UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

# Classificação de Variações Linguísticas do Português do Brasil por meio da Fala

**Ariadne Nascimento Matos**

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

**Ariadne Nascimento Matos**

# Classificação de Variações Linguísticas do Português do Brasil por meio da Fala

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestra em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Moacir Antonelli Ponti

**USP – São Carlos**
**Junho de 2024**

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

M434c

Matos, Ariadne Nascimento
Classificação de Variações Linguísticas do Português do Brasil por meio da Fala / Ariadne Nascimento Matos; orientador Moacir Antonelli Ponti. -- São Carlos, 2024.
121 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2024.

1. PROCESSAMENTO DIGITAL DE VOZ. 2. APRENDIZADO DE MÁQUINA. 3. VARIAÇÃO LINGUÍSTICA. 4. LÍNGUA PORTUGUESA. 5. FONÉTICA. I. Ponti, Moacir Antonelli, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

**Ariadne Nascimento Matos**

# Classification of Linguistic Variations in Brazilian Portuguese using Speech

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Moacir Antonelli Ponti

**USP – São Carlos**
**June 2024**

# AGRADECIMENTOS

Agradeço a Deus, o Alfa e Ômega de todas as coisas. A fonte de sabedoria inesgotável e quem guia meus passos.

Aos meus pais Joelma Cristina e Jairo Ilson. Sou muito grata pelo apoio de vocês, pela confiança constante, pelas palavras de sabedoria. Agradeço especialmente a minha mãe, pois sempre acreditou na educação. Obrigada por apontar o caminho, sua força, sabedoria e inteligência são fontes de inspiração e auxiliam inúmeras pessoas, por fazer sempre da sua vida uma sala de estar para a educação, seja na sala de aula ou fora dela. Obrigada pelos seus esforços constantes em proporcionar o melhor, por sempre está comigo em todas as circunstâncias. Esse trabalho aconteceu porque um dia a senhora acreditou, seu apoio foi fundamental. As minhas irmãs Indyra e Beatriz por serem combustível de alegria e pela companhia de vocês. Obrigada por tudo e por tanto

Agradeço ao professor Moacir Ponti e Arnaldo Cândido pela orientação e coorientação excepcionais. Ao professor Moacir, pela paciência constante em ensinar e explicar de forma clara e objetiva e sua expertise que é sem dúvidas, diferenciada. Sou grata também por sua compreensão e apoio. Ao professor Arnaldo Cândido por me ensinar pacientemente, desde termos mais simples aos mais avançados, por sua didática notória, por me auxiliar na escrita e domínio na área. Esse trabalho é fruto da orientação de vocês. Agradeço a professora Sandra Aluísio também pelas ricas contribuições ao longo desse trabalho.

Aos professores Paulo Ambrósio e Thiago Barbosa por terem concedido a oportunidade de iniciar na área científica e pela confiança.

Aos meus colegas do NILC e amigos por todo o apoio e companhia. Em especial, a minha amiga de longa data Maria Luíza, por seu apoio e alegria nos momentos desafiadores, por ser uma amiga irmã nessa jornada.

Agradeço a USP - ICMC por proporcionarem um ambiente propício ao desenvolvimento da pesquisa e por fomentarem da melhor forma possível com estrutura e profissionais capacitados.

Agradeço à Nathália Batista e ao professor Dr. Lee Luan Ling, que possibilitaram o desenvolvimento desta pesquisa, e pela disponibilização da base de dados Braccent.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001

*“Tudo quanto vier à mão para realizar, faze-o com o melhor das tuas forças.”*
*(Eclesiastes 9:10)*

# RESUMO

MATOS, A. N. **Classificação de Variações Linguísticas do Português do Brasil por meio da Fala**. 2024. 121 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

As variações linguísticas estão presentes em diversas localidades e fazem parte do cotidiano. Por meio delas, é possível identificar a origem linguística de uma pessoa. Classificar essas variações é importante para aplicações voltadas ao processamento de fala, sobretudo para melhorar sistemas de reconhecimento automático. Neste trabalho, com o objetivo de auxiliar na classificação das variações linguísticas do Português Brasileiro, foram exploradas redes convolucionais e técnicas que incorporam o mecanismo de atenção, como o Wav2vec 2.0 XLSR e o Audio Spectrogram Transformer. Os experimentos foram conduzidos em dois cenários: um com poucos locutores e outro com muitos locutores, utilizando três conjuntos de dados distintos: Spotify Podcasts, CORAA-ASR e Braccent. Conforme relatado na literatura, os cenários closed-set, nos quais a validação é realizada no mesmo conjunto de dados de treinamento, não refletem adequadamente a realidade. Portanto, foi adotada a validação com um conjunto de dados diferente do conjunto de treinamento, conhecida como validação cruzada. Os resultados indicaram que, mesmo no cenário closed-set, os modelos enfrentaram dificuldades para classificar as variações linguísticas com mais de duas classes. Além disso, foi observado que é necessária uma maior diversidade de locutores para abranger determinado sotaque e alcançar um desempenho satisfatório dos modelos. Para a classificação binária com muitos locutores, o modelo Wav2vec 2.0 XLSR obteve sucesso tanto no cenário closed-set, com um F1-score geral de 83%, quanto no cenário de validação cruzada, com 75%. As contribuições deste trabalho incluem o desenvolvimento de um classificador de regionalismos para Pernambuco e São Paulo capital, além da criação de subconjuntos derivados do dataset do Spotify Podcasts, abrangendo nove variações linguísticas. Apesar dos avanços significativos, a classificação dos sotaques brasileiros ainda é um desafio e exige a exploração de novas abordagens para cenários multiclasse.

**Palavras-chave:** Classificação de Variações Linguísticas, Reconhecimento Automático, Variações Linguísticas, Wav2vec 2.0 XLSR.

# ABSTRACT

MATOS, A. N. **Classification of Linguistic Variations in Brazilian Portuguese using Speech**. 2024. 121 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2024.

