

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO  
CENTRO TECNOLÓGICO  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA  
DISSERTAÇÃO DE MESTRADO



Lucas Grigoletto Scart

AUTOMATIC SPEECH RECOGNITION IN  
PORTUGUESE APPLIED TO RADIO  
COMMUNICATION

Vitória-ES

2024

Lucas Grigoletto Scart

**AUTOMATIC SPEECH RECOGNITION IN  
PORTUGUESE APPLIED TO RADIO  
COMMUNICATION**

Dissertation submitted to the Graduate Program  
in Electrical Engineering from the Technologi-  
cal Center of the Federal University of Espírito  
Santo as a partial requirement for obtaining a  
Master's Degree in Electrical Engineering

Vitória-ES

2024

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

S285a Scart, Lucas Grigoletto, 1997-  
Automatic speech recognition in portuguese applied to radio communication / Lucas Grigoletto Scart. - 2024.  
84 f. : il.

Orientadora: Raquel Frizera Vassallo.  
Dissertação (Mestrado em Engenharia Elétrica) -  
Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Reconhecimento automático da voz. 2. Redes neurais (Computação). I. Vassallo, Raquel Frizera. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 621.3

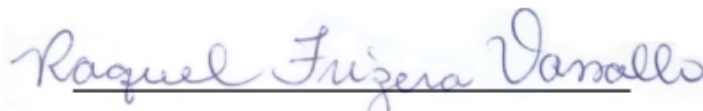
---

Lucas Grigoletto Scart

## **AUTOMATIC SPEECH RECOGNITION IN PORTUGUESE APPLIED TO RADIO COMMUNICATION**

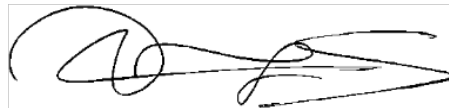
Dissertation submitted to the Graduate Program in Electrical Engineering from the Technological Center of the Federal University of Espírito Santo as a partial requirement for obtaining a Master's Degree in Electrical Engineering.

Work approved. Brazil, March 6th, 2024:



**Profa. Dra. Raquel Frizera Vassallo**

Federal University of Espírito Santo  
Advisor



**Prof. Dr. Jorge Leonid Aching Samatelo**

Federal University of Espírito Santo  
Internal Examiner



**Prof. Dr<sup>a</sup> Mariana Rampinelli Fernandes**

Federal Institute of Espírito Santo  
External Examiner



**Prof. Dr. Filipe Wall Mutz**

Federal University of Espírito Santo  
Internal Examiner

Brasil

2024

## RESUMO

A fala é a principal forma de comunicação utilizada entre seres humanos, de forma que o seu entendimento é um dos principais alvos do processamento de linguagem natural. O reconhecimento automático da fala, foco deste trabalho, é a capacidade de uma máquina reconhecer o conteúdo das palavras e frases numa língua falada e transformá-las num formato textual. Atualmente, métodos baseados em redes neurais profundas tem dominado a área de processamento de fala, apresentando resultados de estado da arte em múltiplas aplicações.

À medida que o campo do reconhecimento automático de fala continua a evoluir, surgem vários desafios quando se tenta adaptar modelos a novas línguas e conjuntos de dados, particularmente no contexto de gravações de comunicações via rádio, como é o caso deste estudo. Em comparação com o inglês, o português tem menos dados de fala anotados disponíveis, o que torna essencial explorar métodos para utilizar de forma eficaz dados não rotulados durante o treino. Além disso, as gravações de comunicações de rádio apresentam um grau substancial de variação no ruído de fundo e nas características do locutor, em comparação com outros conjuntos de dados de áudio. Esta variabilidade pode afetar a precisão e a robustez do modelo.

Este estudo propõe a utilização de dados anotados fora do domínio através de um método de aumento de dados para construir modelos de base. Além disso, explora-se a utilização eficaz de dados não rotulados no domínio através de técnicas de auto-treino, gerando pseudo-rótulos. Por fim, é apresentada uma receita de treinamento eficiente para escalar o treinamento de grandes modelos, minimizando os custos computacionais. Estes modelos foram depois implementados como parte de uma aplicação de processamento de voz, desenvolvida para ajudar no processo de auditoria de comunicações ferroviárias gravadas.

Ao efetuar o treino com os dados simulados, observou-se uma redução relativa de 51,7% na taxa de erro de caracteres considerando o nível de ruído mais desafiadora (SNR de 0 dB), com uma diminuição semelhante em todos os níveis de ruído quando comparado com o modelo original. Com o auto-treino usando dados no domínio, foi observada uma redução de 63,8% na taxa de erro de caracteres quando comparado com o modelo de base. Espera-se que a metodologia desenvolvida neste trabalho abra espaço para o desenvolvimento de modelos de reconhecimento de fala mais robustos com futuras aplicações em radiocomunicação.

**Palavras-chave:** Reconhecimento de fala. Redes Neurais Profundas.

## ABSTRACT

Speech is the main form of communication used between humans, and as such understanding spoken language is one of the main goals of natural language processing. Automatic speech recognition, the focus of this work, is the ability of a machine to recognize the content of words and sentences in a spoken language and transform them into a textual format. Currently, methods based on deep neural networks have dominated the field of speech processing, presenting state-of-the-art results in multiple applications.

As the field of speech recognition continues to evolve, several challenges arise when attempting to adapt models to new languages and datasets, particularly in the context of radio communication recordings, as presented in this study. Compared to English, Portuguese has less available annotated speech data, making it essential to explore methods for effectively utilizing unlabeled data during training. Additionally, radio communication recordings exhibit a substantial degree of variation in background noise and speaker characteristics compared to other audio datasets. This variability can affect the accuracy and robustness of the model.

This study proposes utilizing out-of-domain annotated data through a data augmentation method to build baseline models. In addition, we explore the effective use of unlabeled in-domain data via self-training techniques by generating pseudo-labels. Finally, we present an efficient training recipe for scaling large model finetuning while minimizing computational costs. Those models were then deployed as part of a broader speech processing application that was developed to assist in the auditing process of recorded railway communications.

When performing the training with the simulated data, it was observed a relative reduction of 51.7% in the character error rate considering the most challenging noise level (SNR of 0 dB), with a similar decrease at all noise levels when compared with the vanilla model. With self-training using in-domain data, we observe a reduction of 63.8% in character error rate when compared to the baseline model. We hope that the methodology developed in this work may open space to develop more robust speech recognition models with future applications in radio communication.

**Keywords:** Speech recognition; Deep Neural Networks.

## LIST OF FIGURES

Figure 1 – Overview of a classic speech recognition system. . . . .	26
Figure 2 – Overview of a modern speech recognition system. . . . .	27
Figure 3 – QuartzNet BxR architecture. . . . .	35
Figure 4 – Overview of the proposed methodology for training the acoustic model. . .	42
Figure 5 – Simplified model of speech transmission over a radio communication system in the presence of additive noise. . . . .	43
Figure 6 – Overview of the voice communication analysis system. . . . .	48
Figure 7 – Speech segments with corresponding information extracted via multiple models	50
Figure 8 – Machine learning system lifecycle . . . . .	54
Figure 9 – Training and validation loss from different recipes. . . . .	64

## LIST OF TABLES

Table 1 – Speech quality dimensions as estimated by NISQA. Value between 1.0 and 5.0. Higher is better. . . . .	58
Table 2 – Example of character error rate calculation. . . . .	60
Table 3 – Example of character error rate at different levels. . . . .	60
Table 4 – Multiple training recipes compared . . . . .	64
Table 5 – Comparison of Character Error Rates (in %) on the Commonvoice Test Set .	66
Table 6 – Character error rate on the augmented commonvoice test set (without frequency shift). . . . .	66
Table 7 – Character error rate on the augmented commonvoice test set (with frequency shift) . . . . .	67
Table 8 – Character error rate on the private test set, out-of-domain training. . . . .	67
Table 9 – Character error rate on the private test set, in-domain training. . . . .	68



## LIST OF ABBREVIATIONS AND ACRONYMS

ACV	Voice communication analysis
API	Application Programming Interface
ASR	Automatic Speech Recognition
AWGN	Additive White Gaussian Noise
BERT	Bidirectional Encoder Representations Transformers
BPE	Byte-Pair-Encoding
BRSD	Brazilian Portuguese Speech Dataset
CER	Character Error Rate
CNN	Convolutional Neural Network
CORAA	Corpus of Annotated Audios
CPU	Central Processing Unit
CTC	Connectionist Temporal Classification
DevOps	Development and Operations
FFT	Fast Fourier Transform
FM	Frequency Modulation
GSM	Groupe Special Mobile
GNU	GNU's Not Unix
GPU	Graphics Processing Unit
HF	High Frequency
ITU	International Telecommunication Union
ITU-T	ITU's Telecommunication Standardization Sector
LAS	Listen Attend Spell
LoRA	Low-Rank Adaptation

MFCC	Mel-Frequency Cepstral Coefficients
ML	Machine Learning
MLOps	Machine Learning Operations
MLS	Multilingual LibriSpeech
NBFM	Narrowband Frequency Modulation
PL	Pseudo-label
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
SQL	Structured Query Language
TPU	Tensor Processing Unit
UFES	Universidade Federal do Espírito Santo
UI	User Interface
UHF	Ultra High Frequency
VAD	Voice Activity Detection
VHF	Very High Frequency
WER	Word Error Rate

## LIST OF SYMBOLS

<b>X</b>	Input speech frames
<b>Y</b>	Transcription
<b>W</b>	Target output tokens
<b>F</b>	Sequence of phonemes
$P(\mathbf{W})$	Language model
$P(\mathbf{X} \mathbf{W})$	Acoustic model
$P(\mathbf{F} \mathbf{W})$	Pronunciation model
$f$	frequency
$A_{\mathbf{X},\mathbf{Y}}$	Set of alignments between <b>X</b> and <b>Y</b>
$\alpha$	Output token

# CONTENTS

<b>1</b>	<b>INTRODUCTION</b>	<b>13</b>
1.1	Motivation and Research Object	13
1.2	Problematic	16
1.3	Hypothesis	17
1.4	Proposed Solutions	18
1.5	Objectives	18
1.6	Materials and Methods	19
1.7	Contributions	20
1.8	Structure	20
<b>2</b>	<b>THEORETICAL BACKGROUND</b>	<b>23</b>
2.1	Automatic Speech Recognition	23
2.1.1	Characteristics of Speech Signals	24
2.1.2	Architecture of a Speech Recognition System	25
2.1.3	Advanced Architectures	32
2.2	Text Processing	33
2.3	End-to-End Speech Recognition Models	34
2.3.1	QuartzNet	34
2.3.2	Wav2vec	35
2.3.3	Model Adapters	36
2.4	Robust Speech Recognition	38
<b>3</b>	<b>PROPOSAL</b>	<b>41</b>
3.1	Simulation of the Acoustic Characteristics of Radio Communication	42
3.2	Training Recipes	45
3.3	Pseudo-labeling	46
3.4	Production Deployment	47
3.4.1	Voice Communication Analysis System	47
3.4.2	Frontend	49
3.4.3	Backend	50
3.4.4	Machine Learning Operations	51
3.4.5	Testing Machine Learning Systems	53
<b>4</b>	<b>EXPERIMENTS</b>	<b>56</b>
4.1	Speech Datasets	56
4.2	Code Libraries	58

4.3	<b>Evaluation Metrics</b>	<b>59</b>
4.4	<b>Implementation Details</b>	<b>61</b>
4.4.1	Optimized Training with Scarce Resources	61
4.4.2	Model Specific Regularization Techniques	61
4.5	<b>Results and Discussion</b>	<b>62</b>
4.5.1	Comparing training recipes	62
4.5.2	Baseline model using radio data augmentation	65
4.5.3	Self-training via pseudo-labeling	67
5	<b>CONCLUSION</b>	<b>69</b>
	<b>BIBLIOGRAPHY</b>	<b>70</b>
	<b>ANNEX</b>	<b>77</b>
	<b>ANNEX A – LEARNING PARADIGMS</b>	<b>78</b>
	<b>ANNEX B – DEEP NEURAL NETWORKS</b>	<b>80</b>
B.1	<b>Convolutional Neural Networks</b>	<b>80</b>
B.2	<b>Transformer</b>	<b>81</b>
B.2.1	Attention Mechanism	81
B.2.2	Multi-Head Attention	82
B.2.3	Positional Embeddings	82
B.2.4	Transformer Blocks	83

# 1 INTRODUCTION

## 1.1 Motivation and Research Object

The main form of communication between humans is speech, and therefore, understanding spoken language is one of the primary goals of natural language processing. Signals derived from speech can provide various types of information depending on the desired application. According to Nassif et al. (2019), these signals contain information that enables the identification of speaker, language, accent among speakers of the same language, emotional state, health condition, age, and gender. Furthermore, machines can perform automatic speech recognition, allowing them to recognize the content of words and sentences in a spoken language and convert them into a textual format.

The information extracted using a speech recognition system can be used to implement human-machine interfaces in the form of voice-driven information processing systems (HE; DENG, 2013). Some typical tasks include spoken language translation, spoken language understanding, and information retrieval. Spoken language translation involves translating the source speech into a different language to facilitate communication between people who speak different languages.

Spoken language understanding is concerned with identifying the domain to which spoken utterances<sup>1</sup> belong, parsing the semantic information contained within it and extracting relationships between different pieces of information. This involves analyzing the meaning of individual words and sentences in spoken language and understanding how those words and sentences relate to each other within a particular context. Some examples of spoken language understanding tasks include identifying the topic of a conversation, recognizing named entities such as people or organizations, and extracting information about relationships between entities.

Information retrieval considers the input utterance as a query that ranks a list of documents by relevance. The goal is to return the most highly rated documents in the top-rated ones. In addition to these specific applications, spoken language processing is also used in many other areas, including text-to-speech synthesis and emotion detection. Spoken language processing techniques are used in various industries, such as healthcare, finance, education, and customer service, to name a few. These techniques help to improve the accuracy and efficiency of human-machine interactions, enabling more natural and intuitive communication between humans and technology.

A significant amount of development is required to reach the point where speech recognition is a technology that enables voice-driven applications. Earlier work was based on a combination

---

<sup>1</sup> An utterance is the smallest unit of continuous speech, starting and ending with a clear pause.

of classic statistical methods for recognizing basic sound units in phonemes and rule-based systems that attempted to model linguistic information using expert knowledge. The mixture of acoustic, phonetic, and linguistic information at different levels was the basis for primitive speech recognition systems, which started with a limited vocabulary and a limited number of speakers in controlled recording scenarios. These systems required substantial manual effort from linguist experts to encode phonetic and language rules, resulting in a lack of robustness. Later, some manual rules were replaced by automated, learning-based systems. Statistical methods such as the hidden Markov model dominated the field for a long time, with the most popular implementation being the Kaldi toolkit (POVEY et al., 2011).

Integrating deep neural networks into speech recognition started with acoustic model development, using a hybrid model constituted by convolutional neural networks and hidden Markov models (CNN-HMM) (ABDEL-HAMID et al., 2012). A few years later, Deep Speech (HANNUN et al., 2014) was the first end-to-end speech recognition system based on deep neural networks that could scale to state-of-the-art performance on a widely studied dataset by exploiting the capacity of neural networks to learn complex mappings from large datasets. Deep Speech is an end-to-end speech recognition model that predicts the probability of each character in a sequence given frequency bin information from the input speech and the history of predicted characters.

It uses novel data synthesis techniques to artificially increase the amount of training data available for the model alongside an optimized training process that can be executed on multiple GPUs. The increased training scale allowed for the development of a simpler, more effective model that is less prone to errors in noisy environments compared to traditional systems.

Deep Speech-2 (AMODEI et al., 2016) then scaled the model to learn English or Mandarin Chinese, two markedly distinct languages, using the same architecture. The breakthrough was implementing high-performance computing techniques to achieve a  $7\times$  speedup over previous systems, enabling a faster iteration speed over experiments with deeper models. Beyond achieving state-of-the-art, the developed model was competitive with the transcription from human workers. Wav2Letter (COLLOBERT; PUHRSCHE; SYNNAEVE, 2016) presented a simpler end-to-end model, using only convolutional layers and a novel automatic segmentation criterion for training.

In addition to the emergence and popularization of deep learning models for speech recognition, many datasets were created to provide different data types for training and comparing such models. After that, more complex and automatic speech recognition systems began to be developed, including systems for the Portuguese language. Based on the Kaldi toolkit, which uses traditional approaches compared to deep neural networks, the authors Batista, Dias e Sampaio Neto (2018) have developed base models for the Portuguese language using several open and paid datasets,

the largest among them being CETUC. Using models based on Markov chains and Gaussian mixtures, the authors achieved a word error rate of 6.5%. As a proof of concept, a deep neural network was trained to replace one of the components of the system, reducing the error rate to 4.5%. However, there was still a reliance on the phonetic dictionary and language models from traditional systems.

A significant milestone in developing deep neural networks that perform end-to-end speech recognition in Portuguese is the work of Quintanilha (2017). Using a combination of smaller datasets, totaling approximately 13 hours, they trained an end-to-end model, which achieved a character error rate of 25.13%. Subsequently, Quintanilha, Netto e Biscainho (2020) expanded their earlier work, building a dataset with 158 hours in total. Using an architecture based on DeepSpeech-2 and language models for post-processing the predictions, the authors achieved an error rate per character of 10.49%.

With the emergence of Wav2vec 2.0, the first work focused on Portuguese was Gris et al. (2021). The study achieved a word error rate of 34% using only one hour of labeled data when evaluated with the commonvoice test suite. This work was expanded on Gris et al. (2022), using 427 hours of audio for training the model achieved by joining different datasets. This work presented an average word error rate of 12.4% when evaluated over seven different datasets. By post-processing the predictions with language models, the average error rate decreased to 10.5%.

Simultaneously with the publication of the CORAA dataset, the work of Junior et al. (2021) trained a baseline model using the new data together with Wav2vec 2.0. Compared to the result obtained by Gris et al. (2022), the trained model had a higher error rate when evaluated over the original *commonvoice* test set; however, it showed better results with the new data.

While trying to transcribe speech by any speaker in any environment is still an open research problem, developing large-scale models with thousands of hours of speech data has advanced the technology to a point where its practical application is now viable. Unfortunately, not all languages possess the labeled datasets necessary to achieve this level of success, which presents an unfortunate limitation in the field.

As the field of speech recognition continues to evolve, several challenges arise when attempting to adapt models to new languages and datasets, particularly in the context of radio communication recordings, as presented in this study. Compared to English, Portuguese has less available annotated speech data, which highlights the importance of exploring methods for effectively utilizing unlabeled data during training. Additionally, radio communication recordings exhibit a substantial degree of variation in background noise and speaker characteristics compared to other audio datasets. This variability can affect the accuracy and robustness of the model.



As the size of the audio dataset increases, the computational resources required to train the model become a significant limitation, there is a compelling need to develop efficient algorithms that can handle large-scale models while minimizing computational costs. Furthermore, radio communication recordings have unique requirements for speech recognition due to their nature, and it is crucial to comprehend how the model can be tailored to these specific conditions while still maintaining high accuracy.

The speech recognition system developed in this work results from a partnership project between Vale S.A. and the Federal University of Espírito Santo (UFES). The software under development aims to assist in auditing radio communications between train drivers and traffic controllers on the company's railways.

For the safe operation of railroads, traffic controllers need to exchange information constantly with each train driver through a radio communication system. A series of protocols dictate communication, including identification protocols, specific words that denote the end of a message, and confirmation of commands before their execution. To enforce these protocols and verify they were followed all communication that happens by radio is recorded and audited afterward.

As adopted at the beginning of the partnership, with the work of Ribeiro (2020), using data extracted from recordings and labeled manually considerably increased the quality of the model obtained, with the best result consisting of mixed training with a large volume of open data in conjunction with data from the target domain. However, the best results still had a character error rate of 40%, making the model infeasible to assist in auditing the communication. Therefore, there is a need to develop automatic and application-specific speech recognition models for radio communication, which will feed the developed software and assist in the audits carried out.

