://nvidia.
        github.io/apex/amp.html for more information
    loss_scale: int = 1  # Loss scaling used by Apex. Default is 1 due to warp-
        ctc not supporting scaling of gradients
```

```
@dataclass
class DeepSpeechConfig:
    defaults: List[Any] = field(default_factory=lambda: defaults)
    optim: Any = MISSING
    model: Any = MISSING
    checkpointing: Any = MISSING
    training: TrainingConfig = TrainingConfig()
    data: DataConfig = DataConfig()
    augmentation: AugmentationConfig = AugmentationConfig()
    apex: ApexConfig = ApexConfig()
    visualization: VisualizationConfig = VisualizationConfig()
```

A.2 Base Inference Configuration (inference_config.py)

```
from dataclasses import dataclass
from deepspeech_pytorch.enums import DecoderType

@dataclass
class LMConfig:
    decoder_type: DecoderType = DecoderType.greedy
    lm_path: str = '/lm/cp.bin' # Path to an (optional) kenlm language model
        for use with beam search (req'd with trie)
    top_paths: int = 1 # Number of beams to return
    alpha: float = 0.0 # Language model weight
    beta: float = 0.0 # Language model word bonus (all words)
    cutoff_top_n: int = 40 # Cutoff_top_n characters with highest probs in
        vocabulary will be used in beam search
    cutoff_prob: float = 1.0 # Cutoff probability in pruning,default 1.0, no
        pruning.
    beam_width: int = 10 # Beam width to use
    lm_workers: int = 4 # Number of LM processes to use

@dataclass
class ModelConfig:
    use_half: bool = True # Use half precision. This is recommended when using
         mixed-precision at training time
    cuda: bool = True
    model_path: str = ''

@dataclass
class InferenceConfig:
    lm: LMConfig = LMConfig()
```

```
    model: ModelConfig = ModelConfig()

@dataclass
class TranscribeConfig(InferenceConfig):
    audio_path: str = '' # Audio file to predict on
    offsets: bool = False  # Returns time offset information

@dataclass
class EvalConfig(InferenceConfig):
    test_manifest: str = '' # Path to validation manifest csv
    verbose: bool = True  # Print out decoded output and error of each sample
    save_output: str = '' # Saves output of model from test to this file_path
    batch_size: int = 20  # Batch size for testing
    num_workers: int = 4

@dataclass
class ServerConfig(InferenceConfig):
    host: str = '0.0.0.0'
    port: int = 8888
```