Linguistic variations are present in various locations and are part of everyday life. Through them, it is possible to identify a person's linguistic origin. Classifying these variations is important for applications focused on speech processing, particularly to enhance automatic recognition systems. In this work, aiming to assist in the classification of linguistic variations in Brazilian Portuguese, convolutional networks and techniques incorporating attention mechanisms, such as Wav2vec 2.0 XLSR and Audio Spectrogram Transformer, were explored.Experiments were conducted in two scenarios: one with few speakers and another with many speakers, using three different datasets: Spotify Podcasts, CORAA-ASR, and Braccent. As reported in the literature, closed-set scenarios, where validation is performed on the same training dataset, do not adequately reflect reality. Therefore, validation with a different dataset from the training set, known as cross-validation, was adopted. Results indicated that, even in the closed-set scenario, models faced difficulties in classifying linguistic variations with more than two classes. Additionally, it was observed that a greater diversity of speakers is necessary to encompass a particular accent and achieve satisfactory model performance. For binary classification with many speakers, the Wav2vec 2.0 XLSR model succeeded in both the closed-set scenario, with an overall F1-score of 83%, and the cross-validation scenario, with 75%. Contributions of this work include the development of a regionalism classifier for Pernambuco and São Paulo capital, as well as the creation of subsets derived from the Spotify Podcasts dataset, covering nine linguistic variations. Despite significant advances, classifying Brazilian accents remains a challenge and requires exploration of new approaches for multiclass scenarios.

**Keywords:** Linguistic Variations Classification, Automatic Recognition, Linguistic Variations, Wav2vec 2.0 XLSR.

# LISTA DE ILUSTRAÇÕES

# LISTA DE QUADROS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| AP20-OLR | Oriental Language Recognition Challenge 2020 |
| ASR | *Automatic Speech Recognition* |
| BiLSTM | Bidirectional Long Short Term Memory |
| CFPB | Corpus Forense do Português Brasileiro |
| CNNs | Convolutional Neural Networks |
| CTC | Connectionist Temporal Classification |
| EER | Equal error rate |
| FFT | Fast Fourier Transform |
| GD | Gradient Descent |
| GMM-UBM | Gaussian Mixture Model - Universal Background Model |
| HMMs | Hidden Markov Models |
| LSTM | Long Short Term Memory |
| MTL | Multi-task learning |
| PLN | Processamento de Língua Natual |
| PSD | power spectral density |
| ReLU | Rectified Linear Unit |
| RNN-T | Recurrent Neural Network Transducer |
| RNNs | Recurrent Neural Networks |
| SGD | Stochastic Gradient descent |
| SVM | Suport Vector Machine |
| TDNN | Time Delay Neural Network |
| WER | Word Error Rate |

# SUMÁRIO

CAPÍTULO

1

# INTRODUÇÃO

A fala é uma forma fundamental de comunicação humana. Por meio dela, é possível expressar ideias, informações e atender a diversos propósitos, como a compreensão e análise do discurso. Dada sua importância para o ser humano, há um incentivo em estudar métodos automáticos para processar e entender fala. Técnicas que utilizem inteligência artificial e aprendizado de máquina podem ser particularmente úteis nesse cenário.

Com o avanço dos recursos computacionais e a crescente disponibilidade de dados, o reconhecimento automático de fala, também conhecido como *Automatic Speech Recognition* (ASR) ganhou viabilidade. Essa tarefa é definida como o mapeamento de um sinal acústico que contém uma sentença falada para uma sequência de palavras escritas, ou texto. De acordo com Yu e Deng (2015), esta área tem sido considerada uma ponte importante para melhorar a comunicação entre humanos e máquinas. A tecnologia de reconhecimento de fala aliada a um tradutor automático, por exemplo, permite que pessoas de diferentes idiomas se comuniquem ou que seja possível a transcrição automática de mensagens de voz deixadas por um interlocutor. Além disso, é notável o sucesso de assistentes virtuais por comando de voz em dispositivos inteligentes, como a Siri[1] e a Alexa[2], as quais contam também com métodos de síntese de voz (CASANOVA *et al.*, 2022) para interagir com os usuários.

Sistemas de ASR tradicionais são compostos por quatro componentes principais: processamento de sinal e extração de características, modelo acústico, modelo de língua e, por fim, a combinação destes componentes com o léxico para decodificar o sinal (RISTA; KADRIU, 2020). Esses primeiros métodos centraram-se, portanto, na extração de características de forma manual e utilização dos Modelos Ocultos de Markov (Hidden Markov Models (HMMs)) (BAUM; EAGON, 1967). No entanto, progressos com técnicas de aprendizado profundo tornaram possível o treinamento de modelos ponta-a-ponta (ou *end-to-end*, em inglês) e que requerem menor quantidade de etapas explícitas.

[1] https://www.apple.com/br/siri/
[2] https://alexa.amazon.com/

Assim, mais recentemente tem sido utilizadas as redes neurais profundas com arquiteturas convolucionais (Convolutional Neural Networks (CNNs)), as quais processam dados de forma espacial, e/ou recorrentes (Recurrent Neural Networks (RNNs)), que processam dados em sequência. Mais recentemente o Transformer (VASWANI *et al.*, 2017), que utiliza um mecanismo conhecido por "atenção" alcançou o estado-da-arte (PAPASTRATIS, 2021)[3]. Assim, a medida que os modelos avançam em complexidade, faz-se necessário que se tenha uma maior cobertura possível do vocabulário do idioma (CHAN *et al.*, 2016).

Na Língua Portuguesa, para a qual as bases de dados disponíveis ainda são limitadas, a área tem apresentado importantes avanços, como o uso do método DeepSpeech2 para a tarefa de ASR (QUINTANILHA; NETTO; BISCAINHO, 2020), e o estudo de Gris *et al.* (2021), que utilizam o método Wav2vec 2.0 com poucos dados para o reconhecimento de voz para o português brasileiro. Apesar dos significativos resultados e que possibilitam variadas aplicações, ainda existem dificuldades no que diz respeito à amostra representativa de diferentes variedades linguísticas do português falado no Brasil (ALCAIM; SOLEWICZ; MORAES, 2015).

Por variação linguística entende-se como o padrão de pronúncia e acústica na fala que pode identificar a origem linguística de uma pessoa (TEIXEIRA; TRANCOSO; SERRALHEIRO, 1996). Existem dois tipos diferentes de variações linguísticas. O primeiro se refere ao estrangeirismo, que ocorre quando a pessoa fala uma língua utilizando algumas das regras e sons de outra língua, e o outro tipo ocorre dentro da própria língua materna (TEIXEIRA; TRANCOSO; SERRALHEIRO, 1996). Sendo assim, considerando a diferença entre as variações da fala em cenários reais, identificar e usar as características dessas variações para aplicações em processamento de fala, como melhorar a qualidade do ASR, ainda é um desafio (DENG; CAO; MA, 2021).