## **1.2 Problematic**

Despite the significant advance provided by neural models, one of the criticisms for their use is the large volume of data required for learning to occur (MARCUS, 2018). Since there is no mechanism for learning abstractions through explicit definitions, it takes thousands, in some cases even millions, of examples for the model to learn the desired mapping. Recent research has achieved a significant breakthrough in this area, allowing the efficient use of unlabeled data for learning abstractions. In natural language processing and computer vision, the self-supervised learning paradigm allows the pre-training of models that extract relevant features from unlabeled data and later utilize them for various target tasks, as reported in the following works (PETERS et al., 2018; RADFORD et al., 2018; DEVLIN et al., 2018; HÉNAFF et al., 2019; BACHMAN;

HJELM; BUCHWALTER, 2019; MISRA; MAATEN, 2019; HE et al., 2019; CHEN et al., 2020).

When applied to speech recognition, the main work in this research area is the Wav2vec 2.0 methodology (BAEVSKI et al., 2020), which is based on a two-stage training process. In the first step, a large volume of unlabeled data is used with a self-supervised learning paradigm. The model is constrained to produce discrete representations of audio snippets while contextualizing the representations based on the complete audio. Thus, given a set of representations extracted from the audio, incomplete in time, the target task of the model is to predict which other discrete representations correctly fill the missing moments to reach the complete representation of the audio. In the second step, the learned representations are mapped to a target vocabulary using a linear projection, and labeled data guides the model toward understanding the relationship between the audio tokens and the corresponding transcription.

Regardless of the improvement achieved with unlabeled data and the training of large models that learn shared representations between different languages, there is still a barrier to the practical use of speech recognition in Portuguese, particularly within the specific context of the target application. This happens because the data used for training contains mostly recordings of prepared readings and speeches. These are audios recorded in controlled environments with high-quality microphones, containing no noise or external interference, and a calm tone of voice and clear diction.

In contrast, audio extracted from radio communication has a high level of ambient noise, lossy data compression used to achieve a long communication distance at the cost of quality, fast speech with dubious diction, and eventual overlap between different speakers.

### 1.3 Hypothesis

Given the challenges associated with transcribing speech, particularly in the context of radio communication recordings, we have proposed several hypotheses to address these issues:

- Radio speech communication, with its restricted vocabulary and structured format, may be more amenable to speech recognition than general audio transcription despite the challenges posed by a noisy environment.
- Data augmentation using clean Portuguese speech datasets can simulate the settings encountered in radio communication recordings. This technique could positively impact on the target task by expanding the available dataset without requiring additional annotations.

- The use of deep learning models pretrained with diverse data from various languages has shown promise in reducing the need for labeled data in the target domain. By leveraging these pretrained models, it may become feasible to deploy speech recognition technology in Portuguese radio communication recordings with a lower requirement for annotated data.

## 1.4 Proposed Solutions

This work proposes a methodology to adapt existing speech recognition models, aiming to improve its performance and robustness in conditions encountered in radio communication.

In this context, our study proposes utilizing out-of-domain annotated data through a data augmentation method to build baseline models. In addition, we explore the effective use of unlabeled in-domain data via self-training techniques by generating pseudo-labels. Finally, we present an efficient training recipe for scaling large model finetuning while minimizing computational costs. By doing so, we aim to enhance accuracy and robustness when executing speech recognition tasks in Portuguese using radio communication recordings.

## 1.5 Objectives

The main objective is to adapt an automatic speech recognition model based on deep neural networks to work in the context of radio communications. Consequently, the adapted model can be used in the currently developing auditing software.

To achieve the main objective, the following specific objectives need to be met:

- Obtain public domain datasets that contain audio in Portuguese and perform label standardization.
- Study the main characteristics of radio communication systems and perform computer simulations of the operation of these systems.
- Study the training methodologies of deep neural networks for robust speech recognition to domain variations.
- Systematically evaluate the results obtained from our proposed model and compare them with those presented in existing literature.

- Validate the model utilizing a privately-held dataset comprising samples of radio communication speech that were gathered by the partner company.
- Integrate the developed model with the auditing software.

## 1.6 Materials and Methods

The present work conducts an applied research with exploratory-descriptive objectives, analyzes data quantitatively, and employ bibliographic and experimental procedures.

The initial stage of the project began with a bibliographical survey of the works developed in the field of automatic speech recognition. The main objective was to identify the deep learning architectures and training methodologies used by modern speech recognition systems, considering both broader works using English as the target language and works focused on Portuguese. We kept a constant bibliographical survey in parallel with all the other tasks until we wrote the present manuscript. Thus, it was possible to map the main works that emerged in this same line of research.

The next stage of development started with (SCART, 2019) and (RIBEIRO, 2020), and can be described as: *(i)* relevant speech recognition datasets in Portuguese were collected, including creating a custom dataset for the target task; *(ii)* an initial model was trained. Despite the potential of our developed model, constraints related to the quantity of data utilized and domain drift prevented its application in the target domain. As a result, we shifted our focus to investigating techniques centered on robust speech recognition during this development phase due to these limitations. Multiple deep learning models were trained, evaluated, and compared with different approaches related to speech denoising, neural network robustness, and data augmentation. The main results were summarized and presented as images, tables, and charts.

After reaching an acceptable level of performance for our collection of multiple deep learning models in the target domain, we moved on to a new stage of development: creating software that could be used by end-users. To accomplish this, we adopted an architecture that separates the frontend, backend, and model inference components. We developed these elements concurrently to streamline the process. Weekly meetings with a limited group of target users guided this development phase, where the workflows and interface components were tailored according to the expected usage and live previews. With our web application complete, users can efficiently review radio communication recordings through its intuitive interface. By combining existing deep learning models with a user-friendly software solution, we aim to facilitate effective audits of voice communications for end-users.

## 1.7 Contributions

The main contributions achieved by this dissertation include:

- a) A data augmentation technique that simulates acoustic characteristics of radio communication
- b) A recipe to efficiently adapt large acoustic models into new languages with limited amounts of labeled data
- c) An open-source library that streamlines the training of acoustic models <sup>2</sup>
- d) Acoustic and language speech recognition models focused on radio quality speech
- e) A production deployment system that enables fast auditing of radio communication recordings

In addition to this manuscript, some of the contributions cited are recorded in the form of an article, with peer reviews at a national congress:

- SCART, L. G. ; Vassallo, R. F. ; SAMATELO, J. L. A. . Aplicação de um Modelo Neural para Reconhecimento de Fala em Áudios com Características de Comunicação via Rádio. In: CBA 2022 - XXIV Congresso Brasileiro de Automática, 2022, Fortaleza. CBA 2022 - XXIV Congresso Brasileiro de Automática, 2022.

## 1.8 Structure

For a better understanding, this manuscript is structured as follows:

**Chapter 1 - Introduction:** In this chapter, we will provide the necessary justification for our research while highlighting the primary issues that arose during the study. We will also introduce the hypotheses we have formulated and outline the approaches we propose to address these problems. Additionally, we will describe the objectives of our research, which serve as a roadmap for the subsequent sections of this document. By providing a clear and concise overview of our research in this introductory section, we hope to establish a solid foundation that will guide readers through the remainder of the text.

<sup>2</sup> Available at: <https://github.com/scart97/thunder-speech>

- Chapter 2 - Theoretical background:** This chapter provide a thorough explanation of the fundamental principles and terminology that are essential for comprehending the proposals presented in our study. By explaining these concepts, we aim to facilitate an easier understanding of both the methods employed during our experiments and the outcomes obtained. This section serves as a crucial foundation upon which the remaining sections of this document will build, ensuring that readers possess the necessary background knowledge to fully grasp the subsequent content.
- Chapter 3 - Proposal:** In this chapter, we will explore the techniques employed to develop robust automatic speech recognition systems intended for use within the radio communication environment. Our methodology involves creating an initial model utilizing a mix of data from publicly available datasets along with a unique data augmentation technique. This initial model serves as a foundation upon which pseudo-labels are generated in the target domain, significantly reducing the time-consuming and costly labeling process. We will then describe our deployment system, designed to be accessible to end-users, concluding this chapter. By presenting these techniques, we aim to offer a comprehensive overview of our methods for developing robust automatic speech recognition systems specifically suited to the radio communication environment.
- Chapter 4 - Experiments:** In this chapter, we will provide a comprehensive overview of the relevant datasets, code libraries utilized, evaluation metrics, and implementation details. We will then carry out several experiments aimed at verifying the hypotheses formulated in this study. Throughout these experiments, we will provide a detailed explanation of our reasoning, describe the selected hyperparameters, and offer an analysis of the results obtained. By presenting these critical elements, we aim to offer readers a complete understanding of both the data and methods employed in this research.
- Chapter 5 - Conclusion:** In our final chapter, we will summarize the key outcomes of our proposals and the experiments conducted, highlighting which results confirm the hypotheses raised in this work. Furthermore, we will discuss any limitations that emerged during these experiments as well as potential areas for further research to advance this field of study. We will also examine the issues that remained unaddressed throughout this investigation and explain why they were not a primary focus of our study. By presenting these conclusive insights, we aim to offer readers

an in-depth understanding of both the findings obtained and the future prospects for this research area.

## 2 THEORETICAL BACKGROUND

This chapter introduces the theoretical background necessary to understand the automatic speech recognition task implemented in modern systems. An overview of speech recognition is presented first, including a description of the task and its different blocks, starting with raw speech signals and going to the final transcription. Next, we discuss text processing methods, emphasizing the techniques necessary to encode textual labels into a format understandable by the speech recognition model and the subsequent decoding of the model predictions into text. After that, the main end-to-end speech recognition models used in this work are presented, with models that use convolutional and transformer layers to build their architecture. Finally, we introduce a definition of robust speech recognition, and divide the related work into two categories based on the methodologies used: those centered around training robust models and those focused on pre-processing audio signals to mitigate noise.

### 2.1 Automatic Speech Recognition

Speech as a communication method presents a high degree of variability, and, as such, automatic speech recognition is a complex task with internal subdivisions based on the source characteristics. According to (JURAFSKY; MARTIN, 2009), the speech recognition task can vary in four dimensions. The first dimension is the size of the vocabulary used in the task, varying from a closed vocabulary, such as recognizing a set of pre-defined voice commands, to an open-ended vocabulary, such as transcribing human conversations or videos. The second dimension to consider is the intended audience of the speaker. When speaking to machines, which is the case of read speech, humans tend to speak loudly and clearly, making the recognition task easier. In contrast, conversational speech is more chaotic, with filler sounds, incomplete dictation, and potential overlap. The third dimension is the channel and noise. The quality of the recording equipment and the conditions of the recording environment can significantly impact the difficulty of speech recognition. The last dimension is accent or speaker-class characteristics. The same word can have different pronunciations depending on regional or ethnic dialects, making it harder to recognize new dialects that the system was not trained to recognize. Then, to understand speech recognition, we should start by understanding how speech is produced by humans and recorded by machines.



### 2.1.1 Characteristics of Speech Signals

The vocalization of speech occurs through variations in air pressure, generating sound waves, which can then be captured by a microphone as continuous analog signals, generating analog electric signals that require digital conversion to be processed by computers or digital systems. This analog-to-digital conversion has two steps, namely sampling and quantization. These two steps discretize the signal both in time and amplitude. The sampling process happens by measuring the amplitude at a particular time, following a constant time spacing between samples. Here, the sampling rate defines the number of samples taken per second. The quantization process involves mapping the amplitude of each sample taken to a fixed number of possibilities following a scale that buckets nearby values.

Nyquist's law dictates that to achieve proper reconstruction, one must acquire at least two samples of a sinusoidal wave for each cycle. This property limits the maximum frequency that can be measured to be half of the sample rate, creating the Nyquist frequency. Classic digital telephony uses a sampling frequency of 8000 Hz, creating a theoretical bandwidth of 4000 Hz for the speech signal. The actual target frequencies for telephony systems are those present between 300 Hz and 3400 Hz, also known as narrowband speech, and were chosen based on earlier work by Fletcher e Galt (2005) that investigates the speech intelligibility by measuring the probability of errors in phoneme understanding. Researchers discovered that this range of frequencies allows for more than 97% of all sounds to be understood and that the understanding of whole sentences was around 99% given that humans have the natural ability to interpolate missing information based on the surrounding context.

Also, based on the work of Fletcher e Galt (2005), the sampling frequency of 16000 Hz was chosen to represent wideband speech signals. With a target to cover frequencies ranging from 50 Hz up to 7000 Hz, this bandwidth is found to have a more natural and full sound. While narrowband speech can represent most of the sounds produced during conversational speech, it focuses on voiced sounds like vowels. In contrast, the extended bandwidth on wideband speech captures unvoiced sounds better, contributing to a more natural feeling.

In order to ensure accurate results during the development and validation stages of a speech recognition system, it is crucial to maintain consistent audio quality. Depending on the source of the speech signal, up-sampling or down-sampling may be required to prevent potential mismatches. Throughout this work, we assume that input speech will undergo resampling to the wideband speech sample rate before any model or feature extractor processes it. This approach allows us to capture all the variability of the human voice and subsequently guarantees results with good accuracy in the automatic speech recognition system being developed. For instance, if the input speech signal is provenient from a narrowband source sampled at 8000 Hz, we would

up-sample it to a wideband sample rate of 16000 Hz for processing by our speech recognition model. This ensures that all the necessary spectral information is captured and presented during training and testing, resulting in more accurate recognition outcomes.

### 2.1.2 Architecture of a Speech Recognition System

The main objective of a speech recognition system is to produce the most likely sequence of words given an input speech signal. This objective is achieved by treating both the input and output as sequences of discrete elements to perform the necessary computation that maps between them (HUANG et al., 2001). The input speech is windowed into a series of segments  $X = \{x_1, x_2, \dots, x_n\}$ , where each segment  $x_i$  (also known as a frame) is processed to create a prediction. Similarly, the target words are modeled with tokens representing individual letters or bigger pieces as subwords or complete units. A sequence of output tokens creates the target vector  $W = \{w_1, w_2, \dots, w_m\}$ . The learning objective of speech recognition can be expressed as follows:

$$\hat{W} = \arg \max_w P(W|X) \quad (2.1)$$

Classical speech recognition systems tried to simplify this learning objective and break the system into multiple individual, independent steps. First, applying the Bayes theorem over the probability  $P(W|X)$ :

$$P(W|X) = \frac{P(X|W)P(W)}{P(X)} \quad (2.2)$$

Two more simplifications are usually applied. First, the probability  $P(X)$  can be assumed to be the same for all predictions, as the input is fixed, so it's ignored when training the model. Second, classic speech recognition systems add another step between the input speech and output words to model the phonemes  $F$  present in the audio. This is done by expanding the probability  $P(X|W) = P(X|F)P(F|W)$ . With those modifications, the learning objective of speech recognition systems can be expressed as:

$$\hat{W} = \arg \max_w P(X|F)P(F|W)P(W) \quad (2.3)$$

This equation defines the three main parts of a classic speech recognition system (HUANG et al., 2001).  $P(X|F)$  is called the **acoustic model** which is responsible for identifying the sequence of phonemes, given the input speech. Next,  $P(F|W)$  is the **pronunciation model** that tries to map the sequence of phonemes into the most probable words. Finally,  $P(W)$  is the **language model**. Using only information from the predictions gives a score to multiple hypotheses for the same input speech, making it possible to rank them.

The traditional implementation of an speech recognition system follows from Figure 1. Given the extend of the input speech vector, a feature extracting step is employed in order to extract compact representations from each speech window. Using these features built from the input speech, the acoustic model scores the possible phonemes present at each time step. Classic speech recognition systems trained those three models independently, then during inference they were chained together by a custom decoding process.

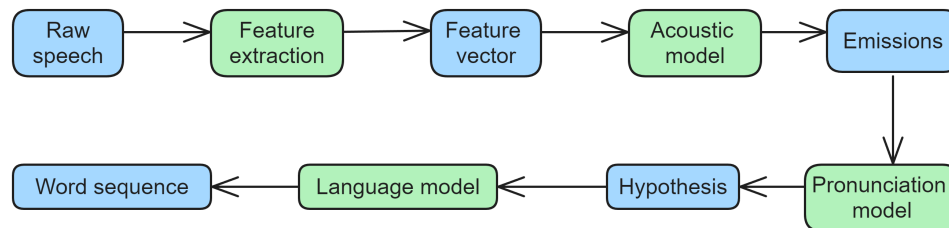


Figure 1 – Overview of a classic speech recognition system.

In the era of deep learning, there has been a significant shift in speech recognition technology. The intermediate step of modelling phonemes was abandoned, in favor of directly modelling the probability corresponding to the different hypothesis (YU; DENG, 2014; DENG; LI, 2013; HINTON et al., 2012). As many parts of the system try to be incorporated inside a single end-to-end model as possible, with modern methods taking raw speech as input and training the feature extractor as the first few layers of the model. Despite these advances, a custom decoding process remains necessary for generating predictions, and external language models still contribute to enhancing the overall system’s accuracy.

The emergence of deep learning has revolutionized the field of speech recognition, leading to a major shift in the way we approach this technology. In the past, phonemes were modeled as an intermediary step in the process, but recent advancements have eliminated this requirement altogether. Today, instead of modeling individual phonemes, we directly model the probability associated with each possible hypothesis.

This change has been made possible by the growing power and complexity of deep learning algorithms, which allow us to tackle more complex tasks than ever before. As part of this evolution, we are now working towards incorporating as many components of the system into a single end-to-end model as possible. This approach has the added benefit of reducing the number

of parameters that need to be trained and optimized separately, which can lead to improved performance and reduced computational costs.

One key development that has contributed to this trend is the use of connectionist temporal classification (CTC) models (GRAVES et al., 2006), which allow us to perform sequence-level labeling tasks directly on input sequences without requiring explicit segmentation or alignment. In CTC models, the decoding process is also part of the training process, which means that we can optimize both the feature extraction and decoding steps simultaneously. Despite these advances, a custom decoding process remains necessary for generating predictions, and external language models still contribute to enhancing the overall system's accuracy.

Figure 2 shows the simplified approach taken by end-to-end speech recognition systems. In the next sections, we will explain in more detail each of these components.

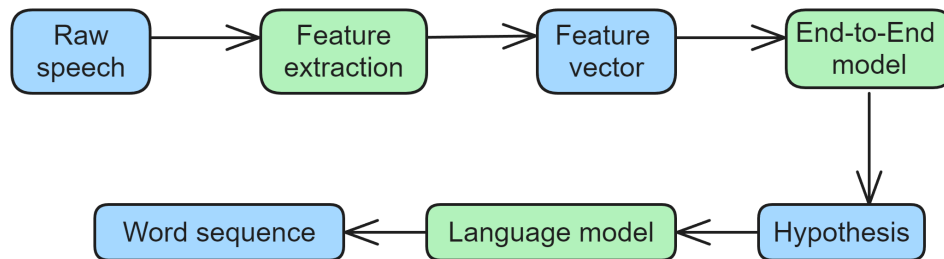


Figure 2 – Overview of a modern speech recognition system.

## Feature extraction

The first step of a speech recognition system is to extract feature vectors from the digital representation of speech. Feature extraction methods can be categorized into two groups based on manual or learned transformations.

- **Methods based in manual features.** The prevailing manual approach involves utilizing a sequence of log mel spectrum vectors. They are created by initially windowing the signal into smaller sections called frames, with a fixed window size and a shift between windows defined by a frame stride. For each frame, a windowing function is applied to reduce spectral problems caused by the discontinuous boundary, with the Hamming window being the most common one used. Then, a discrete Fourier transform is performed for each window, extracting the spectral information of each discrete frequency band represented by the magnitude and phase components. Here, it is essential to mention that the phase component is discarded because most information is contained in the magnitude component.