No inglês, há trabalhos com o objetivo de identificar variações para auxiliar na tarefa de ASR com o uso de diferentes tipos de arquiteturas (JAIN; UPRETI; JYOTHI, 2018; SHI *et al.*, 2021a). Outros trabalhos, como o de Wang *et al.* (2018), propuseram uma arquitetura *end-to-end* para diferentes tipos de sotaques de diferentes países, e outros trabalhos propuseram diferentes arquiteturas de redes profundas (DENG; CAO; MA, 2021; HÄMÄLÄINEN *et al.*, 2021; WENINGER *et al.*, 2019).

Para o português brasileiro, o primeiro trabalho voltado para a identificação automática de sete tipos de variações linguísticas foi proposto por Batista (2019). Dois outros trabalhos utilizam a base de dados proposta por Batista (2019). O primeiro modelando outras arquiteturas de redes neurais (TOSTES *et al.*, 2021), e um segundo que compara serviços em nuvem para transcrição de fala considerando áudios com variações regionais (LOPES; ANDRADE; KOMATI, 2021).

No entanto, no que diz respeito à classificação de variações linguísticas no Brasil, ainda há muito a ser realizado. De acordo com Alcaim, Solewicz e Moraes (2015), as realizações

[3] Mais detalhes sobre essas arquiteturas serão mencionados no Capítulo 2

acústicas dos fonemas são altamente dependentes do contexto em que aparecem, e considerando as diferenças existentes entre as regiões do Brasil, o contexto muda e a pronúncia do que foi falado também. Tendo em vista as diferentes variações linguísticas existentes no português Brasileiro, justifica-se investigar modelos para classificação essas variações, com vistas a analisar o que realmente as arquiteturas de redes profundas aprendem nesse contexto. Isso pode impactar, também na melhoria de tarefas de ASR.

## 1.1 Justificativa e Motivação

A classificação de variações linguísticas no português brasileiro ainda é uma área que necessita de aprimoramentos. Embora alguns avanços significativos tenham sido alcançados, ainda existem lacunas a serem preenchidas. Por exemplo, um dos trabalhos apresentados por Tostes *et al.* (2021) obteve resultados promissores com o uso de arquiteturas que combinam redes convolucionais, Long Short Term Memory (LSTM) e Bidirectional Long Short Term Memory (BiLSTM). No entanto, é importante destacar que esses resultados não foram avaliados utilizando a técnica de validação externa, que permite verificar a capacidade de generalização dos modelos para outros conjuntos de dados.

O estudo de Batista (2019) mostra que, apesar de ser possível alcançar altas taxas de acerto em um mesmo *dataset*, em condições de validação externa (ou *cross-dataset*) há uma degradação nas métricas dos modelos de classificação com relação às variedades linguísticas. Além disso, com relação ao reconhecimento automático de fala, o artigo de Lopes, Andrade e Komati (2021) indica que sistemas de transcrição de áudio disponíveis comercialmente tem performance diferente para diferentes variações linguísticas existentes no Brasil. Em particular, mostraram que falas de cariocas e nortistas são mal reconhecidas pelos sistemas comerciais analisados.

Assim, é relevante desenvolver classificadores de variações linguísticas que contemplem as diferentes variantes existentes e funcionem em cenários de validação externa. Para isso, não se deve limitar a apenas identificar a variedade linguística prevista, mas também considerar a interpretabilidade do resultado obtido. É igualmente importante avaliar a capacidade de generalização dos modelos para outros conjuntos de dados, sobretudo os modelos mais recentes na literatura e verificar se os mesmos conseguem obter resultados semelhantes aos alcançados em outros idiomas.

## 1.2 Objetivos e Contribuições

O objetivo geral deste trabalho é o desenvolvimento de um classificador de variedade linguística que seja capaz de obter taxa de acerto similar nas diferentes variedades do Português do Brasil, com foco particular na generalização para diferentes datasets.

Os objetivos específicos deste trabalho consistem em:

1. Projetar e avaliar modelos estudando arquiteturas e estratégias de treinamento para o problema;
2. Treinar os modelos com áudios de falantes de diversas regiões do Brasil, realizando validação interna (obtendo métricas dentro de um mesmo *dataset*) e externa (obtendo métricas entre diferentes *datasets*), e comparando com resultados existentes na literatura;
3. Investigar quais *datasets* possuem melhor capacidade de transferência para outros conjuntos;
4. Analisar métricas por variedade linguística, melhorando o entendimento sobre quais são mais difíceis de aprender;
5. Utilizar técnicas de explicabilidade para melhorar o entendimento sobre os padrões aprendidos pelo modelo.

   As principais contribuições deste trabalho são:

   a) Organização de dois subconjuntos de dados do dataset Spotify Podcasts, consistindo em aproximadamente 90 horas de segmentos de áudio para fala espontânea, com aproximadamente 204 locutores contendo variações linguísticas de 9 estados brasileiros;

   b) Estudo com diferentes configurações dos conjuntos de dados para fala espontânea e preparada ( Spotify Podcasts, CORAA ASR e Braccent) em dois cenários distintos, *closed-set* (validação interna) e *cross-dataset* (validação externa), obtendo conclusões importantes sobre os desafios da tarefa de classificação e fornecendo *insights* sobre melhores maneiras de resolver o problema sob um cenário mais realista.

## 1.3 Pergunta de Pesquisa e Hipótese

Considerando a quantidade de variações linguísticas existentes no português brasileiro, quais modelos de aprendizado profundo são mais adequados para classificar as variedades em diferentes tipos de *datasets*?

A hipótese relacionada à esta pergunta, é que um modelo com mecanismos de atenção e apoiado por parâmetros pré-treinados consiga generalizar melhor para diferentes *datasets* considerando diferentes variações linguísticas.

## 1.4 Organização da Dissertação

A presente monografia é organizada em cinco capítulos.

No Capítulo 2, é apresentada a fundamentação teórica, principiando pelas redes neurais artificiais como o Perceptron. São abordados alguns conceitos-chave como o Backpropagation e as funções de ativação, e também redes profundas como as convolucionais, Autoencoder e recorrentes, além de conceitos importantes como o mecanismo de atenção, a arquitetura do Transformer, Wav2vec 2.0, Vision Transformer e Audio Spectrogram Transformer. Nesse capítulo, também são abordados conceitos relacionados ao áudio e processamento, como frequência, amplitude, amostragem, quantização e espectrogramas e uma seção apresentado alguns aspectos fonéticos e fonológicos do Português do Brasil.