The current result includes the energy for each frequency band, with equal weighting across all bands. However, human hearing is less sensitive at higher frequencies. Meaningful information about formants is contained at low frequencies, which is crucial for distinguishing vowels or nasals, while the high frequencies mostly have noise related to speech production. To fix this problem, the frequency bands are adjusted following a Mel scale, which was developed based on experimental tests of the auditory system. The Mel frequency ( $mel(\bullet)$ ) is computed from the acoustic frequency  $f$  following the equation:

$$mel(f) = 1127 \ln \left( 1 + \frac{f}{700} \right) \quad (2.4)$$

This step is implemented by a filter bank that collects energy from each frequency band, with a logarithmic spread following the Mel frequencies. Finally, the log of each value is calculated (similar to the frequency response, the human response to audio amplitude is also logarithmic).

- **Methods based in learned transformations.** Contrastingly, certain models take raw audio as input and internally derive meaningful representations through various architectures. For example, SincNet (RAVANELLI; BENGIO, 2018) employs a specific parametrization for its first convolutional layer, compelling it to learn band-pass filters based on sinc functions. Additionally, time-domain filterbanks (ZEGHIDOUR et al., 2018) can be used for dynamic feature extraction, where the filters utilized in the Mel spectrum features can be trained as a parameter. Finally, models such as wav2vec (BAEVSKI et al., 2020) only use standard convolution layers to process raw audio.

Speech signals are unique in their temporal length, which presents a significant challenge when it comes to processing them for speech recognition applications. The long sequence length of input speech can make the computational complexity of downstream components prohibitively high, making feature extraction a critical step in this process.

To address this issue, subsampling has emerged as an important technique for reducing computational costs while preserving essential features of the input signal. This technique involves discarding individual frames, stacking features across multiple frames, or using an aggregation function that operates over a sliding window. By doing so, we can significantly reduce the dimensionality of the feature space and make it more manageable for downstream components.

One key advantage of subsampling is that speech signals are considered to have limited variability over time, which makes them a good candidate for this type of dimensionality reduction. By focusing on the most informative features and discarding redundant ones, we can achieve higher accuracy and lower computational costs compared to naive approaches like processing each frame independently.

## End-to-end Acoustic model

The development of deep learning techniques has led to recent speech recognition systems that aim to establish a direct relationship between the extracted feature vectors and various hypotheses. However, one significant challenge remains when considering the connection between input and output. Both input and output can change in length, and the relationship between their length can vary too. Additionally, we assume that an accurate alignment between input and output is unavailable. Consequently, any learning method utilized must be invariant to this issue.

Two techniques can primarily be employed to model this type of relationship in end-to-end systems, specifically the encoder-decoder and the connectionist temporal classification. The former involves decoupling the input processing and hypothesis generation into two distinct steps, as demonstrated by the Listen Attend Spell (LAS) model (CHAN et al., 2015).

The LAS model consists of two main components: an encoder that processes the audio input and a decoder that generates the text output. The attention mechanism (BAHDANAU; CHO; BENGIO, 2014)<sup>1</sup> acts as a “coupling” between the encoder and decoder, allowing the model to focus selectively on different parts of the audio input sequence while generating the text output. In this way, the LAS model can learn the mapping between the audio input and text output sequences without the need for precise alignment information by using the attention mechanism to selectively focus on the most relevant parts of the audio input at each step of the decoding process.

One alternative to encoder-decoder models is using a loss function designed for variable-length sequences. The connectionist temporal classification (CTC) algorithm and its corresponding loss function effectively address the lack of alignment between input and output sequences (GRAVES et al., 2006). For a given input value of  $X$ , it generates an output distribution for all possible values of  $Y$ . This distribution can be used to infer the most likely output or to assess the probability of a particular output. The fundamental concept involves treating the network’s outputs as a probability distribution over all possible label sequences conditioned on a given input sequence. The network can be trained with standard backpropagation by deriving an objective function that directly maximizes the probabilities of the correct labelings. Once trained, the network can be used as a classifier by selecting the most probable labeling for a given input sequence.

The CTC algorithm employs an innovative approach to handling misalignment between input

---

<sup>1</sup> Attention mechanisms allow neural networks to selectively focus on relevant regions of an input signal through a weighted combination of all the encoded input vectors. This dynamic process improves efficiency and accuracy in predicting outputs through context awareness and the ability to learn complex relationships between input features.

and output sequences. This involves adding a special blank token  $\varepsilon$  to the target space  $L$  of all output tokens present in the target texts. The model then outputs a sequence with the same length as the number of frames in the input. Each element of this sequence  $Y^*$  is produced by a softmax output layer, with dimensionality  $|L| + 1$ . The first  $|L|$  activations represent the probability of observing the corresponding labels at specific times, while the extra activation is the probability of observing a blank. The algorithm uses blank tokens to distinguish between natural repetition and repetition caused by sounds spanning multiple frames. During CTC decoding, the algorithm first collapses repeated tokens and removes the blank token. For example, the sequence  $[A, A, A, A]$  is collapsed into  $[A]$ , while  $[A, A, \varepsilon, A, A]$  produces  $[A, A]$ .

The loss function tries to maximize the probability that the output sequence  $Y^*$  collapses into the correct label  $Y$  when processed by the decoding algorithm. Given that multiple sequences can be collapsed into the same result, to get the probability of a specific output, given an input, it is necessary to find all possible alignments between the two. That means, for a specific label  $Y$ , find all sequences  $\alpha_t$ , with the same length of  $Y^*$ , that can be decoded to this label. Each of these sequences is a single alignment. To calculate the CTC conditional probability, the probability of all alignments is computed given the model output  $Y^*$ , as presented in the following equation:

$$P(Y|X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^T p_t(\alpha_t|X) \quad (2.5)$$

The conditional probability given by the previous equation first marginalizes over the set of valid alignments between input and output, then computes the probability for each single alignment step-by-step. It has two strong assumptions to work: that each output step is independent of the output at any other time and that the output sequence that includes the blank tokens is strictly longer than the desired sequence  $Y$ . The CTC loss requires an efficient way of calculating conditional probabilities for individual labelings. A straightforward approach involves computing the score for each alignment and summing them all up, but this becomes inefficient due to the large number of paths corresponding to a given labeling. Fortunately, dynamic programming can provide a solution. Similar to the forward-backward algorithm for hidden Markov Models (RABINER, 1989), we can merge alignments that have reached the same output at the same step. This way, the probability of the multiple alignments can be efficiently computed.

After training a speech recognition model, we aim to obtain the most likely labeling for an input speech through inference. This process is commonly referred to as decoding the predictions of the model. In practice, two main methods are employed to approximate the optimal result, given that there is no general, tractable algorithm to compute the probability of all possible alignments. The first method, called best path decoding or greedy decoding, assumes that the most probable

hypothesis will correspond to the most likely output at each time step. Although this method is trivial to compute, and the prediction can be computed after applying the CTC collapsing algorithm, it may fail to find the most probable labeling since multiple alignments can result in the same prediction. The second method used to decode predictions is an adaptation of the beam search algorithm. In contrast to the standard approach, which generates new sets of hypotheses at each input step by extending each hypothesis with all possible output characters and keeping only the top candidates, this modified version stores accumulated scores for a given prefix based on all the alignments that map to it. This allows the model to consider multiple alignments and produce more accurate decoding results.

## Language models

In order to improve predictions using an acoustic model, a language model is usually employed during the decoding phase. Given multiple hypotheses  $W^*$  generated from the same input sequence, the language model attempts to assign a probability  $P(W)$  for each hypothesis, allowing them to be ranked. Thus, it tries to assign probabilities to specific sequences of words.

The language model most frequently employed in speech recognition is the  $n$ -gram model. An  $n$ -gram represents a sequence of  $n$  words, with 2-grams referred to as bigrams and 3-grams called trigrams. This model is used to make the following operations: estimate the probability of the final word in an  $n$ -gram given the previous ones and assign probabilities to entire sequences of words.

To comprehend how it operates, let us start with computing  $P(w_n|h)$ , which represents the probability of observing a specific word  $w_n$  given some history  $h$ . One way to estimate this probability is through relative frequency counts. For instance, take an extensive corpus and count the number of times we encounter the sequence  $h$ , as well as the number of times it is followed by  $w_n$ .

By analyzing a large enough corpus of text, we can compute the frequency of each word and estimate the probability of any given sequence. However, when trying to find the complete sequence leading to a specific word, the count may often be zero, making it impossible to calculate the probability this way. The intuition behind the  $n$ -gram model is that we can approximate the history of a sentence by just the last few words. For example, the bigram model approximates the probability of a word given all the previous words  $P(w_n|h)$  by using only the conditional probability of the preceding word  $P(w_n|w_{n-1})$ .

To calculate the probability of a complete sequence of words  $W = w_1, w_2, \dots, w_n$ , we start by



decomposing it using the chain rule of probabilities:

$$P(w_{1:n}) = \prod_{k=1}^n P(w_k | w_{1:k-1}) \quad (2.6)$$

Applying the  $n$ -gram assumption, we obtain:

$$P(w_{1:n}) = \prod_{k=1}^n P(w_k | w_{k-n:k-1}) \quad (2.7)$$

Using this formula, we can recursively compute the probability of any sentence given its words.

To apply a language model into the CTC decoding process, the following objective is optimized:

$$W^* = \underset{W}{\operatorname{argmax}} P(W | X) \cdot P(W)^\alpha \cdot L(W)^\beta \quad (2.8)$$

Where  $P(W | X)$  is the CTC conditional probability,  $P(W)$  is the language model probability, and  $L(W)$  is a word insertion bonus that rewards longer sequences by assigning a score based on the number of words in  $W$ . Since language model scores are only computed for complete words, and decoding tokens can consist of individual letters or subwords, this biases the search towards shorter hypotheses. The word insertion bonus helps to mitigate this effect by encouraging longer sequences. Both  $\alpha$  and  $\beta$  are parameters that control the weight of the language model and length bonus during the decoding process.

### 2.1.3 Advanced Architectures

Some advanced speech recognition architectures go beyond the scope of this work. The first one is the combination of CTC and encoder-decoder into the same model, where two prediction outputs are used and the two loss functions are weighted. Inference using this model involves a complex decoding step combining the two outputs.

Another architecture tries to improve CTC to be used in streaming applications, where low latency is necessary and the output need to be produced before the full input is available. It is called a Transducer, with specific variations such as the RNN-Transducer and the Transformer-Transducer. It has two components: the CTC acoustic model and an additional language model called the predictor that conditions the output token history.

## 2.2 Text Processing

In speech recognition models, the labels utilized consist of textual data; however, deep learning models need numerical features by computing the loss function in the training process. To convert this textual data in numerical representations, it is essential to establish a text processing pipeline constitute by three steps: text normalization, tokenization, and numericalization.

During inference, this process is reversed, with the first step being the CTC decoding to remove repeated elements and the blank token. Then, a vocabulary is used to convert from numerical predictions to tokens. Those tokens are joined into words and sentences, and special tokens that indicate the beginning/end of sentences or space between words are processed accordingly.

The components of the text processing pipeline are described as follows:

- **Text Normalization.** Ensuring a close match between transcriptions and audio content is crucial when training speech recognition models. Text normalization is a technique that reduces unreliable representations from the transcriptions by expanding compressed information into its verbalized form. As an illustration, monetary values can be extended to verbal forms, such as converting “\$100.50” to “One hundred dollars and fifty cents.”
- **Tokenization.** After processing transcriptions, the next step is tokenization, that is, to take the normalized text and divide each sentence into smaller components named tokens. Avoiding an excessively high number of individual tokens in the training dataset is crucial, as this could lead to each token appearing only a few times. In this context, the most common form of tokenization is to split a text into a sequence of individual characters, simplifying the target vocabulary from potentially thousands of individual words to a basic set of characters. One problem with character tokenization is that the speech recognition model will need to produce the transcription of a character by timestep, meaning that it also needs to learn an implicit language model that knows how to spell words. An alternative way to split sentences into tokens is based on subword units. Here, tokens can be bigger than a single character but smaller than a complete word. The Byte-Pair-Encoding (BPE) algorithm can be utilized to perform subword tokenization, allowing an open vocabulary to be represented by a fixed-size vocabulary of variable-length tokens (GAGE, 1994).
- **Numericalization.** Since the sentence is divided into a list of predefined tokens, the next step is numericalization. It takes the vocabulary of tokens created in the last step and creates a mapping that uniquely identifies each token by a number. Then, it iterates over the tokenized input and replaces each token with the appropriate numerical representation, creating a tensor that is used by the training script.

Finally, inverse text normalization is an optional step to improve the transcription legibility that tries to reverse the normalization step done before training and compressing verbalized expressions.

## 2.3 End-to-End Speech Recognition Models

With the advent of deep learning, end-to-end speech recognition models have achieved state-of-the-art results on many speech recognition tasks. This work investigates models that use convolutional and transformer layers, exploring different trade-offs between model size and accuracy. Here, we will explain two models used in this work. First, the QuartzNet architecture is purely convolutional, meant to be computationally efficient and compact. Then, the wav2vec model employs the transformer architecture, with a bigger model and more complex training procedure spanning multiple steps.

### 2.3.1 QuartzNet

NVIDIA developed QuartzNet to be compact and computationally efficient compared to state-of-the-art systems. The model is composed of multiple blocks with residual connections between them. Each block consists of one or more modules with one-dimensional convolutions, batch normalization, and the ReLU activation function. Inspired by MobileNets (HOWARD et al., 2017) and Xception (CHOLLET, 2017), the main novelty of QuartzNet is the use of time-channel separable convolutions. This kind of convolution breaks the computation into two steps that completely decouple the time and channel-wise parts of the processing.

A depthwise convolutional layer replaces each conventional convolution layer with kernel length  $K$ , that operates on each channel individually but across  $K$  frames, and a pointwise convolutional layer, that works on each frame independently but across all channels.

The QuartzNet architecture is presented in Figure 3. The input of the model consists of Mel scale filterbanks. It is processed by an initial convolutional layer  $C_1$ , with a stride of 2. This stride value has the objective of subsampling the features. Then, a sequence of blocks follows, where each block  $B_i$  is repeated  $S_i$  times and has residual connections between the blocks. Each block  $B_i$  consists of the same base modules repeated  $R_i$  times and contains three layers: the 1D time-channel separable convolution, batch normalization, and the ReLU activation. Finally, three convolutional layers  $C_2, C_3, C_4$  complete the model.

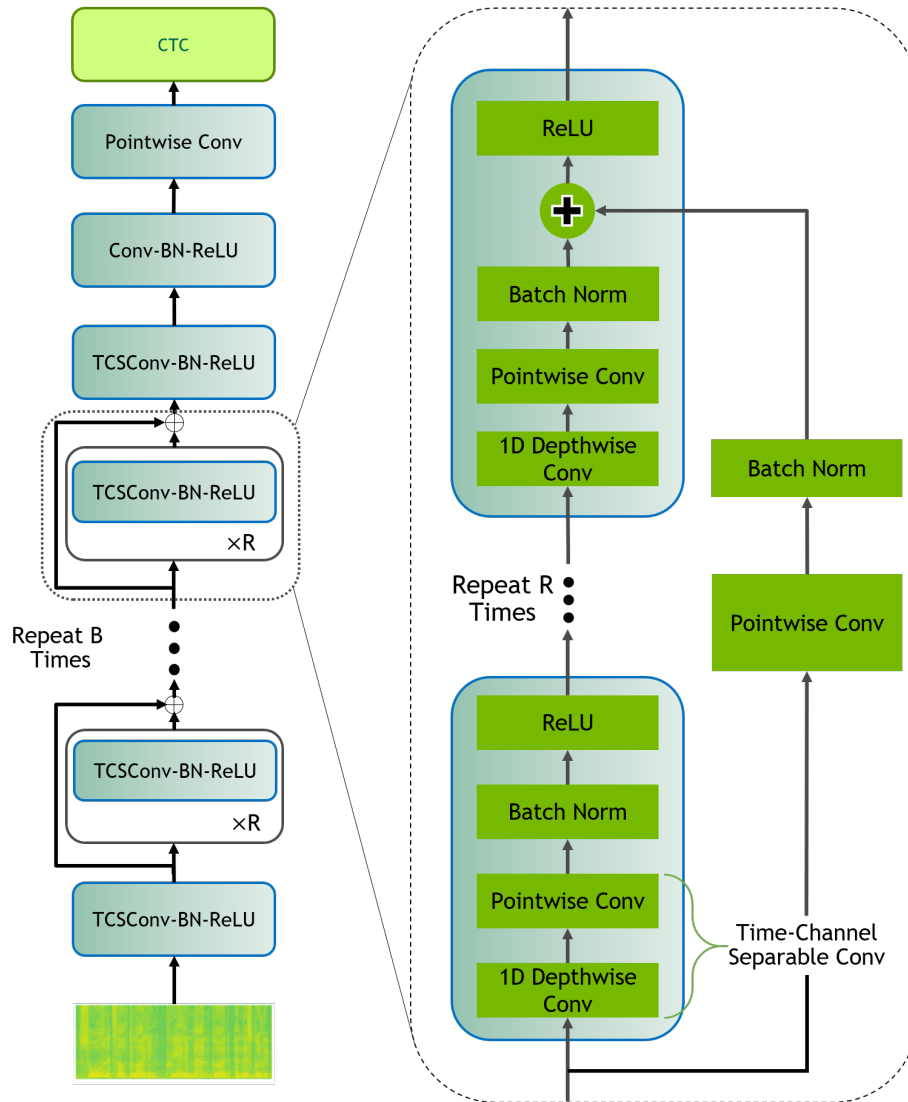


Figure 3 – QuartzNet BxR architecture.

Quartznet has two variants, with a naming convention following the pattern  $QuartzNet - R_i \times S_i$ . The first is  $QuartzNet - 5 \times 5$ , with 6.7 million parameters; the second is  $QuartzNet - 15 \times 5$ , with 18.9 million parameters. Five unique blocks are used by both models,  $B_1 - B_5$ .  $QuartzNet - 5 \times 5$  repeats each block only once, while the  $QuartzNet - 15 \times 5$  has three repetitions of each block.

Since this model is compact and computationally efficient while being trained following the CTC criterion, it will serve as the baseline for comparison.

### 2.3.2 Wav2vec

The wav2vec family of models seeks to learn primary speech units exclusively from raw audio and transfer these representations to a speech recognition system with limited labeled data.

This objective is achieved through a two-stage training process. In the first stage, the model is trained using a self-supervised approach with only raw speech samples. Here, the objective is to learn relevant speech representations by having the model predict masked parts of the speech using the rest as context. In the second stage, the learned representations are mapped to a target vocabulary through a small linear layer added on top. This layer is fine-tuned using labeled data to understand better the relationship between the audio representations and their corresponding transcriptions.

The wav2vec 2.0 model (BAEVSKI et al., 2020) pushes the boundaries of speech recognition systems by leveraging a large amount of unlabeled training data to enhance performance across various languages, dialects, and domains. Similar to the Bidirectional Encoder Representations from Transformers (BERT) architecture (DEVLIN et al., 2018) used in natural language processing, the model is trained by predicting masked tokens from the input using the surrounding tokens as context. However, unlike BERT, speech audio is a continuous signal, encompassing multiple aspects of the recording without a precise segmentation between words or other units. To address this challenge, wav2vec 2.0 learns basic units that are 25ms long, allowing the model to develop contextualized representations. These units are subsequently used to describe a diverse range of speech audio recordings and enhance the model’s robustness.

The self-supervised task employed in this model learns a set of fundamental speech units, each shorter than a phoneme, to describe the spoken audio sequence. As the set of representations used is finite, it cannot represent all possible variations such as background noise, and the units guide the model to focus on the most crucial aspects of the speech, enabling it to capture the underlying content more accurately. The model begins by analyzing the raw speech audio using a multi-layer convolutional neural network, breaking down the input speech into segments of 25ms each and deriving a continuous representation from each segment. Then, these segments are processed through a quantizer and a transformer. The quantizer selects the appropriate speech units from an inventory of learned units based on the input audio representation, with approximately half of the audio representations being masked before entering the transformer. The transformer leverages information from the entire audio sequence, followed by a contrastive task that challenges the model to accurately identify the correct quantized speech units for the masked positions.

### **2.3.3 Model Adapters**

The typical process of fine-tuning transformer-based models involves loading pretrained checkpoints trained on vast amounts of speech data. This pretraining step is crucial for achieving good performance in downstream tasks with limited amounts of labeled data, as it provides a strong foundation for further training. However, the large number of trainable parameters in these

pretrained models can pose significant challenges during fine-tuning. The simultaneous optimization of billions of parameters can lead to a massive demand for memory during training and storage space for inference. To put this into perspective, consider that modern transformer-based models can have upwards of 300 million parameters (BAEVSKI et al., 2020), with some of the largest speech recognition models even reaching 2 billion parameters. Researchers have explored various techniques to overcome these challenges, such as gradient checkpointing (CHEN et al., 2016), mixed-precision training (MICIKEVICIUS et al., 2018), and knowledge distillation (HINTON; VINYALS; DEAN, 2015). These methods aim to reduce the memory requirements during training while maintaining the model’s performance. While these techniques can introduce additional computational overhead or require careful tuning for optimal results, they offer promising solutions for scaling transformer-based models to larger datasets and more complex tasks.

Adapters are a promising technique for improving the efficiency of transformer-based models during fine-tuning. The basic idea behind adapters is to insert small bottleneck layers within the individual blocks of the transformer model, effectively creating a lightweight alternative to full fine-tuning. The original model weights are frozen during training, while only the randomly initialized weights of the adapter layers are trained. These adapters allow for efficient parameter sharing across tasks, enabling the model to leverage knowledge learned from one task to improve its performance on another. By training only a fraction of the total parameters, models modified by adapters can achieve competitive performance with much fewer parameters, significantly reducing memory requirements and computational overhead associated with full fine-tuning.

Microsoft Researchers have proposed a technique called LoRA (Low-Rank Adaptation) to adapt large models to specific tasks and datasets (HU et al., 2021). Based on previous studies that have shown that over-parameterized large models often occupy a low intrinsic dimension, the core idea behind LoRA is that the weight changes during model fine-tuning also belong to a lower-dimensional space. In more detail, if  $W$  represents the weights of a single layer and  $\delta W$  represents the alterations in weights during fine-tuning, the authors propose that  $\delta W$  is a compact matrix with a low intrinsic rank. The technique then modifies the update matrix  $\delta W$  by decomposing its rank into two lower-rank matrices,  $A$  and  $B$ . This transformation replaces the original forward pass of the layer,  $Wx$ , with  $Wx + BAx$ . In order to improve the initial convergence of the training,  $A$  is initialized with a random Gaussian distribution, while  $B$  is set to zero. During training, the update  $BA$  is scaled by a factor  $\alpha/r$ . These two values are hyperparameters that control the LoRa layers added to the model.  $r$  is the rank of the update matrices  $A$  and  $B$ , with lower rank resulting in smaller update matrices with fewer trainable parameters.  $\alpha$  is a scaling factor that adjusts the magnitude of the combined result between the original model weights and the new LoRa ones.

## 2.4 Robust Speech Recognition

The existence of noise in speech signals is a natural phenomenon that can significantly impact the accuracy of transmitted information, making it challenging to understand the information contained within the speech signal. As a result, noise suppression techniques are crucial for the effective computational processing of speech signals. Illustrating this point involves examining the various noise categories reported in Benesty et al. (2008): (i) Additive noise originates from ambient sound sources and can be present in all communication channels, making it difficult to separate the speaker's voice from background noise. (ii) Interference occurs when speech signals generated by multiple sources are mixed together, resulting in a cluttered and distorted signal that can impede understanding. (iii) Reverberation is caused by sounds reflected by surfaces within an enclosed space, resulting in echoes and delays that make it difficult to determine the source of the sound. (iv) Echo is due to nearby microphones and speakers, causing a delayed reproduction of the speaker's voice that can obscure the intended message.

Noise suppression is crucial for maintaining the integrity of speech signals ensuring accurate message delivery. Effective techniques can be developed by understanding the different types of noise that impact speech signals. These techniques improve the quality of speech signals, making them more intelligible and easier to process computationally, with significant implications for applications such as voice recognition systems, audio transcription software, or simple conversations in noisy environments.

An effective way to address the noise in speech recognition applications is to train variance-resistant models directly using data from the target domain. This approach involves leveraging the target domain's data to build a more robust model that can handle the unique characteristics of the target population. By training the model on diverse and representative data from the target domain, the model becomes better equipped to handle the variations in speech patterns, accents, and noise levels typical of real-world environments. In situations where real-world data is challenging to obtain or annotate accurately, using clean datasets in combination with simulating acoustic characteristics of the target domain can help increase the amount of available data.

Several techniques are employed to enhance the robustness of speech recognition models, as discussed in Narayanan et al. (2018). Training with a large volume of data from different domains helps improve the model's generalizability, reducing the need for extensive training in new domains. Adverse conditions are also simulated by introducing various types of noise, altering bandwidths, and using codecs for lossy audio compression. In (BALAM et al., 2020), knowledge transfer occurred using a model previously trained on clean data only. Extensive

simulation of adverse conditions was performed, including adding impulse response to mimic the acoustic properties of various environments. (LUO et al., 2021) and (POL'AK; BOJAR, 2021) showed the effectiveness of using pre-trained models as a basis for more robust models with variations in accent, language, and application domain. Moreover, pre-training on unlabeled speech data from the target domain has been proposed to improve model robustness. Robust-wav2vec (HSU et al., 2021) and wav2vec-switch (WANG et al., 2021b) are two approaches that use the unsupervised training step of the wav2vec family of models to leverage unlabeled speech data in the target domain. While robust-wav2vec uses only noisy data for this training step, wav2vec-switch requires clean-noisy pairs. These techniques improve the model's ability to handle diverse and representative data from the target domain.

Another approach is pre-processing the audio before feeding it into the speech recognition model. This step involves using noise reduction, echo cancellation, and spectral subtraction techniques to extract only the relevant information from the noisy audio and make it closer to the clean data used for training the model. By removing unwanted noise and interference, the pre-processing step helps improve the quality of the input data, which in turn enhances the accuracy of the speech recognition model.

According to Benesty et al. (2008), algorithms that perform separation between speech and noise signals focus on three areas: *(i)* filtering, where it is assumed that speech and noise have very distinct spectral characteristics and are stationary in the window of interest so that it is possible to create specific filters to separate the signals. By applying these filters, unwanted noise can be removed, leaving only the speech signal; *(ii)* spectral restoration, where the problem is treated as an estimation of the clean spectrum from the noisy spectrum. This area involves using techniques such as spectral subtraction or wavelet denoising to remove noise from the signal and reveal the underlying speech pattern; *(iii)* model-based, which seeks to mathematically model the speech signal, transforming the problem into a parameter estimation exercise. By fitting a statistical model to the data, these methods can separate speech from noise based on the patterns and characteristics of the speech signal itself.

The choice of technique for speech separation depends on the specific application and the nature of the noise. When applying these techniques, evaluating the trade-offs between noise reduction and speech distortion is essential. Combining these approaches enables the development of robust speech recognition models capable of handling various noisy environments. However, there is a problem when using classical noise reduction algorithms as a pre-processing step for speech recognition systems. These algorithms aim to achieve the best noise reduction for human perception, but they introduce distortions in the audio that negatively impact speech recognition performance (IWAMOTO et al., 2022). To address this issue, implementing methods to reduce these distortions or developing models that directly aim to improve the overall performance of



the speech recognition system is crucial, as discussed in (SEGBROECK; NARAYANAN, 2013; YOSHIOKA; GALES, 2015; KINOSHITA et al., 2020; O'MALLEY et al., 2021).

Both approaches aim to improve the robustness of the speech recognition model by better handling the challenges posed by domain switching. These techniques, whether utilizing data from the target domain or pre-processing the audio to reduce noise, ensure that the model can generalize well to new and unseen environments, improving accuracy and reliability in real-world applications.

### 3 PROPOSAL

In this chapter, we present the main proposal of our dissertation, which aims to develop a robust speech recognition system based on deep neural networks for use in radio communications. The system follows the modern architecture for speech recognition models, with an end-to-end acoustic model and an external language model to boost predictions during inference time. While the language model was easily trained using standard components and proven methodologies, the acoustic model development was the main focus of this work.

To tackle the challenge of limited data availability in the target domain, we employ a combination of labeled data from publicly available datasets and an innovative data augmentation technique to bootstrap the initial acoustic models. This approach enables us to use more training data, improving the model's performance and robustness.

In order to train the initial model efficiently on a large corpus of augmented data, different training recipes will be compared based on the multiple implementations of speech recognition models from various libraries, and their performance will be evaluated. In this step, parameter efficient training procedures emerge in order to provide more efficient training with limited computational resources, in the form of LoRA adapters.

Next, we describe a pseudo-labeling methodology that leverages the initially trained model to label data in the target domain at a lower cost. This methodology uses the initially trained model to label data in the target domain, creating pseudo-labels, which are then filtered based on rules such as high confidence predictions.

Finally, we describe the deployment system projected to provide accurate and reliable speech recognition capabilities in real-world radio communication scenarios.

The proposed methodology is illustrated in Figure 4, where we depict the data flow from labeled datasets to pseudo-labels and ultimately to trained models on the target domain. Data augmentation is performed using the publicly available datasets in order to create the first dataset that will be used to train the initial acoustic model. This initial acoustic model will act as the teacher to the final model, and produce pseudo-labels on the unlabeled section of the proprietary data. We merge the pseudo-labels with a small portion of hand-labeled data to create the final dataset and train the acoustic model in the target domain to the desired performance.

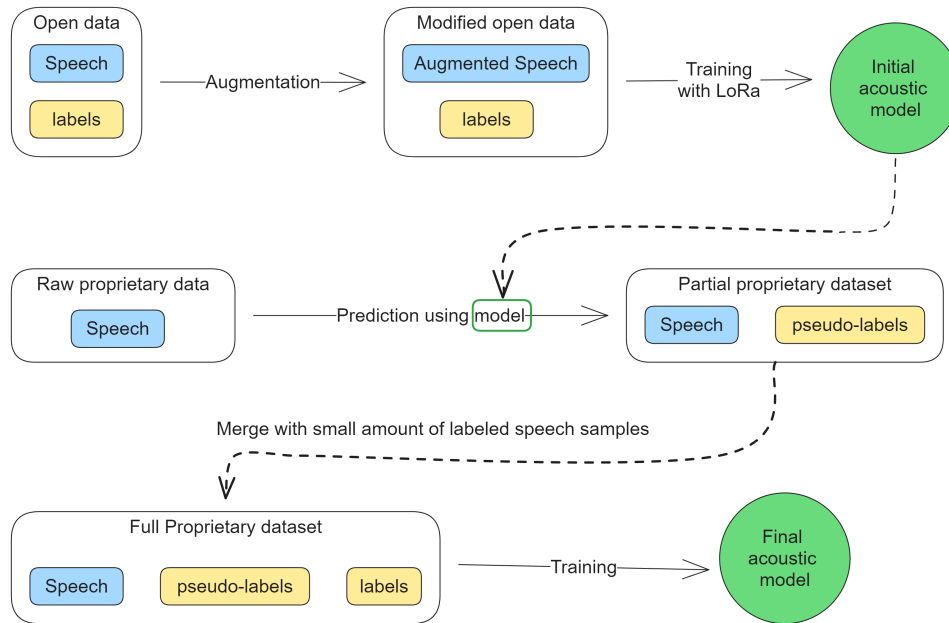


Figure 4 – Overview of the proposed methodology for training the acoustic model.

### 3.1 Simulation of the Acoustic Characteristics of Radio Communication

Overcoming the limited availability of labeled data is a crucial step towards developing robust speech recognition models. One approach to address this challenge is to augment publicly available datasets using custom, hand-crafted augmentation methods that simulate the operating characteristics observed in radio communication. These methods can help increase the diversity of the training data and improve the model’s ability to generalize to unseen speakers and environments. By augmenting the dataset in this manner, we can better train our models to recognize speech in real-world scenarios, where the acoustic conditions may vary significantly from one recording to another.

One way to perform this data augmentation is through a series of signal processing operations, as described in Hirsch e Finster (2005), including: *(i)* utilizing impulse responses to simulate reverberation caused by devices indoors with the microphone positioned at a certain distance from the mouth; *(ii)* adding ambient additive noise to mimic the background noise present in real-world environments; *(iii)* applying a filter representing the frequency characteristics of radio transmissions, as described by ITU-T in its recommendation G.712; *(iv)* transforming the signal using a codec typically used in voice communication systems, such as the GSM codec, to simulate lossy data compression; *(v)* simulating packet loss and bit inversion through various methods to account for transmission over non-ideal channels.

When applied to radio communication in the Portuguese language, we have as reference the work of Duarte e Colcher (2021). This study aims to develop a dataset of noisy audio recordings,

specifically focusing on degenerated audio signals caused by interference in radio transmissions. The case study of interest is voice transmission over a military analog radio transmission channel in Brazil, where one of the most commonly used frequency bands is HF due to its wide coverage in areas such as the Amazon region. To simulate the communication channel, four scenarios were created using different approaches: a simple Additive White Gaussian Noise model, direct noise collection from live antennas, simulation using the PathSim HF radio propagation software that simulates the multipath propagation of the speech signals using three paths with configurable delay and frequency spread, and finally hardware-based simulation of HF and V/UHF channels using hardware built for evaluating modems and waveforms compliant with military standards. The evaluation of the datasets created using such scenarios pointed that their use during training can improve the speech recognition performance, mainly when applied in a noisy environment.

In this work, we leverage the power of GNU Radio (BLOSSOM, 2004), a versatile code library that provides blocks implementing various modulations and channel models. Although its primary purpose is to develop software-defined radio devices, it can also be used for simulating complex telecommunication systems by enabling the modeling of voice signal transmission under challenging conditions. By utilizing this library, we can simulate the operating conditions with greater accuracy and reliability.

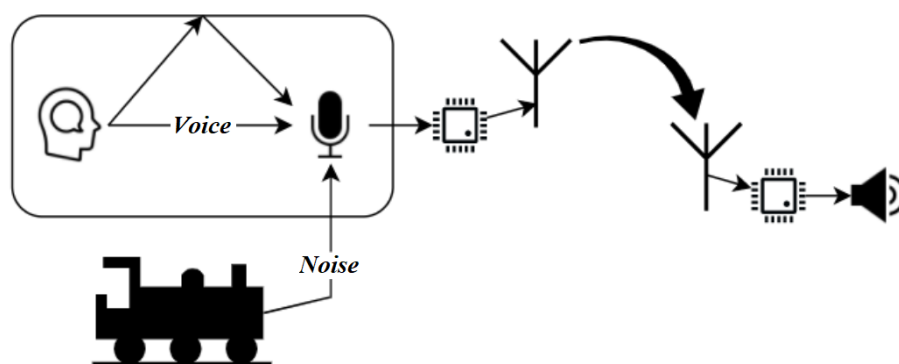


Figure 5 – Simplified model of speech transmission over a radio communication system in the presence of additive noise.

In the context of speech recognition technology, a voice communication system involves four distinct stages: audio collection, encoding, transmission, and decoding, as depicted in Figure 5. During the initial stage of audio collection, undesirable environmental noises, such as additive noise and reverberation, arise due to the location where the audio is recorded. This contaminated audio is subsequently encoded using an audio codec that condenses the information in order to extend the transmission distance. It should be noted that each of these stages introduces distinct side effects that impact the final quality of the transmitted audio. Furthermore, due to their non-linear nature, the order in which these steps occur also makes a significant difference. In this study, we implemented the narrowband FM transmitter and receiver. These transceivers are

commonly used in standard communication radios because they transmit high-quality audio signals over long distances.

The transmitter's input consists of audio files saved in WAV format with a sampling rate of 16 kHz, which is the standard rate for speech signals. We apply a bandpass filter as the first block to process the input signal and prepare it for transmission. This filter limits the frequencies present in the input signal to the range of 300 Hz to 3400 Hz, which is the common frequency range used for pure speech signals. The filter has a unitary gain and a transition band of 200 Hz, ensuring that the signal remains within the desired frequency range.

Next, we employ the Narrow Band FM Transmitter block to transform the audio samples into a complex frequency-modulated signal representation. This process involves modulating the carrier frequency with the input signal, resulting in a transmitted signal with a frequency that is a multiple of the sampling rate. As it is a narrowband type transmission, this block features an internal low-pass filter of the input with a cut-off frequency of 4.5 kHz. This filter removes any high-frequency signal components beyond the desired range, ensuring that only the relevant information is transmitted. The carrier frequency used in modulation was chosen based on the criterion of four times the value of the signal's sampling rate, resulting in a carrier at 64 kHz. This frequency is appropriate for speech signals, as it allows for high-quality transmission with minimal distortion. Finally, a three-times interpolation is applied to increase the transmitted signal's frequency to 192 kHz. This final operation amplifies the signal and ensures that it remains within the desired range for reception.

In the receiver section of the FM narrowband radio system, the signal first passes through a bandpass filter centered at 192 kHz with a 6 kHz window. This filter removes unwanted signals outside of this frequency range and decimates the signal three times, resulting in an intermediate output signal with a sampling rate of 64 kHz. The next step is to use the NBFM (Narrowband Frequency Modulation) Receive module, which transforms the complex modulated signal back to its real representation. We can obtain the final waveform after processing by recovering the frequency-modulated signal.

A simple channel model is used in this system to simulate the transmission process, which takes into account two essential factors: additive white Gaussian noise and frequency shift. The former represents random fluctuations in the signal strength, while the latter refers to the potential changes in frequency caused by various factors during transmission. While this simplified channel model provides a basic understanding of the transmission process, more complex models can be used in GNU Radio to simulate additional effects such as multivariate propagation, Doppler shift, and time-varying distortions.

Additionally to the narrowband FM transmission system implemented, the audio is also transformed using the GSM codec to emulate the lossy data compression in the target radio communication domain.

## 3.2 Training Recipes

Speech recognition models are trained using a set of optimization algorithms and hyperparameters, which collectively make up the training recipe. The optimizer algorithm updates the model's weights during training, and its main hyperparameters determine how these changes are executed. Parameter groups divide segments of the model into smaller sub-sections that use specific optimizer hyperparameters to improve the performance of the entire model. Learning rate schedulers adjust the maximum learning rate used by the optimizer during training, implementing warm-up and decay phases to stabilize the training process better. Regularization techniques like L2 regularization, dropout, and early stopping aim to prevent overfitting by adding a penalty term to the loss function or randomly dropping some neurons during training. These techniques help improve generalization performance on unseen data. Overall, these diverse hyperparameters and algorithms work together to improve the performance of speech recognition models during training.

In addition to the key hyperparameters and algorithms used in speech recognition training recipes, it is crucial to consider the specific code libraries or frameworks being utilized. There are many different ways to implement these hyperparameters and algorithms, depending on the specific code library or framework used, leading to variations in the overall performance of the models. Therefore, it is essential to compare multiple speech recognition training recipes from various libraries to determine which approach yields the best results.

In this study, the recipes from huggingface, fairseq and NeMo will be compared. A common point in all libraries is the use of the Adam optimizer (KINGMA; BA, 2015) or a variant (AdamW (LOSHCHILOV; HUTTER, 2018), NovoGrad (GINSBURG et al., 2019)), alongside the utilization of learning rate scheduling. Since the learning rate is the main hyperparameter that controls the training process, hand-crafted schedulers try to make the optimization process smooth and fast by increasing and decreasing the value at critical points. The warmup is a common occurrence, trying to minimize the instability of the first training steps while the optimizer state is starting to accumulate statistics.