O Capítulo 3 é composto pelos materiais e métodos. Os materiais abrangem os *datasets*, *hardware* e *software* utilizados para o desenvolvimento. Nos métodos, são apresentadas as técnicas utilizadas para pré-processamento do *dataset*, protocolo experimental utilizado, como foi realizada a divisão dos conjuntos de dados para treinamento, avaliação e explicabilidade.

O Capítulo 4 são apresentados os resultados e discussões. Os resultados são apresentados em dois cenários: Cenário com poucos locutores e com muitos locutores. São apresentados os resultados obtidos para cada um dos modelos e *datasets* utilizados e por fim os resultados obtidos aplicando uma técnica de explicabilidade.

No Capítulo 5 a conclusão e trabalhos futuros. Por fim, no Apêndice são apresentados o artigo publicado, e gráficos contendo as métricas de precisão, revocação e F1-score para cada modelo para a classificação multiclasse.

## 1.5 Publicações

MATOS, A.; ARAÚJO, G.; JUNIOR, A. C.; PONTI, M. Accent classification is challenging but pre-training helps: a case study with novel Brazilian Portuguese datasets. In: Proceedings of the 16th International Conference on Computational Processing of Portuguese. Santiago de Compostela, Galicia/Spain: Association for Computational Linguistics, 2024. p. 364–373. Disponível em: <https://aclanthology.org/2024.propor-1.37>.

CAPÍTULO

2

# FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados conceitos sobre o processamento de áudio e arquiteturas de redes neurais relacionadas ao reconhecimento automático de fala. Para tanto, serão introduzidos primeiramente os conceitos sobre as redes neurais, desde seu surgimento até os conceitos basilares da aprendizagem profunda ou *deep learning*, bem como os Transformers.

Desde 1943, diversos pesquisadores contribuíram para o avanço de sistemas de aprendizagem como as redes neurais, com o desenvolvimento de modelos cada vez mais detalhados, tendo como base o neurônio biológico e abrangendo, assim, o campo da neurociência computacional (RUSSELL; NORVIG, 2010).

## 2.1 Variedades linguísticas no português falado

Nesta Seção serão apresentadas as variedades linguísticas do Português brasileiro, contendo inicialmente uma fundamentação sobre o processamento e a representação espacial do sinal de áudio. Por fim, os aspectos fonéticos, fonológicos e as principais características das variedades linguísticas no Português do Brasil.

### *2.1.1 Fundamentação do Processamento de Áudio*

O processamento digital, de acordo com McLoughlin (2009), é o método utilizado para o tratamento do áudio e da fala, uma vez que as aplicações e sistemas de áudio são predominantemente de natureza digital. Como fenômeno físico, o som refere-se a ondas que se originam em qualquer lugar e depois viajam através de algum meio para outro local, onde podem ser ouvidas ou medidas (CHRISTENSEN, 2019).

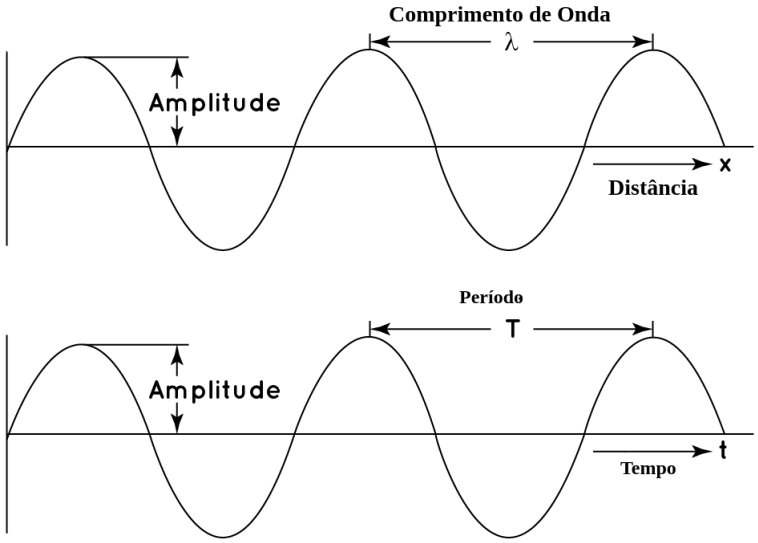
A análise automática do áudio, conforme Giannakopoulos e Pikrakis (2014), tem apresentado interesse crescente, sendo a tarefa de traduzir um sinal de fala para texto um dos domínios mais antigos da análise de áudio. De acordo com Christensen (2019), o processamento digital

refere-se basicamente a qualquer coisa que se pretenda realizar com os sinais sonoros, desde a manipulação dos sinais de áudio para fins musicais, por exemplo, até a identificação do que a pessoa está falando ou sotaque, por exemplo.

Considerando a importância dos aspectos que englobam o processamento de áudio, tem-se conceitos principais como a frequência, amplitude, amostragem, quantização e espectrograma, que são importantes para a compreensão da fala.

**Frequência e Amplitude**: O sinal sonoro para o sistema de reconhecimento de fala, tal como a entrada para o ouvido humano, é uma série complexa de mudanças na pressão do ar (JURAFSKY; MARTIN, 2009). As ondas sonoras, nesse caso são representadas, conforme Jurafsky e Martin (2009), traçando a mudança na pressão do ar ao longo do tempo. Essa variação de pressão da onda sonora, segundo Weenink (2022), pode ser modelada com uma função senoidal.

Figura 1 – Onda senoidal




Fonte: Adaptada de Koulidis (2017).

Conforme a Figura 1 em que são retratados o domínio do espaço e o tempo ( eixo $x$), a amplitude dessa função senoidal corresponde a altura da onda e o período marcado por um tempo $T$ que conforme Weenink (2022), corresponde ao tempo necessário para completar um ciclo, definido pela Equação 2.1. O número de períodos da onda que cabem em um segundo é denominado frequência da onda; a dimensão ou a unidade da frequência é chamada hertz, abreviada como Hz (WEENINK, 2022).

$$T = \frac{1}{f} \tag{2.1}$$

**Amostragem e Quantização**: De acordo com Jurafsky e Martin (2009), o primeiro passo na digitalização de uma onda sonora é converter as representações analógicas para um sinal

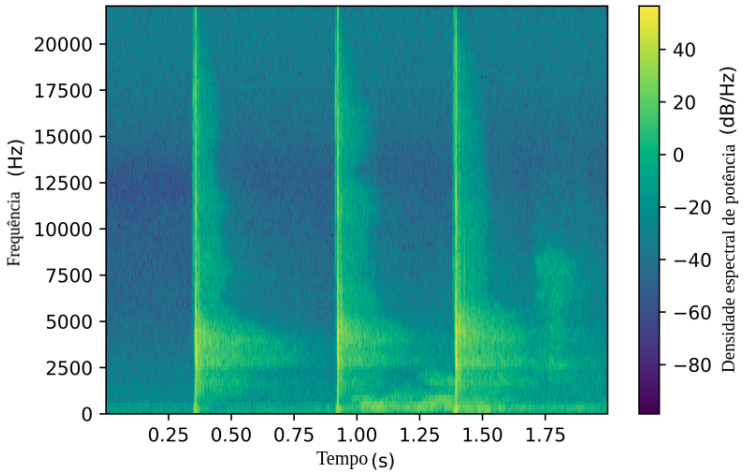
digital. Essa conversão analógico-digital é composta por duas etapas: amostragem e quantização. Para colher amostras de um sinal, são medidas as amplitudes em um determinado momento; a taxa de amostragem é o número de amostras colhidas por segundo (JURAFSKY; MARTIN, 2009).

Na amostragem, o sinal no domínio analógico é convertido para o digital, a amostragem uniforme é a mais comumente utilizada. O período de tempo entre duas amostras seguidas do sinal ($\Delta T$) define a taxa de amostragem do conversor através da equação $T = \frac{1}{\Delta T}$, que representa a quantidade de amostras realizadas por unidade de tempo, geralmente expressa em Hz (SILVA *et al.*, 2014).

Segundo McLoughlin (2009), as amostras de áudio precisam ser quantificadas de alguma forma durante a conversão de quantidades analógicas para representações finitas no computador. Na quantização, números de valor real são representados como inteiros. Sendo assim, é definida como o processo de mapear um grande conjunto de valores de entrada para um conjunto menor de valores discretos.

**Espectrograma e Espectrograma Mel**: Embora algumas características fonéticas como sonoridade, presença de sonoridade, por exemplo, possam ser interpretadas diretamente a partir da forma de onda, a maioria das aplicações computacionais, como os sistemas de reconhecimento de fala, são baseados em uma representação diferente do som em termos de frequência (JURAFSKY; MARTIN, 2009). Sendo assim, para à maioria dessas aplicações utiliza-se o espectrograma que é uma forma de inferir dados de áudio e convertê-los em uma imagem onde o eixo vertical representa a frequência do áudio. O eixo horizontal representa o tempo, enquanto a intensidade da cor caracteriza à amplitude da frequência em determinado tempo (KARIM, 2022), como representado pela Figura 2.

Figura 2 – Representação do espectrograma do sinal




Fonte: Adaptada de Khodzhaev (2022).

Os sinais variam na sua amplitude com o tempo, para analisar esses sinais deve ser

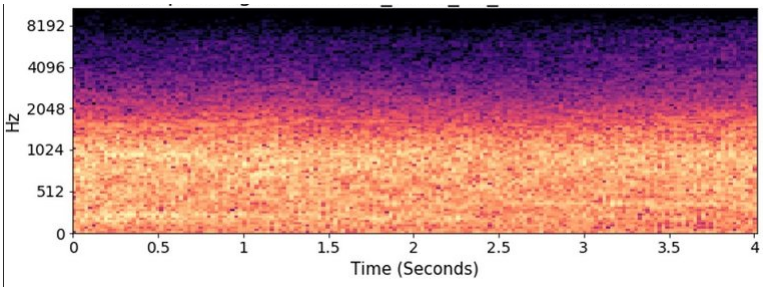
estabelecida uma forma de onda para que essa possa ser visualizada e transformada em dados úteis (KHODZHAEV, 2022). Conforme Oppenheim (1970), a razão principal para a importância do espectrograma sonoro na análise da fala é que muitos sons podem ser considerados como sendo produzidos pelo trato vocal, com uma excitação quase periódica ou ruidosa.

Segundo Oppenheim (1970), a transformada rápida de Fourier ou Fast Fourier Transform (FFT), fornece um mecanismo para gerar o espectrograma de forma eficiente. A FFT é comumente utilizada para converter o sinal do domínio do tempo para o domínio da frequência, permitindo analisar o conteúdo espectral do sinal ao longo de pequenos intervalos de tempo (MERGU; DIXIT, 2012). Isso é essencial para a construção do espectrograma, que é uma representação visual do espectro de frequência do sinal em função do tempo (OPPENHEIM, 1970). Os dados amostrados digitalmente, no domínio do tempo, segundo Mergu e Dixit (2012), são desagregados em partes que normalmente se sobrepõem, então a transformada de Fourier calcula a magnitude do espectro de frequência para cada janela.

Os espectros ou segmentos do tempo são então colocados lado a lado para formar uma imagem ou superfície tridimensional (EMRESOY; EL-JAROUDI, 1998). Na Figura 2, de acordo com Khodzhaev (2022), o espectrograma do sinal é um gráfico do tempo versus a frequência e contém a magnitude da densidade espectral de potência ou PSD (do inglês, *power spectral density*) do sinal por barra de cores . O power spectral density (PSD), segundo Dempster (2001), fornece uma forma de representar a distribuição dos componentes da frequência do sinal.

O espectrograma mel, como representado na Figura 3, difere-se do espectrograma por ter as frequências convertidas para a escala de mel, que é uma transformação logarítmica aplicada aos componentes do domínio da frequência (LACERDA *et al.*, 2021), possibilitando representar os componentes do som que o ouvido humano é mais sensível. Conforme Solovyev *et al.* (2020), se o ser humano se adapta a um som de 1000 Hz, então quando a frequência do som é aumentada para 2000 Hz, o ouvido humano não consegue perceber essa mudança. Se a frequência é baixa, a frequência mel muda rapidamente, caso contrário, se a frequência é alta, a frequência mel muda lentamente (SOLOVYEV *et al.*, 2020).