First of all, the huggingface library is reference in the implementation of transformer-based model architectures across multiple tasks. Their recommendation is to train the whole model since the beginning of training and use a long warmup period along with gradient clipping to

increase stability. Next, fairseq contains the original implementation of the wav2vec architecture used in this study. It applies a tri-state scheduler, and also initially freezes the encoder section of the model for a predefined number of steps before unlocking it and optimizing the whole model. Finally, NeMo library incorporates QuartzNet, the fully convolutional model to be contrasted against the transformer model. It uses the NovoGrad optimizer with weight decay and beta values that vary from  $(\beta_1 = 0.95, \beta_2 = 0.5)$  to  $(\beta_1 = 0.8, \beta_2 = 0.25)$ . When finetuning with a large volume of data, it is recommended to train the whole model; otherwise, only the decoder and batch normalization layers should be unfrozen.

Experiments to compare the performance of these approaches will be carried out in Chapter 4, and their outcomes will be synthesized into an optimized training recipe by taking the best aspects from each library.

### 3.3 Pseudo-labeling

Large-scale audio transcription can be challenging due to its cost and time requirements, which has led to the development of semi-supervised training techniques. These methods take advantage of the abundant amount of available unlabeled audio to improve speech recognition models' performance. One such approach is pseudo-labeling, which generates noisy labels from an initial model trained on a smaller labeled dataset (LEE, 2013; KAHN; LEE; HANNUN, 2020).

Pseudo-labeling involves using the initial model to generate labels for an unlabeled dataset. These pseudo-labels (PLs) are then used to train a new model. There are several variants of pseudo-labeling, including using the same model to generate and then training or using PLs generated by a big and slow teacher model to train a smaller student model (HIGUCHI et al., 2022).

The effectiveness of pseudo-labeling can be further improved when more raw data is available (LUGOSCH et al., 2022). By leveraging these techniques, it is possible to improve the performance of speech recognition models without requiring a large amount of manually transcribed data.

The quality of the pseudo-labels depends on several factors, including the amount and quality of training data available. To generate high-quality PLs, it may be necessary to use techniques such as including a language model during the inference step or using a teacher model to provide more accurate PLs.

This work implements a single round of pseudo-labeling using language models during inference.

We compare the outcomes of utilizing the same model versus distilling knowledge from the pseudo-labels into a smaller model when training. Confidence scores from the language model based on each possible alignment of the hypothesis are extracted to filter predictions. For each pseudo-label, the mean confidence score is calculated by normalizing the language model's output with the number of words in the prediction. We discovered that this metric has a strong correlation with input and label quality. When the input has minimal noise and the acoustic model can accurately predict a transcription, the language model scores tend to be high. However, the presence of noise, interference, or fast speech reduces the clarity of the acoustic model's predictions, resulting in low-scoring pseudo-labels.

### **3.4 Production Deployment**

The speech recognition model presented in this work is just one component of a larger research project focused on developing software tools for auditing radio communications between train drivers and traffic controllers. The objective of this project is to ensure that these critical personnel are following proper communication protocols in order to improve the safety and efficiency of railway operations. In order to achieve these goals, our system integrates deep learning models and manages their lifecycle throughout the training and deployment process.

The speech recognition model itself is a critical component of this system, as it provides the ability to transcribe and understand spoken language in real time. In this section, we will provide a detailed description of the system architecture and the deep learning models used in the system. We will also discuss the lifecycle management strategies employed to ensure that the models remain accurate and up-to-date over time. By presenting these details, we hope to provide a comprehensive overview of the speech recognition component of our larger research project and its potential applications in the field of radio communications auditing.

#### **3.4.1 Voice Communication Analysis System**

The voice communication analysis (ACV) system being developed is a complex software tool designed to analyze voice communication data and provide insights into various aspects of speech patterns, acoustics, and content. It is composed of multiple components that are broadly divided into three categories: *(i)* frontend components build the interface that final users directly use; *(ii)* backend that holds all of the metadata and predictions into a structured format that can be efficiently queried and filtered; *(iii)* a set of components responsible for the data and model lifecycle. A broad overview of the relationship between the different components can be seen in



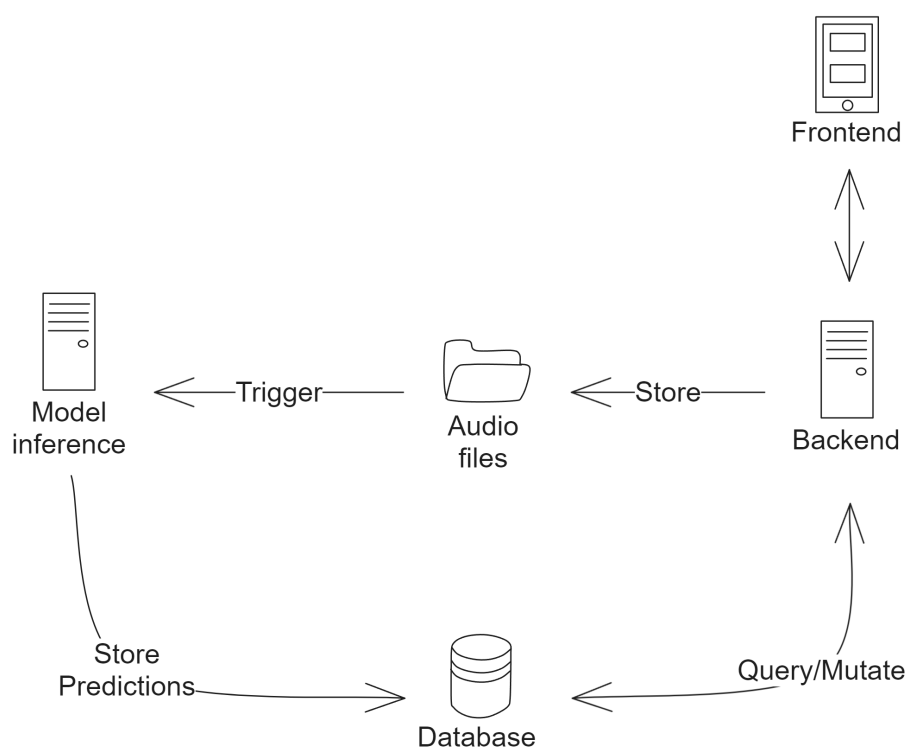


Figure 6 – Overview of the voice communication analysis system.

Figure 6.

The division of the ACV system into frontend, backend, and data/model lifecycle components was developed in response to the unique requirements of deep learning models in production. Unlike traditional backend systems, deep learning models require more robust machines with high-performance graphics processing units (GPUs) to speed up inference. This need for GPUs is because the computational complexity of deep learning algorithms can be substantial. GPUs provide a significant performance boost by offloading the workload from the central processing unit (CPU).

Additionally, predictions made by deep learning models are more efficient when multiple inputs can be batched for processing simultaneously. This type of implementation is because batching allows the model to process multiple examples simultaneously, significantly reducing computational resources and improving overall system performance.

Furthermore, the code, model weights, and training data of a deep learning system define a dynamic and sensitive machine learning system that is prone to distribution drifts<sup>1</sup>. This can lead to performance degradation; therefore, it is vital to have systems in place for monitoring

<sup>1</sup> Distribution drift occurs when changes in the input data or other environmental factors cause variations in the system's output

performance and addressing any issues that may arise due to distribution drifts.

The audio files that the ACV system will process consist of multiple recordings, each lasting between ten seconds and one minute in duration. These recordings are created during various communication activities, which can span multiple audio files and involve different numbers of speakers with varying quality conditions. In order to generate information that is useful for auditing purposes, a variety of models must be employed to make predictions over each recording.

To begin, a voice activity detector (VAD) is used to segment the original audio into continuous regions where speech occurs. Within each region, we apply multiple models to execute various tasks such as speaker identification, speech recognition, and audio quality extraction. Speaker identification is the process of determining who is speaking in an audio recording. Audio quality extraction refers to the process of assessing the overall fidelity and clarity of an audio recording, which can help to identify issues such as background noise or distortion. In addition to these speech-related tasks, we also classify the audio recordings based on their corresponding work areas in order to provide added context for analysis purposes. This information can be useful for identifying trends related to specific job environments or workflows.

By using multiple models in this way, the ACV system can generate highly accurate and valuable information that can aid in the auditing process, allowing users to identify patterns and trends in communication activities and make data-driven decisions about how to improve communication protocols and practices.

### **3.4.2 Frontend**

The frontend of the ACV system is a React application that leverages the Apollo client for communication with the backend service through a GraphQL API. Chrakra UI serves as both the component system and design library for the frontend, providing a consistent look and feel throughout the developed interfaces.

The frontend follows the single-page application paradigm, including a project management section, an advanced structured labeler, and summary reports. In the project management section, users can create new projects by uploading audio files to be processed, as well as to edit or delete existing projects. The structured labeler is the most developed part of the frontend, allowing users to view and correct predictions generated by various models.

Each project contains a list of related audio recordings organized based on their metadata. When

as a user clicks on an audio recording, a new interface opens with the structured predictions as shown in Figure 7. An audio player is included, displaying the waveform overlaid with individual speech segments. Each segment is color-coded according to its group classification (traffic controllers or train drivers), making it easier for users to visualize the conversation structure.

Below the audio player, all information extracted by each model is displayed in a compact format that maximizes utility and avoids information overload. Information about who was speaking at each segment, their group classification (CC for traffic controllers and MM for train drivers), start and end time for each segment, confidence extracted from model predictions and accuracy when the audio is labeled by a user. This layout is developed through an iterative process that takes feedback from weekly meetings to ensure that the frontend remains user-friendly and effective for auditing purposes.

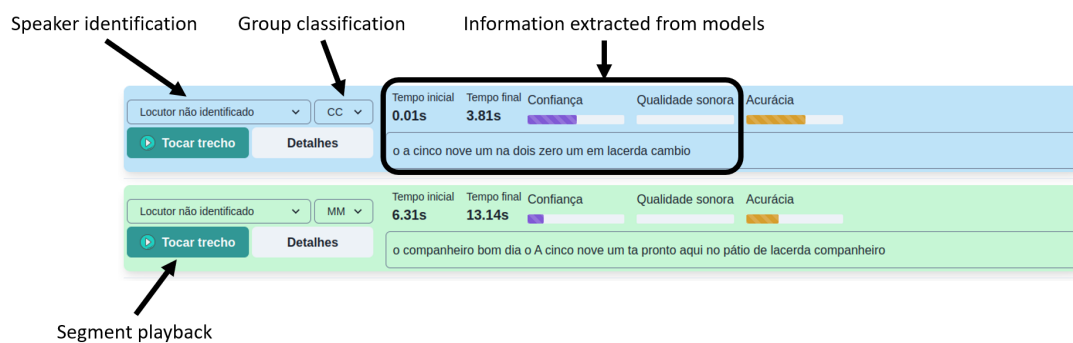


Figure 7 – Speech segments with corresponding information extracted via multiple models

### 3.4.3 Backend

A KeystoneJS server powers the backend of the ACV system in conjunction with a SQL database. Metadata about audios, model predictions, and user labels are stored in a structured format that enables the frontend to query this data for analysis purposes.

Due to the complexity of the relationships between the different prediction types, GraphQL is utilized as the interface between the frontend and backend to facilitate efficient data retrieval. Facebook developed this query language to provide a flexible and powerful means of querying and modifying data using a declarative language. It follows a strongly typed interface based on the backend schema, enabling users to perform two types of operations: queries and mutations.

Queries request specific data from the backend, allowing users to extract information with precision and flexibility. They can specify filters and complex relationships using a single query, making it easy to retrieve the desired information without having to execute multiple queries. On the other hand, the mutation operation enables users to make changes to the data, including

creation, deletion, or updates as needed. This powerful combination of query and mutation operations provides a robust interface between the frontend and backend components of the ACV system, enabling efficient data management and analysis.

### **3.4.4 Machine Learning Operations**

Machine Learning Operations (MLOps) is a vital component of modern software development that aims to streamline the process of deploying, managing, and maintaining machine learning models in production environments. It combines the principles of both machine learning and DevOps to create an efficient and scalable system for managing complex data-driven projects.

The core components of an MLOps system encompass several critical aspects of machine learning development, including data management, infrastructure and tooling, testing, deployment, and monitoring. Data management involves the efficient storage, processing, and retrieving of large datasets necessary for training and validating models. Infrastructure and tooling include the selection and configuration of hardware and software resources required to run machine learning workloads at scale. Testing plays a crucial role in ensuring the accuracy and reliability of machine learning models, as well as detecting and fixing issues before deployment. Deployment involves the seamless integration of trained models into production environments, where they can be used to generate insights and drive decision-making. Finally, monitoring is essential for maintaining model performance and identifying potential issues that may arise in real-world scenarios.

Overall, MLOps provides a comprehensive framework for managing machine learning projects from development to deployment, ensuring efficient and reliable model management in production environments. By integrating DevOps practices with the unique requirements of machine learning, MLOps enables organizations to realize the full potential of their data-driven projects while minimizing risk and maximizing performance.

Machine learning powered systems follow a continuous cycle after their initial research and development stages. This process includes deploying models into production environments, monitoring their performance, and making updates to ensure they continue to provide accurate insights. However, the ever-evolving nature of data distributions makes it challenging to maintain model accuracy over time.

Several factors can contribute to a decline in model performance, including changes in the input data distribution, the relationship between input and output variables, and even when collected samples do not adequately approximate the target use case. In some cases, these issues can lead to data drift, making it necessary to collect new data relevant to the target use case and retrain the

model. To address these challenges, machine learning systems must be designed with a focus on continual learning and adaptation. This entails constantly updating deployed models with new data and retraining them to ensure they remain accurate and relevant. A comprehensive approach to machine learning requires keeping track of changes in input data distribution and addressing any issues that arise promptly.

The base of the ML cycle is a robust infrastructure and set of tools that enable efficient data processing. This includes programming languages, libraries, and hardware used. Python is the dominant language in data science due to its flexibility, readability, and extensive support for scientific computing and machine learning libraries. Deep learning models require a specific set of optimized operations for large-scale training, including an automatic differentiation engine (autograd), efficient vectorized linear algebra kernels, and accelerator support in GPUs and TPUs. These functions are integrated into deep learning frameworks such as PyTorch and TensorFlow.

PyTorch was selected because of its growing popularity in the data science community and its emphasis on providing low-level abstractions to build deep learning models. High-level frameworks, such as Fast.ai, Pytorch Lightning, Hugging Face Transformers, and Torch Image Models, use PyTorch as their foundation to provide developers with a seamless experience. These tools allow data scientists to focus on building complex models rather than worrying about the underlying infrastructure.

The models developed in this work utilize the thunder-speech framework. This cutting-edge tool integrates the structure and training algorithms of PyTorch Lightning with pretrained transformer models from Hugging Face. This combination provides a streamlined interface for rapid experimentation and seamless production deployment.

To bring models to production, the Ray framework is employed. Ray offers a unified interface for distributed computing, where tasks serve as the basic unit of resource allocation and scheduling from a single machine to a cluster. The framework is responsible for scheduling the computing required. Tasks can be interconnected, facilitating the creation of training and inference pipelines. In particular, the Ray Serve component is used during the inference step, which allows for the specification of a complex model pipeline with independent deployment, update, and scaling of components. This enables multiple deep learning models to be developed and deployed independently without affecting each other's performance or scalability. That way, the multiple deep learning models that are used in inference can be developed independently of each other.

### 3.4.5 Testing Machine Learning Systems

When deploying machine learning systems in production, it is critical to ensure that they are functioning correctly and will continue to do so. Testing plays a vital role in this process, as it can help identify any issues before they become major problems. However, testing alone cannot guarantee that the system will be fully functional.

Several types of software testing can be used to evaluate traditional software. Unit tests focus on individual pieces of code and their functionality. Integration tests evaluate multiple subsystems simultaneously to ensure they work together as expected. Finally, end-to-end tests simulate realistic user behavior while using the entire system to identify any major issues or flaws.

To improve the testing process, best practices recommend automation and a continuous integration system that automatically runs the test suite when new code is pushed. This approach ensures that tests are reliable and run quickly, allowing for a more efficient development cycle. Additionally, following the testing pyramid can help prioritize testing efforts by focusing on unit-level tests as the foundation, integration tests, and end-to-end tests.

Machine learning systems differ from traditional software systems because they comprise code and data. This unique aspect of ML systems requires specialized testing methodologies to ensure their correct functioning. Instead of directly implementing the desired functionality, as in traditional software systems, the code in those systems defines an optimization process that might produce the desired “program” in the form of a model.

When testing ML systems, it is essential to consider the data and the model itself. Overlooking the importance of testing the data can lead to flawed results. Additionally, building a granular understanding of the model’s predictions before deploying it is crucial for ensuring it effectively meets business requirements. It is also essential to measure the relationship between model performance metrics and business metrics to determine if the ML system is achieving its desired goals. However, relying too heavily on automated testing can be risky in ML systems as the data may change or have unexpected outcomes, which may not be captured by automated tests. Therefore, it is crucial to have monitoring and testing in production to ensure that the ML system continues to function correctly as new data becomes available.

Figure 8 represents an ML system in production. The process starts with the training system, which uses code and data to produce an ML model as an artifact of the training process. This model is then wrapped into a prediction system, which performs raw data pre-processing from different sources into the format expected by the model and post-processing of predictions to convert them into a human-understandable format.

The serving system uses this prediction system to deploy the model in an online fashion, taking requests from users and scaling computations as needed. Production data can be collected to identify drifts and perform a new labeling round to improve the next model. The labeling system should take raw data from production, provide it in a format that helps the labeling process, and store it for use by the training system.

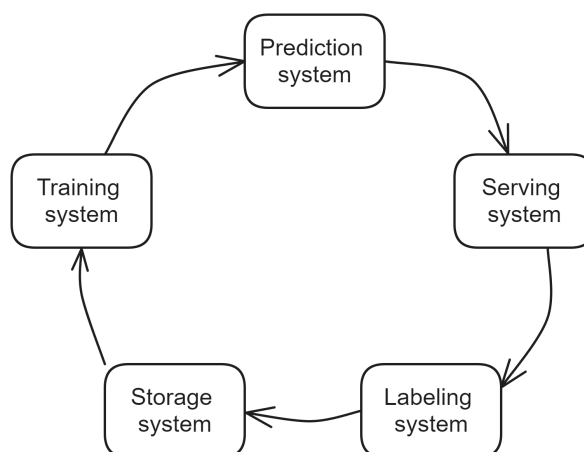


Figure 8 – Machine learning system lifecycle

Infrastructure tests are an essential aspect of ML systems testing, and in this work, we implement infrastructure tests at the training systems level to ensure that the training pipeline is functioning correctly. We use PyTorch Lightning, which abstracts the engineering aspects of the implementation, for our training system. The implemented tests confirm that mixed-precision training happens without any numeric underflow or overflow problems and provide support for distributed training via data and model parallelism. However, we need to be cautious when testing the speech recognition system, as the CTC loss function needs to be computed with full precision to avoid numerical issues. Therefore, we must carefully design and implement the tests for this system to ensure that it meets the required performance standards and functions correctly in production. Overall, infrastructure tests play a critical role in ensuring the reliability and effectiveness of ML systems, and we need to implement them thoroughly to ensure their success.

Next, we implement training tests that integrate the training system with the data storage system. This ensures reproducibility of training runs and maintains consistency in the model's performance. Memorization testing is critical at this stage, as it assesses the model's ability to memorize a small fraction of training data. If the model fails this test, it cannot generalize to broader situations.