Figura 3 – Representação do espectrograma mel

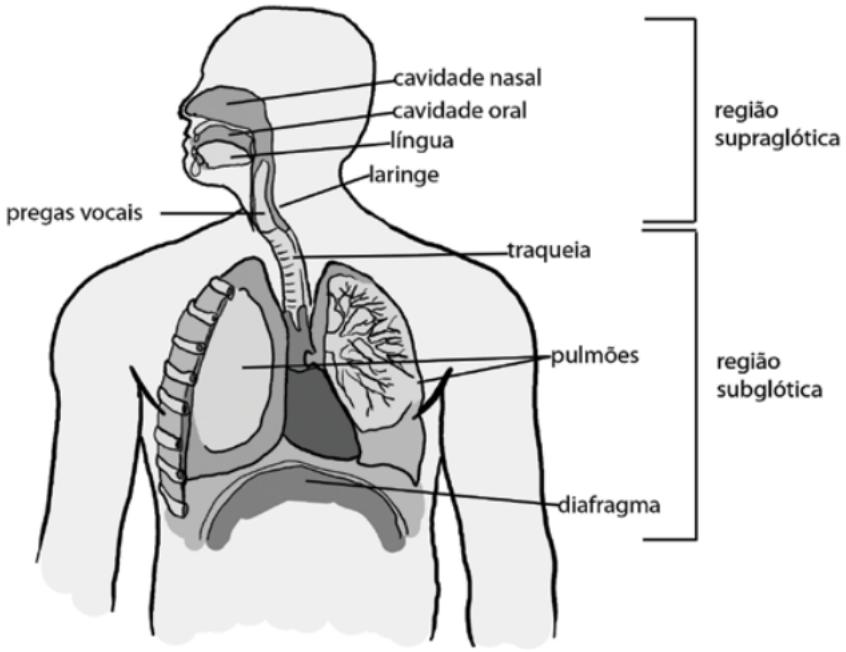



Fonte: Hoai, Oh e Yi (2020).

### 2.1.2 Aspectos Fonéticos e Fonológicos

De acordo Seara, Nunes e Lazzarotto-Volcão (2017) a voz pode ser definida como o som produzido a partir da vibração das pregas vocais enquanto a fala é o resultado da articulação desse som. Para produção de qualquer som em qualquer idioma faz-se uso de uma parte específica do corpo humano denominada aparelho fonador (SILVA, 2003). Conforme apresentado na Figura 4, o aparelho fonador é composto por três grupos: sistema respiratório, sistema fonatório e articulatório. Conforme Silva (2003), o sistema respiratório consiste nos pulmões, brônquios e traqueia; o fonatório pela laringe onde estão localizadas as cordas vocais e o articulatório por faringe, língua, dentes e lábios.

Figura 4 – Aparelho Fonador




Fonte: PARKER (2007).

A fonética estuda os sons como entidades físico-articulatórias isoladas, a fonologia irá estudar os sons do ponto de vista funcional como elementos que integram um sistema linguístico determinado (CALLOU; LEITE, 1990). Assim, à fonética cabe descrever os sons da linguagem e analisar suas particularidades articulatórias, acústicas e perceptivas. À fonologia cabe estudar as diferenças de significação (CALLOU; LEITE, 1990). Em linhas gerais, a fonologia preocupa-se em encontrar padrões de sons em diferentes línguas, de modo a verificar como eles estabelecem relações para formar unidades maiores, como sílaba e palavra (MADRUGA, 2017). A unidade da fonética é o som da fala ou o fone, enquanto a fonologia é o fonema (CALLOU; LEITE, 1990). Ao substituir o [a] da palavra **tapo** por [i], por exemplo, tem-se a palavra **tipo**, sendo assim [a] e [i] representam dois fonemas, que correspondem portanto, aos aspectos fonológicos de um idioma. Em relação aos aspectos fonéticos a letra pode apresentar sons distintos, por exemplo, o [o] pode ter um som de [u] (fechado) ou ainda semifechado, [o] (BATISTA, 2019).

De acordo com Silva (2003) na fonética existem os segmentos fonéticos. No segmento consonantal, o som da fala é moldado por diferentes graus de obstrução do trato vocal, também conhecido como modo de articulação (SILVA, 2003). Existem diferentes modos de articulação. A seguir são tem-se os modos de articulação definidos conforme Silva (2003) e Seara, Nunes e Lazzarotto-Volcão (2017):

- **oclusiva/plosiva**: obstrução total do fluxo de ar nas cavidades supraglóticas. Na palavra [p]aga o som [p] é emitido com obstrução total nos lábios
- **nasal**: ocorre uma obstrução total e momentânea do fluxo de ar nas cavidades nasais. Palavras com sons nasais: *mano*, *banho*. Em [m]a[n]o o som do [m] apresenta obstrução no trato oral e o som do [n] nos alvéolos
- **fricativa**: é produzida com um estreitamento do canal bucal. A passagem do ar nas cavidades supraglóticas geram um ruído de fricção. Palavra com consoante fricativa: *fava*. As duas consoantes [f] e [v] são produzidas com o lábio inferior se dirigindo para os dentes superiores. Ocorre uma obstrução parcial.
- **africada**: ocorre oclusão total e momentânea do fluxo de ar. Palavras *tia* [ˈtʃia] e *dia* [ˈdʒia].
- **tepe**: produzida com oclusão total e rápida do fluxo de ar. Palavras *caro* [ˈkaɾʊ] e *prato* [ˈpraɾʊ].

Além desses, tem-se: vibrante, retroflexa e lateral. Vibrante é quanto a ponta da língua provoca oclusões totais muito breves, como por exemplo, na palavra *roda*. Na retroflexa, ocorre um encurvamento da ponta da língua em direção ao palato duro. Esse som é percebido na pronúncia do /r/ no dialeto caipira, em palavras como *porta* [ˈpoɾta]. E por fim, lateral, em que o ar escapa pelas laterais do trato oral. Ocorre nas palavras *lata*, *sal* e *telha*, por exemplo (SEARA; NUNES; LAZZAROTTO-VOLCÃO, 2017).

### 2.1.3 Variedades Linguísticas no Português do Brasil

Segundo Seara, Nunes e Lazzarotto-Volcão (2017) a prosódia é parte da fonologia que estuda traços fônicos que se acrescentam aos sons da fala. Para a prosódia, três parâmetros acústicos são considerados: a intensidade, a curva de $f_0$ (*pitch*) e a duração. Em suma, tom, intensidade, duração ou *pitch* são as propriedades inerentes ao som e estão relacionadas com as características acústicas das ondas sonoras (MATEUS, 2004).