We apply functionality tests to the prediction system to avoid regressions in the code. Part of this code is composed by data transformations that have specific characteristics, such as feature normalization, that can be tested like usual software projects. Another critical test is loading a trained model and running a prediction on a set of key examples, checking that the expected

predictions are happening.

Evaluation tests are designed to ensure that newly trained models are ready for production by testing integration between training and prediction systems. These tests go beyond simple metric calculations on a fixed test set and instead evaluate different data slices and inference conditions. They also compare the new models with previous versions and relevant baselines. In the case of this speech recognition system, the principal slice to be evaluated corresponds to the different work area groups. Most of the time, the speech from traffic controllers is recorded at the location where it happens, while speech from drivers goes through a transmission step that causes spectral distortions and added noise. It is essential to evaluate the model in those difficult situations and find the granular slices of data that correspond to prediction failure. Some of the failure modes discovered in this process were related to cross-talk, noise levels that distort the speech beyond comprehension, fast speech, and phonetic ambiguous pronunciation.

We can use data augmentation during evaluation tests to simulate how the model performs at different noise levels. This involves creating additional training data by applying various transformations to the existing data, such as directly adding noise or distorting the signal according to the communication channel modeled before. By evaluating the model's performance on this augmented data, we can measure its robustness to different noise levels and better understand its ability to handle real-world scenarios.

The accuracy of predictions made by a machine learning model depends directly on the quality and consistency of the data used for training. Labeling tests are conducted to ensure that the procedures used to generate this data produce high-quality labels. During the initial labeling process, a basic version of this test was developed by Ribeiro (2020) using a word bank containing common terms to check for suspicious labels.

Finally, we also perform expectation tests to ensure that the quality of the input data used for training meets specific criteria. These tests help prevent issues such as overfitting or underfitting by ensuring that the training data is diverse and representative. Implemented tests include: *(i)* verifying that the length of training samples falls between 1.5 seconds and 20 seconds; *(ii)* discarding segments with no speech or empty labels; *(iii)* conducting data leak tests to ensure that training and testing data are separate.



## 4 EXPERIMENTS

### 4.1 Speech Datasets

In recent years, the availability of datasets for automatic speech recognition in Brazilian Portuguese has expanded significantly. As recently as 2019, there were only four publicly accessible datasets in this language, comprising approximately 60 hours of recordings sourced from various materials such as audiobooks and lectures. Today, however, over 500 hours of data are freely available, thanks to contributions from diverse resources. Some of the largest datasets include:

- The CETUC dataset (ALENCAR; ALCAIM, 2008) contains 145 hours of audio with 100 speakers, half being males and females each. A set of 1,000 distinct sentences is used, where each speaker speaks all the sentences once. The audio was recorded in a controlled environment, with a sampling rate of 16 kHz.
- The Multilingual LibriSpeech (MLS) (PRATAP et al., 2020) consists of audio extracts from Librivox project books in the public domain for eight languages, with 160 hours of Portuguese and 36 male and 26 female speakers.
- The Multilingual TEDx (SALESKY et al., 2021) the database includes recordings of TEDx speeches in eight languages. The Brazilian Portuguese corresponds to 164 hours of duration.
- The Common Voice project (ARDILA et al., 2020) aims to provide free speech recognition system training data for numerous languages. To achieve this objective, users contribute voice samples through the website or app by reciting predefined statements. Elsewhere in the process, volunteer submissions undergo verification. Version 7.0 of the publicly available dataset, released in July 2021, includes 84 validated hours of Portuguese audio contributed by 2038 speakers. Due to the lack of a controlled environment during the recording process, the audio may feature varying degrees of background noise and sound quality.
- The Corpus of Annotated Audios (CORAA) (JUNIOR et al., 2021) features 290 hours of audio focused primarily on conversational segments rather than scripted passages and narration, like other databases utilize.

Additionally, we have built a proprietary dataset curated using recordings captured during the interactions between train drivers and traffic controllers. This dataset contains approximately 10 hours of labeled data and more than 30 hours of unlabeled data. Due to the unique characteristics

of our radio recording system, these recordings are organized into distinct radio channels, and information is only saved during active communication. Each audio file may consist of one or multiple speakers contributing sequentially, with a duration ranging between 15 seconds and a minute. Initially, labeling was performed for each file, implying that every file had a single sentence corresponding to the transcription of all spoken sequences contained within it, mixing multiple speakers into a singular annotation.

During this work, we undertook a second labeling process on the labeled set of our dataset in order to more accurately reflect real-world production settings. By partitioning each audio into its speech segments and transcribing each section separately, we enhanced the quality of our labels. As a result, model performance also saw an improvement. Since these labeled regions only contain speech content, our model no longer needs to learn to distinguish between speech, noise, and silence sections because this has already been accomplished by the voice activity detection system in place. Additionally, shorter label sequences lead to faster convergence and more stable training due to a decreased alignment requirement between input and output for the model to learn. Finally, smaller label segments also lessen memory usage during training, thereby enabling larger models to be utilized.

The audios were partitioned using a 80/10/10% split for training, validation, and testing purposes. The audio division was carried out according to the temporal and radio channel information collected during the initial construction of this dataset, as explained by Ribeiro (2020). By implementing this strategy, entire audio recordings from a specific radio channel are maintained within the same split, mitigating issues associated with data leakage while producing the dataset. Notably, train drivers frequently communicate with distinct traffic controllers and appear in multiple splits as a result of their operations. The fact that traffic controllers typically operate on the same radio channel throughout their duties was taken into account during the splitting process.

One significant disparity between publicly available datasets and our privately built dataset lies in audio quality. Audio extracts from train driver and traffic controller communications carry a considerable amount of ambient noise, lossy data compression employed to attain long-distance communication at the cost of sound quality, rapid speech with dubious diction, and potential overlapping between different speakers. In order to estimate speech quality and measure the disparity between public and private data groups, we utilized the NISQA model (MITTAG et al., 2021). This model evaluates overall speech quality but also forecasts four speech quality dimensions: Noisiness, Coloration, Discontinuity, and Loudness. By providing more insight into the root causes of degraded audio quality, these dimensions offer a deeper understanding of the issues at hand.

Table 1 offers additional insights into the quality characteristics of our proprietary dataset compared to openly available alternatives. Initially, both datasets exhibit similar noise scores, but notable differences emerge upon closer examination. Due to the frequent acquisition of open data in uncontrolled environments, it often contains variable degrees of ambient noise and interference. In this aspect, both datasets have a comparable frequency of noise presence. As evidenced by the visualized coloration, the proprietary dataset exhibits substantial compression artifacts provenient from heavy audio compression required for achieving extended communication range, thereby significantly degrading spectral information.

Table 1 – Speech quality dimensions as estimated by NISQA. Value between 1.0 and 5.0. Higher is better.

Characteristic	Open Dataset	Proprietary dataset
Overall quality	3.0	1.3
Noisiness	1.4	1.9
Coloration	3.9	1.7
Discontinuity	4.1	2.3
Loudness	3.8	2.8

Furthermore, the discontinuity metric reveals discrepancies attributed to the acquisition methodology of our privately held dataset. The raw recordings suffer fragmentation from numerous short speech fragments with multiple silence pauses between them, negatively impacting continuity and fluency. Finally, the loudness scores indicates that when there’s the presence of noise in our dataset, the Signal-to-noise ratio is worse on our case.

## 4.2 Code Libraries

The author of this work developed the thunder-speech library<sup>1</sup> to facilitate the training and deployment process of speech recognition models. Based on the Pytorch framework (PASZKE et al., 2017), this library aims to simplify the training and deployment process of CTC-based speech recognition models through a comprehensive set of tools. It offers a unified module for training CTC-based speech recognition models, a versatile text processing pipeline capable of handling either character or subword tokenizations, and user-friendly data loading utilities. One of the key highlights of this library is its emphasis on developer productivity and deployment simplicity. With just a few lines of code, one can easily load a pre-trained model and export it to inference mode using TorchScript serialization format, with preprocessing and post-processing steps encapsulated into a single file.

Users can anticipate various benefits from leveraging the thunder-speech library, such as end-to-end speech recognition model implementations, straightforward fine-tuning procedures, inference

<sup>1</sup> Available at <https://github.com/scart97/thunder-speech>

support as a first-class feature, and developer-oriented API design. However, it must be noted that this library does not encompass a general-purpose speech toolkit, nor does it try to achieve state-of-the-art results while sacrificing inference speed and model size.

The thunder-speech library draws significant inspiration from established best practices within the PyTorch community, particularly from Nemo ASR Toolkit, regarding original model code, fine-tuning, and prediction APIs. Moreover, the data loading and preprocessing mechanisms derive heavily from Fast.AI's principles (HOWARD; GUGGER, 2020) of separating item-level and batch-efficient computational patterns. Similar to PyTorch Lightning, this library tries to minimize code coupling via self-contained modules that try to reduce boilerplate. Finally, the thunder-speech library takes inspiration from the Transformers library style guide, featuring distinct folder structures providing insightful documentation about architectures and preprocessing activities coupled with rigorous testing suites.

During the training process for the language models employed in this work, we utilized the KenLM (HEAFIELD, 2011) library. This library provides efficient data structures for executing probabilistic n-gram language model queries, which are necessary for building accurate and scalable models. The use of compact trie data structures allows for rapid query performance while significantly reducing memory consumption compared to traditional approaches. Additionally, the `pyctcdecode`<sup>2</sup> library was used to implement the CTC decoding procedure, scoring the probabilities from the acoustic model with the language model and enabling more accurate and contextually-aware speech recognition results.

### 4.3 Evaluation Metrics

In the context of speech recognition, the performance of a system is commonly evaluated using metrics that quantify the difference between the recognized transcription and the true output. Two widely used metrics in this domain are character error rate (CER) and word error rate (WER). Both these metrics estimate the error in terms of edit distance, which measures the number of token substitutions, insertions, and deletions required to transform a predicted text into the reference text. In this section, we will discuss how these metrics work, how they can be computed from the edit distance, and provide examples.

The edit distance is computed using the following equation:

<sup>2</sup> Available at: <https://github.com/kensho-technologies/pyctcdecode>

$$edit\_distance = 100 \times \frac{Insertions + Substitutions + Deletions}{number\ of\ tokens\ in\ the\ reference} \% \quad (4.1)$$

An example is the calculation of the character error rate according to the original sentence and the prediction in Table 2. Two insertion operations, three deletions and one substitution are performed to transform the predicted sentence into the original. So, as there were 6 operations, and there are 28 characters in the original sentence (spaces are also taken into account in the calculation), the equivalent error rate is 21.42%. The word error rate follows the same principle, but using entire words as tokens.

Table 2 – Example of character error rate calculation.

	Sentence	Number of operations
original	O céu é azul e o sol amarelo	
prediction	Oh céu é azl e oh sol amriloh	
Insertion	Oh céu é az <u>u</u> l e oh sol am <u>a</u> riloh	2
Deletion	<u>O</u> h céu é azul e <u>o</u> h sol amariloh	3
Substitution	O céu é azul e o sol amare <u>l</u> o	1

To provide further insight into the impact of error rates on speech recognition outcomes, Table 3 presents a series of sentence pairs with progressively increasing character error rates. It is widely accepted that an error rate below 5% is necessary to facilitate human-computer interaction with minimal frustration arising from mispredictions. At around 15% error, the transcription can still be deciphered with the help of contextual cues; however, error rates above this threshold render the transcript too corrupt for accurate comprehension.

Table 3 – Example of character error rate at different levels.

CER of 4%	
original	cco atende a b tres quatro cinco na rh tres oito cambio
prediction	cco atende b tres quatro cinco na rg tres oito cambio
CER of 12%	
original	conferiu manutencao um zero nove na zero um zero tres singela cambio
prediction	conferiu maumutencao um zeronove na zero um zero tres cino de a cambio
CER of 29%	
original	ccm atende sete tres na meia um cambio
prediction	cmeia adento se tres a mea um cambio
CER of 50%	
original	manutencao b um zero nove na zero tres singela chamando cco cambio
prediction	a ae aoo mei um eeo laeco dera e qui quia lamando ccco ambio

## 4.4 Implementation Details

### 4.4.1 Optimized Training with Scarce Resources

Techniques to reduce the memory used by deep neural networks are crucial to handle the significant number of model parameters and memory requirements to hold gradients and optimizer states during training. Deep neural networks are typically trained using 32-bit (single-precision) floating point operations. However, using 16-bit (half-precision) floating point operations can cut memory usage in half, as demonstrated by Micikevicius et al. (2018).

To retain the model’s accuracy, one should take caution when using 16-bit numbers due to reduced numerical precision. Specifically: *(i)* weights, activations, and gradients should be stored and calculated in half-precision; *(ii)* single-precision copy of the weights that accumulate the gradient steps is also recommended; *(iii)* loss scaling is required to preserve small-magnitude gradients and prevent underflow problems; *(iv)* some numerically sensitive operations inside the model should be performed in single-precision, with its output converted to half-precision (mixed precision training is the training technique that uses single and half-precision simultaneously).

Another technique used to improve stability while training models is gradient accumulation. When training with batches of data, the gradient is first calculated for each input sample, then averaged across all batch elements before being passed to the optimizer. Gradient accumulation occurs when the averaging of gradient values is postponed until the model evaluates multiple batches of data, increasing the effective batch size without the additional memory consumption. As a result, this reduces the frequency in which the optimizer is called by the same accumulation factor<sup>3</sup>.

### 4.4.2 Model Specific Regularization Techniques

Regularization techniques prevent overfitting and improve generalization performance in machine learning models. In this study, we adopt a combination of dropout (SRIVASTAVA et al., 2014), layerdrop (FAN; GRAVE; JOULIN, 2020), and specaugment (PARK et al., 2019) techniques to regularize our multiple trained models. These approaches aim to introduce stochasticity during training, making the model more robust.

<sup>3</sup> The accumulation factor refers to the number of batches of data processed before updating the model parameters using the optimizer. For example, if the accumulation factor is four, then the model parameters will be updated only after processing four batches of data. This means that the gradients from these four batches of data are accumulated and averaged before the optimizer is called to update the model parameters.

Dropout is a widely adopted regularization method that alters the structure of the neural network during training. This technique randomly drops out a fraction of the hidden units from the network with probability  $p$  at every forward pass. By doing so, the network learns to distribute the importance of features across different neurons, leading to better generalizability. Layerdrop, on the other hand, extends the concept of dropout to entire layers instead of individual neurons. During training, some layers might be removed based on a random mask generated from a probability. As a result, the model learns to adapt to varying network depths, reducing the risk of overfitting.

Apart from these traditional regularization methods, we also apply specaugment, which introduces data augmentation tailored to spectrogram inputs. Specaugment generates random masks that remove contiguous blocks in frequency and time dimensions during training. This approach encourages the model to focus on the most informative parts of the signal while ignoring irrelevant or noisy components.

## **4.5 Results and Discussion**

### **4.5.1 Comparing training recipes**

In this section, we will compare different training recipes by executing an experiment that utilizes a fixed model architecture and dataset. The experiment aims to investigate how distinct hyperparameter choices and optimization techniques affect the training process of large speech recognition models under conditions of restricted labeled data. For this experiment, we selected the Libri-light dataset, a vast open-source corpus consisting of both unlabelled and labeled speech. In the unsupervised setting, only the unlabelled portion of this dataset is used to evaluate methods for constructing speech representations without supervision. On the other hand, in the semi-supervised setting, where models are trained with restricted supervision datasets, the unlabelled data is utilized in various ways, such as pretraining and obtaining pseudo-labels. This latter setting employs a constrained-resource training set (10 minutes, 1 hour, 10 hours). The testing sets used in this experiment are the same as those employed in LibriSpeech, making it convenient to compare weakly supervised results with state-of-the-art findings obtained through supervised learning.

Due to the limited computational resources available for running experiments, we start by using models that have been pretrained without supervision and then only perform the finetuning step using the labeled training set. To compare the various recipes, we will utilize the word error rate (WER) metric as a performance evaluation metric.

For this experiment, we are using the Wav2vec 2.0 model pretrained on the VoxPopuli dataset (WANG et al., 2021a). This model has been pretrained on over 100,000 hours of speech from 23 different languages. This large amount of pretraining helps make a strong foundation for the finetuning process that will happen later with our limited labeled data. All the different ways we are experimenting with training this model will not use any language models during evaluation. That is because language models significantly impact the results in these low-resource settings compared to the actual speech recognition model. Plus, when using a language model, the way it works with the speech recognition model must be carefully tuned to ensure everything works well together. To avoid accidentally choosing a worse training recipe just because its associated language model happened to have better hyperparameters, we will only compare the performance of the individual speech recognition models during evaluation.

We will be comparing three different training recipes. The first one is identified by fairseq and tries to follow the finetuning procedure of the original wav2vec implementation. Another method we are testing is the huggingface recipe, based on a training recipe recommended by the library that focuses primarily on transformer language models. Finally, our training recipe tries to use the best aspects of each previous training recipe while introducing some subtle changes that improve convergence speed and memory usage while training.

Table 4 shows some of the major hyperparameters used in our three training recipes. All of them use similar optimizers and base learning rates. The fairseq recipe uses the Adam optimizer with a custom beta parameter value. The other approaches use a variation called AdamW, which has fixes for weight decay. Some techniques like gradient clipping are applied to help with training when there's oscillation in loss values between batches during training. This can happen when the dataset has audio samples of different lengths. All three recipes apply learning rate scheduling. The fairseq recipe uses a three-state scheduler, starting with 10% of training epochs devoted to warmup, followed by a constant learning rate for 40% of the remaining time and decay for the last 50%. In contrast, the huggingface recipe uses a more straightforward scheduling method with a fixed number of warmup steps followed by linear decay. Our approach is similar to huggingface's but without any warm-up applied. This helps with convergence when used in combination with our freeze policy. Initially, we keep the encoder frozen and only train a single linear layer on top of it. There is no need for a warm-up in this case. When comparing the freeze policies, the huggingface approach always trains the entire model from the start. Both fairseq and our approaches initially freeze the encoder and then add its weights to the optimization process at some point during training. However, the big difference between the two is that while fairseq unfreezes the original encoder weights, we apply parameter efficient fine-tuning (peft) using the LoRa algorithm to add new weights to the encoder and unfreeze only these new weights.

Figure 9 displays the training loss curves associated with different training recipes. Initially, it



	Fairseq	Huggingface	Ours
Optimizer	Adam	AdamW	AdamW
Optimizer betas	[0.9, 0.99]	default	default
Weight decay	default	default	0.005
Gradient clip value	don't apply	1.0	1.0
Base learning Rate	0.03	0.003	0.03
Scheduler	tri-state	linear with warmup	linear
Warmup steps	10% of training (calculated)	100	0
Initial freeze policy	encoder frozen	train all from start	encoder frozen
Unfreeze policy	unfreeze encoder	don't apply	unfreeze peft weights

Table 4 – Multiple training recipes compared

can be seen that the HuggingFace recipe experiences convergence difficulties. These issues are linked to the training dynamics of models trained using the CTC loss function. As the model initially contains random weights, its predictions are similarly random during the initial phase. During the second stage, the model learns to produce only empty labels by assigning greater probabilities to blank tokens (ZEYER; SCHLÜTER; NEY, 2021). Finally, the model learns to generate accurate label sequences during the third stage. However, in this instance, the model becomes trapped in a local minimum where it solely produces empty labels due to several factors. These include employing a large model, starting the training process simultaneously across all layers despite random initialization for the decoder, and performing training in a low-data regime. Freezing the encoder until the new decoder begins accurately predicting tokens addresses these issues, as observed in the other training recipes, which prevent the encoder from becoming excessively preconditioned towards poor directions during the early stages of learning. Similar behavior can be observed if we unfreeze the encoder too soon on the other recipes.