No Português Brasileiro, o caráter regional das variações são marcados por alguns dos seguintes traços de pronúncia (ILARI; BASSO, 2009):

- palatização de /s/ e /z/ finais de sílaba e da palavra: <mais> pronunciado [ˈmaɪʃ], <rapaz> [ʀaˈpaɪʃ]. É marca registrada da fala carioca, mas também pode ser encontrada no Espírito Santo e em algumas regiões de Minas Gerais, Pará, Amazonas e também em Pernambuco ( Recife )

- realização de /v/ e /ʒ/ como /h/ em início de palavra: <vamos> pronunciado [hamʊ], <gente> pronunciado [hẽtʃɪ]. Área: regiões do Nordeste, principalmente Ceará

- diferentes realizações do /r/ ( <r> de carro)

- ausência de palatização de /t/ e /d/: a palatização (<dente, pratinho,disco > pronunciados [ˈdẽtʃɪ], [praˈtʃiɲʊ], [ˈdʒiskʊ] é generalizado em todo território brasileiro, com exceção do interior de São Paulo e da região Sul (<leite quente> pronunciado [ˈleɪte ˈkẽte] encontrado em alguns lugares do Nordeste.

- palatização de /t/ e /d/ antes do /a/ e /o/: <oito,muito> pronunciados [ˈoɪtʃʊ],[ˈmʊɪtʃʊ].

- presença de entonação descendente: < sei não > pronunciado com um contorno descendente presente no Nordeste.

- abertura das vogais pré-tônicas: <decente> pronunciado [dɛˈsẽtʃɪ] presente no Nordeste.

- pronúncia retroflexa do /r/: palavra <porta> pronunciada [ˈpɔɾtɑ]. Característica do dialeto caipira presente no interior de São Paulo, algumas regiões do sul de Minas Gerais e Mato Grosso, por exemplo.

Devido aos traços de pronúncia característicos de lugares específicos no Brasil, foi proposto o desenvolvimento de um Atlas Linguístico com o objetivo de demarcar a área em que ocorre determinado fenômeno linguístico, delimitando assim, variedades regionais de uma língua (ILARI; BASSO, 2009). O primeiro Atlas linguístico foi desenvolvido por Nascentes (1953) que separou no Brasil dois grupos: Norte, compreendendo o amazônico e o nordestino; Sul, compreendendo o baiano, mineiro, fluminense e o sulista. Atualmente tem-se o Projeto AliB que tem como objetivo descrever a realidade linguística brasileiro (ILARI; BASSO, 2009), englobando diferentes estados e com uma divisão distinta da proposta inicialmente por Nascentes (1953) .

## 2.2 Redes Neurais clássicas

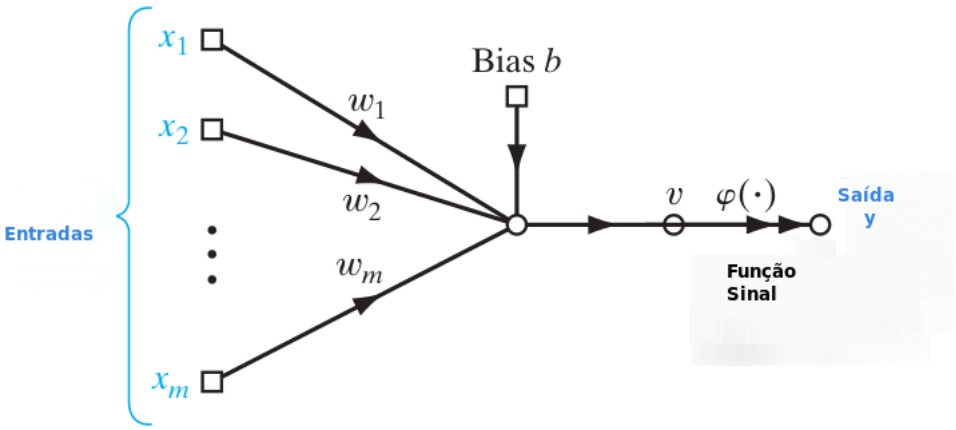
Nesta Seção são apresentadas as redes neurais clássicas como o Perceptron, Multilayer Perceptron, destacando como ocorre o treinamento utilizando o Gradiente Descendente e o Backpropagation. Ao final da seção são apresentadas as principais funções de ativação e de custo utilizadas durante o treinamento.

### 2.2.1 Perceptron

De acordo com Russell e Norvig (2010), o conceito de redes neurais foi introduzido por McCulloch e Pitts (1943), que, baseados no neurônio biológico, trouxeram a ideia do neurônio artificial e o descreveram matematicamente. No entanto, a implementação do primeiro modelo, denominado Perceptron e proposto por Rosenblatt (1958), foi o início da aprendizagem supervisionada, que permite que o modelo aprenda uma função mapeando uma entrada para uma saída respectiva.

Segundo Haykin (2009), o Perceptron pode ser representado como na Figura 5 e é constituído por um único neurônio. Tem-se um vetor de entradas com valores reais, representados por $x_1$, $x_2$,...,$x_m$. Após a entrada desses valores, é realizada a multiplicação com os respectivos pesos, representados por $w_1$,$w_2$,..,$w_m$, ou seja, para cada entrada há um peso específico. Em seguida, é adicionado o *bias* x, representado por $b$. A Figura 5, que consiste no processo de ativação, conforme Haykin (2009), é representado matematicamente pela Equação 2.2, que consiste no cálculo do potencial de ativação.

Figura 5 – Representação do Perceptron




Fonte: Adaptada de Haykin (2009).

$$v = \sum_{i=1}^{m} w_i x_i + b \tag{2.2}$$

Portanto, o potencial de ativação $v$ da Figura 5 é definido pela Equação 2.2 e é sujeito à aplicação da função de ativação sinal representada por $\varphi$, conforme representado pela Equação 2.3:

$$\varphi(v) = \begin{cases} +1 & v \geq 0 \\ -1 & v < 0 \end{cases} \tag{2.3}$$

A função de ativação, que será abordada nas próximas seções, desempenha um papel importante na capacidade da rede neural de aprender funções complexas e não lineares. Essa função é responsável por transformar a saída de um neurônio em uma forma não linear. Um exemplo de função de ativação é a função sinal, representada pela Equação 2.3. Ela transforma a saída y em $+1$ para valores de entrada maiores que 0 e em $-1$ para valores menores que 0. Essa função pode ser utilizada em redes neurais como o Perceptron, no entanto, existem outras funções com melhor desempenho que serão apresentadas no decorrer deste capítulo.