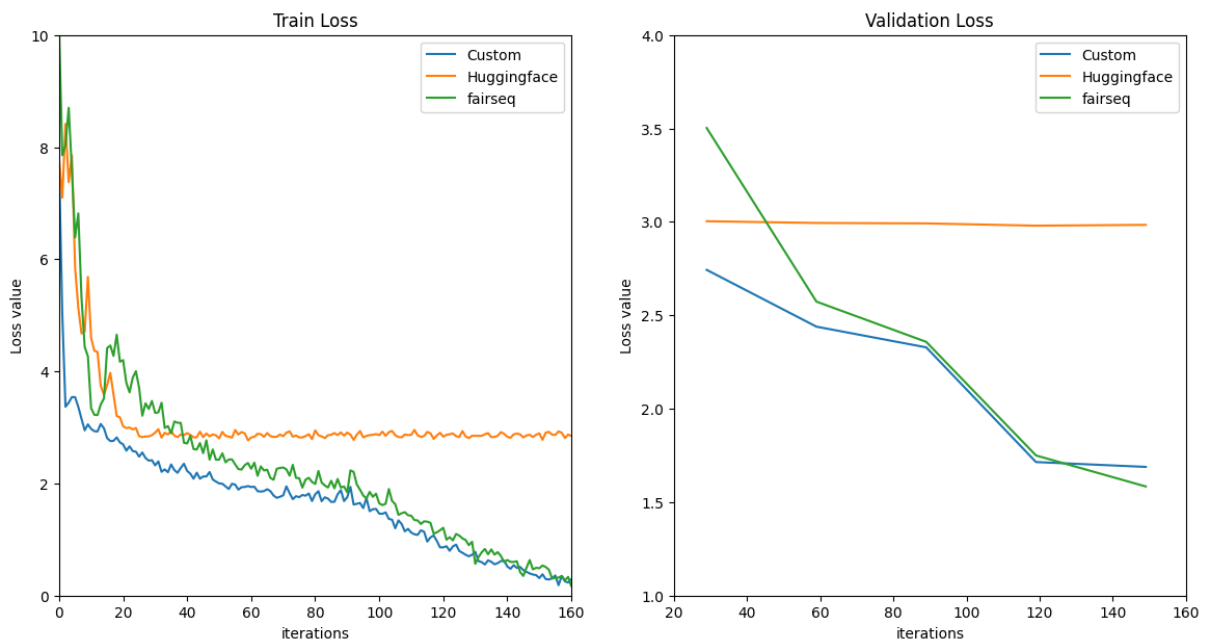


Figure 9 – Training and validation loss from different recipes.

Next, let us compare the fairseq recipe to ours. Both methods show comparable training curves. However, our approach includes additional optimizations. One such optimization is eliminating the need to gradually increase the learning rate during the initial stages of training. Since we start by training just a single linear layer in the decoder, this step is not required, allowing us to achieve faster convergence at the beginning of training. Another improvement is applying a parameter efficient fine-tuning method once we unfreeze the encoder. This reduces the number of trainable parameters, resulting in increased efficiency without significantly impacting overall performance. Specifically, the LoRA matrices incorporated into our model represent less than 5% of the entire weight count for the encoder, yet the performance in the test set is equivalent to complete model training. Overall, these modifications contribute to an enhancement in the speed and efficacy of our training procedure.

#### **4.5.2 Baseline model using radio data augmentation**

We utilized an optimized training recipe to build our baseline Wav2vec model and implemented strong data augmentation simulating speech over radio communication. To create the dataset, two hyperparameters that control the communication channel were defined: signal-to-noise ratio indicating the power of the white Gaussian signal that models the AWGN channel and frequency deviation caused by propagation in the channel. Five values for the signal-to-noise ratio representing incremental increases in noise level were chosen: [20, 10, 5, 3, 0] dB. Two values were used for frequency deviation: no distortion and 0.5% distortion. Higher values of distortion caused a loss of tuning in the system, preventing recovery of the voice signal by the receiver.

For each audio file, ten new versions were created using these hyperparameters and saved to disk, resulting in eleven available versions for each audio during training. When an audio is loaded during the training process, it is chosen randomly between the original and the new augmented versions, with equal probability for all eleven versions. This ensures that each audio is seen only once during a training epoch but introduces variability between different epochs, enhancing the effectiveness of data augmentation in our model training.

To assess the effectiveness of our proposed methodology, we trained two initial models utilizing distinct variations of the commonvoice dataset as part of our experimental methodology. The first model was trained on the original data to validate our approach and provide a comparison with related work. The second model employed data augmented through radio communication simulation.

In order to evaluate the models, we compared them against published outcomes in the literature

utilizing the original version of the commonvoice test set. Table 5 displays the obtained results. Both models developed within this study had higher error rates when compared to other works that also utilized the wav2vec architecture; however, their errors were lower than those obtained by (DUARTE; COLCHER, 2021).

The error rate achieved is in line with expectations, given the volume of data employed during training. While this work and (DUARTE; COLCHER, 2021) both used solely the commonvoice dataset for training, the two best results in the table were trained on a larger volume of data, with (GRIS et al., 2022) utilizing a union of seven different datasets during training. It can also be observed that training using simulated data led to a relative increase of 13.5% in error rate when evaluated using the original data.

Table 5 – Comparison of Character Error Rates (in %) on the Commonvoice Test Set

Method	Architecture	Error Rate
(GRIS et al., 2022)	Wav2vec 2.0	4.15
(JUNIOR et al., 2021)	Wav2vec 2.0	6.34
(DUARTE; COLCHER, 2021) <sup>4</sup>	DeepSpeech	30.0
This study (Commonvoice base)	Wav2vec 2.0	8.63
This study (Commonvoice augmented)	Wav2vec 2.0	9.80

Next, we modified the commonvoice test set using the same radio simulation as during training to create a new test set with controlled noise levels. This allowed us to evaluate how our speech recognition system performed in different noise environments. Table 6 shows the results for different signal-to-noise ratios when there is no frequency deviation in the transmission channel. The base model, trained only using original data, has a high error rate in all situations, even with low noise levels. This is because the limited amount of spectral information in these conditions significantly affects the quality of our predictions.

Table 6 – Character error rate on the augmented commonvoice test set (without frequency shift).

SNR (dB)	20	10	5	3	0
This study (Commonvoice base)	23.60	33.32	40.43	43.59	48.59
This study (Commonvoice augmented)	13.54	16.51	19.43	20.9	23.57

Table 7 displays the results when we simultaneously applied frequency deviation and Gaussian noise in the transmission channel. There was a higher error rate for all noise levels compared to Table 6. However, the most significant impact occurred on the base model, demonstrating the robustness developed by our augmented model.

Now that we understand how data augmentation affects our speech recognition model’s noise robustness let us create the baseline Wav2vec model using all publicly available training data.

<sup>4</sup> Value estimated using Figure 13 from (DUARTE; COLCHER, 2021)

Table 7 – Character error rate on the augmented commonvoice test set (with frequency shift)

SNR (dB)	20	10	5	3	0
This study (Commonvoice base)	26.31	35.41	41.99	44.87	49.73
This study (Commonvoice augmented)	13.88	16.91	19.85	21.33	24.29

We applied the same radio augmentation technique as before to train and evaluate the new model on our private test set. The goal of this test is to determine if using more out-of-domain data with augmentation leads to better error rates in real-world scenarios. Table 8 shows the results obtained by training our model with all publicly available data. When evaluated on a test set created from recordings made in real radio communication environments, the model trained with more augmented data had significantly lower error rates. This supports our hypothesis that increasing the amount of training data at this stage could be beneficial. However, the error rates obtained on our private test set are much higher than those on the Commonvoice test set, even when we applied the radio simulation. This indicates that relying solely on clever data augmentation with publicly available data is insufficient to achieve satisfactory performance on the target domain. In our next experiment, we will introduce data from the target domain as a solution to this problem.

Table 8 – Character error rate on the private test set, out-of-domain training.

Model	Error rate
This study (Commonvoice augmented)	52.24
This study (baseline model)	41.12

### 4.5.3 Self-training via pseudo-labeling

In this experiment, we focus our attention on the private dataset. After training our baseline Wav2vec model on a diverse open dataset, we can use it to create pseudo-labels for unlabeled audio segments from the private data. First, we split the audio files into distinct speech segments using a voice activity detector. Then, we run inference with a language model and calculate confidence scores for each prediction. We remove any predictions with low confidence to build a high-quality pseudo-labeled dataset.

When we applied this technique to the labeled portion of the private dataset, we found that 74 segments had problematic annotations. Most of these segments had high noise levels or overlapping speech, and the labels given by the annotator were based on a best guess. We removed these segments from the dataset, which led to a decrease in spikes in the loss during training.

With our full proprietary dataset, which contains both true labels and pseudo-labels, we distill our knowledge into two different model architectures. The first is the same Wav2vec model we have been using so far, while the second is a smaller QuartzNet model with similar trainable parameter budgets to the modified Wav2vec model when applying LoRa. This comparison will show how transformer-based and convolutional models perform with similar trainable parameter budgets.

Results from training both models using the full private dataset are present on Table 9. We can see that there’s a huge decrease in error rate when compared to the results obtained in Table 8, now that we are using in-domain data with the pseudo-labels and a small portion of manually labeled data. We can notice that the distillation process using pseudo-labels is effective, so that the smaller QuartzNet model could achieve a better result compared to the teacher model trained only on augmented data previously. On the other hand, retraining the same model with this private dataset has achieved the best results, with a 50% reduction in error rate when compared to the smaller model, and a 63.8% reduction in error rate when compared to the baseline model.

Table 9 – Character error rate on the private test set, in-domain training.

Model	Error rate
QuartzNet	28.04
Wav2Vec (final)	14.85

## 5 CONCLUSION

In our study, we have identified two major challenges associated with applying speech recognition techniques to low-resource languages like Portuguese in the context of radio communication recordings. Firstly, the scarcity of annotated data for these languages poses a significant challenge as it limits the amount of training data available for model development. Secondly, radio communication recordings are characterized by a high degree of variability in background noise and speaker characteristics compared to other audio datasets, which can affect the accuracy and robustness of speech recognition models.

To address these challenges, we have proposed a novel methodology that involves utilizing out-of-domain annotated data through data augmentation and self-training techniques using pseudo-labels generated from unlabeled in-domain data. Using out-of-domain data allows us to expand the training set and provide the model with more diverse input, leading to better generalization performance on new, unseen data. In addition, the generation of pseudo-labels from unlabeled in-domain data enables us to train our models using more data than would otherwise be available, further improving their performance.

Our results have shown significant improvements in character error rate (CER) compared to baseline models when applying this methodology. Specifically, we observed a relative reduction in CER of 51.7% at the most challenging noise level (SNR of 0 dB) when using simulated data for training. We also saw a decrease of 63.8% in CER when applying self-training techniques to in-domain data. These improvements demonstrate the effectiveness of our proposed methodology and suggest that it has the potential to facilitate the development of more robust speech recognition models for low-resource languages like Portuguese, even in the context of radio communication recordings.

As we continue our research in this area, we plan to investigate the usage of foundational models as teachers, which could lead to further improvements in model performance. Additionally, we aim to advance our radio augmentation simulation to make it more similar to the target domain, which will enable us to provide our models with a more realistic training environment and further enhance their robustness. Overall, these initiatives represent exciting opportunities for advancing the state-of-the-art in speech recognition for low-resource languages and have the potential to lead to practical applications beyond rail communication auditing.

## BIBLIOGRAPHY

ABDEL-HAMID, O.; MOHAMED, A.-r.; JIANG, H.; PENN, G. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition. In: IEEE. **2012 IEEE international conference on Acoustics, speech and signal processing (ICASSP)**. [S.l.], 2012. p. 4277–4280.

ALENCAR, V. F. S.; ALCAIM, A. Lsf and lpc - derived features for large vocabulary distributed continuous speech recognition in brazilian portuguese. In: **2008 42nd Asilomar Conference on Signals, Systems and Computers**. [S.l.: s.n.], 2008. p. 1237–1241.

AMODEI, D.; ANANTHANARAYANAN, S.; ANUBHAI, R.; BAI, J.; BATTENBERG, E.; CASE, C.; CASPER, J.; CATANZARO, B.; CHENG, Q.; CHEN, G. et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In: **International conference on machine learning**. [S.l.: s.n.], 2016. p. 173–182.

ARDILA, R.; BRANSON, M.; DAVIS, K.; KOHLER, M.; MEYER, J.; HENRETTY, M.; MORAIS, R.; SAUNDERS, L.; TYERS, F.; WEBER, G. Common voice: A massively-multilingual speech corpus. In: **Proceedings of the 12th Language Resources and Evaluation Conference**. Marseille, France: European Language Resources Association, 2020. p. 4218–4222. ISBN 979-10-95546-34-4. Disponível em: <<https://www.aclweb.org/anthology/2020.lrec-1.520>>.

BACHMAN, P.; HJELM, R. D.; BUCHWALTER, W. Learning representations by maximizing mutual information across views. In: **Proc. of NeurIPS**. [S.l.: s.n.], 2019.

BAEVSKI, A.; ZHOU, H.; MOHAMED, A.; AULI, M. **wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations**. 2020.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. **CoRR**, abs/1409.0473, 2014. Disponível em: <<https://api.semanticscholar.org/CorpusID:11212020>>.

BALAM, J.; HUANG, J.; LAVRUKHIN, V.; DENG, S.; MAJUMDAR, S.; GINSBURG, B. **Improving Noise Robustness of an End-to-End Neural Model for Automatic Speech Recognition**. arXiv, 2020. Disponível em: <<https://arxiv.org/abs/2010.12715>>.

BATISTA, C.; DIAS, A. L.; Sampaio Neto, N. Baseline Acoustic Models for Brazilian Portuguese Using Kaldi Tools. In: **Proc. IberSPEECH 2018**. [S.l.: s.n.], 2018. p. 77–81.

BENESTY, J.; SONDHI, M. M.; HUANG, Y. et al. **Springer handbook of speech processing**. [S.l.]: Springer, 2008. v. 1.

BLOSSOM, E. Gnu radio: Tools for exploring the radio frequency spectrum. **Linux J.**, Belltown Media, Houston, TX, v. 2004, n. 122, p. 4, jun 2004. ISSN 1075-3583.

CHAN, W.; JAITLEY, N.; LE, Q. V.; VINYALS, O. Listen, attend and spell. **arXiv preprint arXiv:1508.01211**, 2015.

CHEN, T.; KORNBLITH, S.; NOROUZI, M.; HINTON, G. A simple framework for contrastive learning of visual representations. **arXiv**, abs/2002.05709, 2020.

CHEN, T.; XU, B.; ZHANG, C.; GUESTRIN, C. Training deep nets with sublinear memory cost. **arXiv preprint arXiv:1604.06174**, 2016.

CHOLLET, F. Xception: Deep learning with depthwise separable convolutions. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2017. p. 1251–1258.

COLLOBERT, R.; PUHRSCHE, C.; SYNNAEVE, G. Wav2letter: an end-to-end convnet-based speech recognition system. **arXiv preprint arXiv:1609.03193**, 2016.

DENG, L.; LI, X. Machine learning paradigms for speech recognition: An overview. **Trans. Audio, Speech and Lang. Proc.**, IEEE Press, v. 21, n. 5, p. 1060–1089, may 2013. ISSN 1558-7916. Disponível em: <<https://doi.org/10.1109/TASL.2013.2244083>>.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805**, 2018.

DUARTE, J. C.; COLCHER, S. Building a noisy audio dataset to evaluate machine learning approaches for automatic speech recognition systems. **Monografias em Ciência da Computação**, PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO, n° 05/2021, 09 2021. ISSN 0103-9741.

FAN, A.; GRAVE, E.; JOULIN, A. Reducing transformer depth on demand with structured dropout. In: **International Conference on Learning Representations**. [s.n.], 2020. Disponível em: <<https://openreview.net/forum?id=SylO2yStDr>>.

FLETCHER, H.; GALT, R. H. The Perception of Speech and Its Relation to Telephony. **The Journal of the Acoustical Society of America**, v. 22, n. 2, p. 89–151, 06 2005. ISSN 0001-4966. Disponível em: <<https://doi.org/10.1121/1.1906605>>.

GAGE, P. A new algorithm for data compression. **C Users Journal**, McPherson, KS: R & D Publications, c1987-1994., v. 12, n. 2, p. 23–38, 1994.

GINSBURG, B.; CASTONGUAY, P.; HRINCHUK, O.; KUCHAIEV, O.; LAVRUKHIN, V.; LEARY, R.; LI, J.; NGUYEN, H.; ZHANG, Y.; COHEN, J. M. Stochastic gradient methods with layer-wise adaptive moments for training of deep networks. **arXiv preprint arXiv:1905.11286**, 2019.

GRAVES, A.; FERNÁNDEZ, S.; GOMEZ, F.; SCHMIDHUBER, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In: **Proceedings of the 23rd International Conference on Machine Learning**. New York, NY, USA: ACM, 2006. (ICML '06), p. 369–376. ISBN 1-59593-383-2.

GRIS, L. R. S.; CASANOVA, E.; OLIVEIRA, F. S. de; SOARES, A. da S.; CANDIDO-JUNIOR, A. Desenvolvimento de um modelo de reconhecimento de voz para o português brasileiro com poucos dados utilizando o wav2vec 2.0. In: SBC. **Anais do XV Brazilian e-Science Workshop**. [S.l.], 2021. p. 129–136.



GRIS, L. R. S.; CASANOVA, E.; OLIVEIRA, F. S. de; SOARES, A. da S.; JUNIOR, A. C. Brazilian portuguese speech recognition using wav2vec 2.0. In: **Computational Processing of the Portuguese Language: 15th International Conference, PROPOR 2022, Fortaleza, Brazil, March 21–23, 2022, Proceedings**. Berlin, Heidelberg: Springer-Verlag, 2022. p. 333–343. ISBN 978-3-030-98304-8. Disponível em: <[https://doi.org/10.1007/978-3-030-98305-5\\_31](https://doi.org/10.1007/978-3-030-98305-5_31)>.

HANNUN, A.; CASE, C.; CASPER, J.; CATANZARO, B.; DIAMOS, G.; ELSSEN, E.; PRENGER, R.; SATHEESH, S.; SENGUPTA, S.; COATES, A. et al. Deep speech: Scaling up end-to-end speech recognition. **arXiv preprint arXiv:1412.5567**, 2014.

HE, K.; FAN, H.; WU, Y.; XIE, S.; GIRSHICK, R. Momentum contrast for unsupervised visual representation learning. **arXiv**, abs/1911.05722, 2019.

HE, X.; DENG, L. Speech-centric information processing: An optimization-oriented approach. **Proceedings of the IEEE**, v. 101, n. 5, p. 1116–1135, 2013.

HEAFIELD, K. KenLM: Faster and smaller language model queries. In: CALLISON-BURCH, C.; KOEHN, P.; MONZ, C.; ZAIDAN, O. F. (Ed.). **Proceedings of the Sixth Workshop on Statistical Machine Translation**. Edinburgh, Scotland: Association for Computational Linguistics, 2011. p. 187–197. Disponível em: <<https://aclanthology.org/W11-2123>>.

HÉNAFF, O. J.; RAZAVI, A.; DOERSCH, C.; ESLAMI, S. M. A.; OORD, A. van den. Data-efficient image recognition with contrastive predictive coding. **arXiv**, abs/1905.09272, 2019.

HIGUCHI, Y.; MORITZ, N.; ROUX, J. L.; HORI, T. Momentum pseudo-labeling: Semi-supervised asr with continuously improving pseudo-labels. **IEEE Journal of Selected Topics in Signal Processing**, v. 16, n. 6, p. 1424–1438, 2022.

HINTON, G.; DENG, L.; YU, D.; DAHL, G. E.; MOHAMED, A.-r.; JAITLY, N.; SENIOR, A.; VANHOUCHE, V.; NGUYEN, P.; SAINATH, T. N.; KINGSBURY, B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. **IEEE Signal Processing Magazine**, v. 29, n. 6, p. 82–97, 2012.

HINTON, G.; VINYALS, O.; DEAN, J. Distilling the knowledge in a neural network. In: **NIPS Deep Learning and Representation Learning Workshop**. [s.n.], 2015. Disponível em: <<http://arxiv.org/abs/1503.02531>>.

HIRSCH, H.-G.; FINSTER, H. The simulation of realistic acoustic input scenarios for speech recognition systems. In: . [S.l.: s.n.], 2005. p. 2697–2700.

HOWARD, A. G.; ZHU, M.; CHEN, B.; KALENICHENKO, D.; WANG, W.; WEYAND, T.; ANDREETTO, M.; ADAM, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. **arXiv preprint arXiv:1704.04861**, 2017.

HOWARD, J.; GUGGER, S. Fastai: A layered api for deep learning. **Information**, v. 11, n. 2, 2020. ISSN 2078-2489. Disponível em: <<https://www.mdpi.com/2078-2489/11/2/108>>.

HSU, W.-N.; SRIRAM, A.; BAEVSKI, A.; LIKHOMANENKO, T.; XU, Q.; PRATAP, V.; KAHN, J.; LEE, A.; COLLOBERT, R.; SYNNAEVE, G.; AULI, M. **Robust wav2vec 2.0: Analyzing Domain Shift in Self-Supervised Pre-Training**. 2021.

HU, E. J.; SHEN, Y.; WALLIS, P.; ALLEN-ZHU, Z.; LI, Y.; WANG, S.; WANG, L.; CHEN, W. Lora: Low-rank adaptation of large language models. **arXiv preprint arXiv:2106.09685**, 2021.

HUANG, X.; ACERO, A.; HON, H.-W.; REDDY, R. **Spoken language processing: A guide to theory, algorithm, and system development**. [S.l.]: Prentice hall PTR, 2001.

IWAMOTO, K.; OCHIAI, T.; DELCROIX, M.; IKESHITA, R.; SATO, H.; ARAKI, S.; KATAGIRI, S. **How Bad Are Artifacts?: Analyzing the Impact of Speech Enhancement Errors on ASR**. 2022.

JUNIOR, A. C.; CASANOVA, E.; SOARES, A.; OLIVEIRA, F. S. de; OLIVEIRA, L.; JUNIOR, R. C. F.; SILVA, D. P. P. da; FAYET, F. G.; CARLOTTO, B. B.; GRIS, L. R. S.; ALUÍSIO, S. M. **CORAA: a large corpus of spontaneous and prepared speech manually validated for speech recognition in Brazilian Portuguese**. 2021.

JURAFSKY, D.; MARTIN, J. **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**. Pearson Prentice Hall, 2009. (Prentice Hall series in artificial intelligence). ISBN 9780131873216. Disponível em: <<https://books.google.com.br/books?id=fZmj5UNK8AQC>>.

KAHN, J.; LEE, A.; HANNUN, A. Self-training for end-to-end speech recognition. In: **ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2020. p. 7084–7088.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **ICLR**, 2015.

KINOSHITA, K.; OCHIAI, T.; DELCROIX, M.; NAKATANI, T. Improving noise robust automatic speech recognition with single-channel time-domain enhancement network. In: **ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2020. p. 7009–7013.

LEE, D.-H. Pseudo-label : The simple and efficient semi-supervised learning method for deep neural networks. **ICML 2013 Workshop : Challenges in Representation Learning (WREPL)**, 07 2013.

LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. In: **International Conference on Learning Representations**. [S.l.: s.n.], 2018.

LUGOSCH, L.; LIKHOMANENKO, T.; SYNNAEVE, G.; COLLOBERT, R. Pseudo-labeling for massively multilingual speech recognition. In: **ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2022. p. 7687–7691.

LUO, J.; WANG, J.; CHENG, N.; XIAO, E.; XIAO, J.; KUCSKO, G.; O'NEILL, P.; BALAM, J.; DENG, S.; FLORES, A.; GINSBURG, B.; HUANG, J.; KUCHARIEV, O.; LAVRUKHIN, V.; LI, J. Cross-language transfer learning and domain adaptation for end-to-end automatic speech recognition. In: **2021 IEEE International Conference on Multimedia and Expo (ICME)**. [S.l.: s.n.], 2021. p. 1–6.

MARCUS, G. **Deep Learning: A Critical Appraisal**. 2018.

MICIKEVICIUS, P.; NARANG, S.; ALBEN, J.; DIAMOS, G.; ELSSEN, E.; GARCIA, D.; GINSBURG, B.; HOUSTON, M.; KUCHARIEV, O.; VENKATESH, G.; WU, H. Mixed precision training. In: **International Conference on Learning Representations**. [s.n.], 2018. Disponível em: <<https://openreview.net/forum?id=r1gs9JgRZ>>.

MISRA, I.; MAATEN, L. van der. Self-supervised learning of pretext-invariant representations. **arXiv**, 2019.

MITCHELL, T. M.; MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-hill New York, 1997. v. 1.

MITTAG, G.; NADERI, B.; CHEHADI, A.; MÖLLER, S. Nisqa: A deep cnn-self-attention model for multidimensional speech quality prediction with crowdsourced datasets. In: **Interspeech**. [s.n.], 2021. Disponível em: <<https://api.semanticscholar.org/CorpusID:233296150>>.

NARAYANAN, A.; MISRA, A.; SIM, K. C.; PUNDAK, G.; TRIPATHI, A.; ELFEKY, M.; HAGHANI, P.; STROHMAN, T.; BACCHIANI, M. Toward domain-invariant speech recognition via large scale training. In: **IEEE. 2018 IEEE Spoken Language Technology Workshop (SLT)**. [S.l.], 2018. p. 441–447.

NASSIF, A. B.; SHAHIN, I.; ATTILI, I.; AZZEH, M.; SHAALAN, K. Speech recognition using deep neural networks: A systematic review. **IEEE Access**, IEEE, v. 7, p. 19143–19165, 2019.

O'MALLEY, T.; NARAYANAN, A.; WANG, Q.; PARK, A.; WALKER, J.; HOWARD, N. A conformer-based asr frontend for joint acoustic echo cancellation, speech enhancement and speech separation. In: **2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)**. [S.l.: s.n.], 2021. p. 304–311.

PARK, D. S.; CHAN, W.; ZHANG, Y.; CHIU, C.-C.; ZOPH, B.; CUBUK, E. D.; LE, Q. V. Specaugment: A simple data augmentation method for automatic speech recognition. **Interspeech 2019**, ISCA, Sep 2019. Disponível em: <<http://dx.doi.org/10.21437/Interspeech.2019-2680>>.

PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DEVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; LERER, A. Automatic differentiation in PyTorch. In: **NeurIPS Autodiff Workshop**. [S.l.: s.n.], 2017.

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTMLOYER, L. Deep contextualized word representations. In: **Proc. of ACL**. [S.l.: s.n.], 2018.

POL'AK, P.; BOJAR, O. Coarse-to-fine and cross-lingual asr transfer. In: **ITAT**. [S.l.: s.n.], 2021.

POVEY, D.; GHOSHAL, A.; BOULIANNE, G.; BURGET, L.; GLEMBEK, O.; GOEL, N.; HANNEMANN, M.; MOTLICEK, P.; QIAN, Y.; SCHWARZ, P. et al. The kaldi speech recognition toolkit. In: **IEEE SIGNAL PROCESSING SOCIETY. IEEE 2011 workshop on automatic speech recognition and understanding**. [S.l.], 2011.

PRATAP, V.; XU, Q.; SRIRAM, A.; SYNNAEVE, G.; COLLOBERT, R. Mls: A large-scale multilingual dataset for speech research. **Interspeech 2020**, ISCA, Oct 2020. Disponível em: <<http://dx.doi.org/10.21437/Interspeech.2020-2826>>.

QUINTANILHA, I. M. End-to-end speech recognition applied to brazilian portuguese using deep learning. **MSc dissertation**, PEE/COPPE, Federal University of Rio de Janeiro, 2017.

QUINTANILHA, I. M.; NETTO, S. L.; BISCAINHO, L. W. P. An open-source end-to-end asr system for brazilian portuguese using dnns built from newly assembled corpora. **Journal of Communication and Information Systems**, v. 35, n. 1, p. 230–242, 2020.

RABINER, L. A tutorial on hidden markov models and selected applications in speech recognition. **Proceedings of the IEEE**, v. 77, n. 2, p. 257–286, 1989.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. **Improving Language Understanding by Generative Pre-Training**. 2018. <[https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language\\_understanding\\_paper.pdf](https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf)>.

RAVANELLI, M.; BENGIO, Y. Speaker recognition from raw waveform with sincnet. In: **IEEE. 2018 IEEE Spoken Language Technology Workshop (SLT)**. [S.l.], 2018. p. 1021–1028.

RIBEIRO, D. J. **DESENVOLVIMENTO DE UM DATASET DE SINAIS DE ÁUDIO E UM MODELO DE ROTULAÇÃO ORIENTADO A APLICAÇÕES DE RECONHECIMENTO DE FALA**. 87 p. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal do Espírito Santo, nov. 2020.

SALESKY, E.; WIESNER, M.; BREMERMAN, J.; CATTONI, R.; NEGRI, M.; TURCHI, M.; OARD, D. W.; POST, M. The multilingual tedx corpus for speech recognition and translation. **CoRR**, abs/2102.01757, 2021. Disponível em: <<https://arxiv.org/abs/2102.01757>>.

SCART, L. G. **RECONHECIMENTO AUTOMÁTICO DE FALA EM PORTUGUÊS UTILIZANDO ARQUITETURAS DE REDES NEURAIS PROFUNDAS**. 42 p. Monografia (Trabalho de Conclusão de Curso) — Universidade Federal do Espírito Santo, dez. 2019.

SEGBROECK, M. V.; NARAYANAN, S. S. A robust frontend for asr: Combining denoising, noise masking and feature normalization. In: **2013 IEEE International Conference on Acoustics, Speech and Signal Processing**. [S.l.: s.n.], 2013. p. 7097–7101.

SRIVASTAVA, N.; HINTON, G.; KRIZHEVSKY, A.; SUTSKEVER, I.; SALAKHUTDINOV, R. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, Ł.; POLOSUKHIN, I. Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.

WANG, C.; RIVIERE, M.; LEE, A.; WU, A.; TALNIKAR, C.; HAZIZA, D.; WILLIAMSON, M.; PINO, J.; DUPOUX, E. VoxPopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation. In: **Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)**. Online: Association for Computational Linguistics, 2021. p. 993–1003. Disponível em: <<https://aclanthology.org/2021.acl-long.80>>.

WANG, Y.; LI, J.; WANG, H.; QIAN, Y.; WANG, C.; WU, Y. **Wav2vec-Switch: Contrastive Learning from Original-noisy Speech Pairs for Robust Speech Recognition**. 2021.

YOSHIOKA, T.; GALES, M. Environmentally robust asr front-end for deep neural network acoustic models. **Computer Speech & Language**, v. 31, n. 1, p. 65–86, 2015. ISSN 0885-2308. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0885230814001259>>.

YU, D.; DENG, L. **Automatic Speech Recognition - A Deep Learning Approach**. Springer, 2014. Disponível em: <<https://www.microsoft.com/en-us/research/publication/automatic-speech-recognition-a-deep-learning-approach/>>.

ZEGHIDOUR, N.; USUNIER, N.; KOKKINOS, I.; SCHAIZ, T.; SYNNAEVE, G.; DUPOUX, E. Learning filterbanks from raw speech for phone recognition. In: **2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2018. p. 5509–5513.

ZEYER, A.; SCHLÜTER, R.; NEY, H. Why does CTC result in peaky behavior? **arXiv preprint arXiv:2105.14849**, 2021.

# Annex

## ANNEX A – LEARNING PARADIGMS

According to Tom Mitchell, the learning problem that is the focus of machine learning methods can be defined as (MITCHELL; MITCHELL, 1997):

A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$ , if its performance of tasks in  $T$ , as measured by  $P$ , improves with experience  $E$ .

Machine learning methods use a collection of data samples to perform their learning. Depending on the kinds of labels used, and their origin, the learning can be categorized in the following methodologies:

- **Supervised Learning.** It is the most common form of learning, where the goal is to learn a relationship  $y = f(x)$ , when the dataset consists of pairs  $(x, y)$ . Because the training data is labeled, the learning algorithm can directly estimate the relationship, with the expectation that the learned mapping will generalize to unseen data.
- **Semi-supervised Learning.** The objective is to use a small amount of labeled data  $(x_l, y_l)$  with a large amount of unlabeled  $(x_u, )$  data, both sets related to the learning task, during the training of a supervised model. This objective is achieved with a technique called pseudo-labeling. First, an initial supervised model is trained with the labeled pairs to learn an approximation of the desired mapping, that is,  $y_l = f(x_l)$ . Predictions from this model are then used to create pseudo-labels for the unlabeled data  $(x_u, \hat{y})$ , where  $\hat{y} = f(x_u)$ . Some filtering procedures discard points where the prediction has low confidence, keeping only the high-quality pseudo labels. A new supervised model is then trained with the extended dataset incorporating true and pseudo labels.
- **Unsupervised Learning.** Only the input samples  $(x, )$  are present in this methodology. Patterns of interest must be extracted from this unlabeled data, by exploiting the relationship between different samples and their innate structure. Examples of unsupervised learning tasks include dimensionality reduction, clustering and anomaly detection.
- **Self-supervised Learning.** This methodology is often associated with a pre-training step, because it focuses on learning meaningful representations from unlabeled data  $(x, )$  that another learning methodology will later use. It tries to formulate a supervised learning criterion, where the labels are extracted from the input alone. One example is contrastive

learning, where the model is trained to predict whether or not two augmented data views were created from the same input sample. Another example is the masked language modeling (DEVLIN et al., 2018), where part of the input tokens are masked, and the model tries to predict them based on the surrounding context.



## ANNEX B – DEEP NEURAL NETWORKS

Deep neural networks are machine learning models that perform pattern recognition using architectures with multiple stacked layers performing non-linear operations trained with data. Using the backpropagation algorithm, it's possible to calculate the gradient of each parameter in relation to a cost function, enabling the use of an optimization process that tries to minimize the cost function by changing the parameter values. Among the different kinds of neural networks, convolutional neural networks and transformers are the most popular choices used to build speech recognition systems. We will explain both models in the following sections.

### B.1 Convolutional Neural Networks

This type of neural network is constituted by convolution layers followed by activation functions and pooling layers. The convolution layers have the property of operating locally on their inputs, thus providing equivariance in relation to translations in the model input.

The main element of a convolutional network is the convolution layer, which gets its name from the operation it performs. The layer's input is a three-dimensional tensor with size  $(C \times H \times W)$ , where  $C$  is the channel numbers,  $H$  is the height, and  $W$  is the width. The layer further has a set of associated weights named filters, each filter of size  $(C \times H_f \times W_f)$ , where  $H_f$  and  $W_f$  are the lengths of the filter. The convolution process is characterized by performing a sliding window of filters over the input tensor, performing at each position an inner product to obtain a scalar. By arranging all the generated scalars according to the relative position between the filter and the input tensor, we obtain a map of activations for each layer filter.

The dimensions of the generated activation map depend on the input and output dimensions, plus two hyper-parameters: stride and padding. Stride controls the distance between two successive applications of the filters during the sliding window process. Thus, a stride of one means that the filters move only one position between two realized inner products. Padding is the addition of values at the edges of the input to allow the sliding window process to take full advantage of the data present at the edges or maintain the same dimension between the input and output tensors. Common choices for the value used in padding include zero or the mean value.

When applied to automatic speech recognition, convolutional models take advantage of their high ability to extract features in a position-invariant manner on the input.

## B.2 Transformer

The Transformer architecture (VASWANI et al., 2017) is a popular deep learning model for sequence to sequence tasks such as machine translation. Unlike traditional recurrent neural network (RNN) architectures that rely on sequential processing, the Transformer processes an entire input sequence all at once using self-attention mechanisms. This allows the Transformer to efficiently process long sequences, making it well suited for natural language processing tasks. The key components of the Transformer include:

- **Attention Mechanism:** A method for weighing different parts of the input when producing an output. In the case of the Transformer, this takes the form of scaled dot-product attention.
- **Multi-Head Attention:** Multiple parallel instances of the attention mechanism with separate learned parameters. This allows the model to focus on multiple aspects of the input simultaneously.
- **Positional Embeddings:** Added to the input embeddings to provide information about the position of each token within the sequence.
- **Transformer Blocks:** Stacked layers consisting of multi-head attention and feedforward networks.

### B.2.1 Attention Mechanism

At the core of the Transformer is the attention mechanism, which computes a weighted sum of input elements based on their relevance to each other. Given three inputs: queries ( $Q$ ), keys ( $K$ ), and values ( $V$ ), the basic formula for calculating the output  $O$  can be expressed as follows:

$$O = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V$$

where  $d_k$  is the dimension of the keys. The softmax function ensures that the weights sum up to one, while  $QK^T$  calculates the similarity between query-key pairs, allowing the model to attend to relevant portions of the input. To increase efficiency, the authors propose scaling the dot product by  $\sqrt{d_k}$ .

Scaled Dot-Product Attention has two main advantages over alternative methods like additive attention or concatenated attention: computational efficiency and simplicity. It requires fewer

operations than the alternatives, reducing memory requirements and increasing speed during training and inference. Additionally, its simple implementation makes it easier to scale to longer sequences without sacrificing performance.

### B.2.2 Multi-Head Attention

In order to allow the model to capture various types of relationships among input tokens, the Transformer employs Multi-Head Attention. This involves running several independent instances of Scaled Dot-Product Attention called “heads” in parallel, each with its own set of learnable parameters. By combining these heads through a linear layer, the final representation captures diverse features from the input. Mathematically, Multi-Head Attention can be defined as follows:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad i = 1, \dots, h$$

Here,  $W_i^Q$ ,  $W_i^K$ ,  $W_i^V$ , and  $W^O$  are learnable parameter matrices, and  $h$  denotes the number of heads. Each head specializes in paying attention to specific patterns in the data, improving overall modeling capacity.

### B.2.3 Positional Embeddings

Since the Transformer does not have any inherent notion of position due to its parallel nature, positional embeddings must be added to preserve ordering information. These embeddings incorporate sinusoidal functions of varying frequencies into the input representations, enabling them to represent positions even beyond the length of the original input. Consequently, the model can generalize better to unseen lengths compared to embedding techniques relying solely on fixed-size vectors.

Formally, given an integer  $pos$  representing the absolute position of a token in a sequence, we compute the corresponding positional embedding  $PE(pos)$  as:

$$PE_{i,2j} = \sin\left(\frac{pos}{10000^{2j/d_{\text{model}}}}\right)$$

$$PE_{i,2j+1} = \cos\left(\frac{pos}{10000^{2j/d_{\text{model}}}}\right)$$

Where  $i$  indicates the dimensional index,  $j$  ranges from 0 to  $(d_{model}/2) - 1$ , and  $d_{model}$  represents the embedding size.

#### **B.2.4 Transformer Blocks**

A Transformer block consists of alternating layers of Multi-Head Self-Attention and Pointwise Feed Forward Networks, followed by residual connections, dropout, and normalization. Specifically, a single Transformer block contains the following sublayers:

1. **Layer Normalization:** Applied before every sublayer, helping stabilize training by standardizing activations across timesteps.
2. **Multi-Head Self-Attention:** Computes attention scores internally, attending to different locations within the same input sequence.
3. **Residual Connection:** Summing the input and the output of the previous sublayer, encouraging gradient flow during backpropagation.
4. **Dropout:** Randomly sets some neuron outputs to zero, preventing overfitting.
5. **Positionwise FeedForward Network:** Consists of two fully connected layers separated by ReLU activation; applied independently to each position in the sequence.

By stacking multiple Transformer blocks together, complex hierarchical structures can be captured, leading to improved performance on challenging NLP tasks.