Segundo Aggarwal (2018), em um cenário em que cada exemplo de treinamento é representado como $(x,y)$, onde $x = [x_1, x_2, ..., x_m]$ denota um vetor de características com $m$ atributos e $y \in [-1, +1]$ indica a classe correspondente. Em aplicações práticas, como classificação de sotaques, essas características podem representar aspectos fonéticos de uma amostra, enquanto a variável da classe o sotaque correspondente a essa amostra.

Posto isso, o treinamento do Perceptron, conforme Mitchell (1997), consiste em aprender um vetor de pesos, que é inicializado aleatoriamente e ajustado iterativamente para cada exemplo de treinamento. Os pesos são modificados se a rede classificar o exemplo de forma incorreta. Matematicamente, a atualização dos pesos $w_i$ associados a uma entrada $x_i$ pode ser definida pela seguinte equação:

$$w_i \leftarrow w_i + \Delta w_i \tag{2.4}$$

onde:

$$\Delta w_i = \eta (y - \hat{y}) x_i \tag{2.5}$$

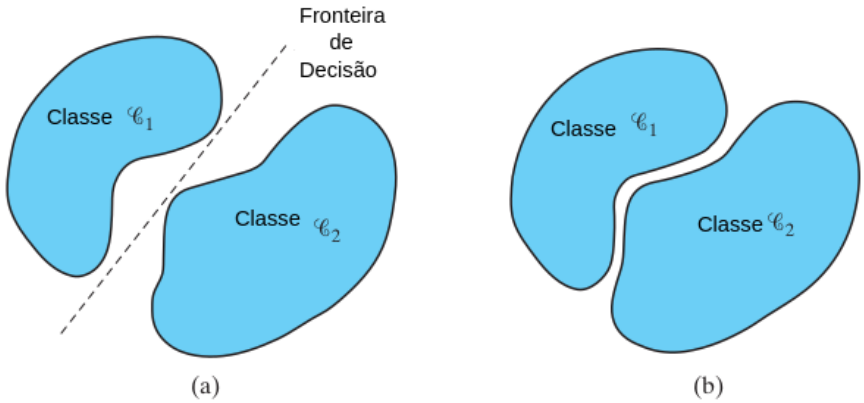
Em que $y$ representa a saída esperada ou o alvo para o exemplo atual, $\hat{y}$ a saída gerada pela rede e $\eta$ é a constante positiva denominada taxa de aprendizagem, cuja função é moderar o grau em que os pesos são alterados em cada etapa da iteração.

De acordo com Haykin (2009), na forma mais simples do Perceptron existe uma fronteira de decisão. Na Figura 6, onde é apresentado um exemplo bidimensional para dois atributos, são apresentadas duas classes. Para que o Perceptron funcione corretamente, $C1$ e $C2$ devem ser linearmente separáveis, como representado na Figura 6(a), onde a fronteira de decisão corresponde a um hiperplano. Contudo, quando as duas classes ou alvos tornam-se muito próximos uma da outra, como na Figura 6(b), elas se tornam não linearmente separáveis.

A regra de treinamento do Perceptron, conforme representada pelas Equações 2.5 e 2.4, converge para exemplos linearmente separáveis, contudo é limitado para problemas que não são, sendo difícil delimitar um hiperplano entre as classes.

De acordo com Goodfellow, Bengio e Courville (2016a), um exemplo comum, que o Perceptron é limitado para solucionar é o problema do “ou exclusivo”, ou XOR que consiste

Figura 6 – Classificação de padrões com duas classes




Fonte: Adaptada de Haykin (2009).

na operação sobre dois valores binários: 0 e 1. A operação retorna 1 quando esses valores são diferentes, por exemplo (0,1) e (1,0). Quando as entradas são iguais, para (0,0) e (1,1), por exemplo, é retornado 0. De acordo com Haykin (2009), essa operação é não linearmente separável porque não é possível traçar apenas uma linha reta como na Figura 6(a).

### 2.2.2 Multilayer Perceptron

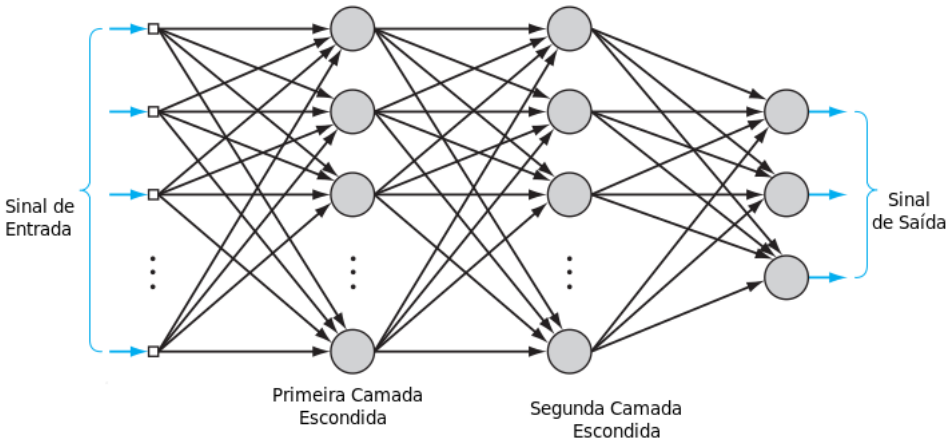
A Multilayer Perceptron (MLP) é uma variante do modelo Perceptron original proposto por Rosenblatt (1958), também conhecida como uma rede *feedforward*. Apresenta uma ou mais camadas escondidas entre a entrada e saída, com um número arbitrário de neurônios. De acordo com Haykin (2009), os sinais nessa rede são transmitidos em uma direção: entrada para saída. Diferentemente do Perceptron, que apresenta um único neurônio de saída, a MLP pode apresentar mais de um neurônio.

Conforme Bishop (2006), as etapas de processamento da rede neural se assemelha ao Perceptron, contudo, uma diferença fundamental com relação a essa, é que a MLP utiliza não-linearidades sigmoidais, por exemplo, nas unidades ocultas. De acordo com Gardner e Dorling (1998), o perceptron multicamadas consiste em um sistema de neurônios simples interligados, onde tem-se um mapeamento não linear entre um vetor de entrada e um vetor de saída. Podemos relacionar o número de neurônios em uma única camada oculta de um MLP como a quantidade de hiperplanos que cortam o espaço de entrada antes de gerar a classificação final (MELLO; PONTI, 2018)

A arquitetura da Multilayer Perceptron é completamente conectada ou *fully connected*, pois um neurônio em qualquer camada está conectado com todos os neurônios da camada seguinte, como exemplificado na Figura 7, que apresenta duas camadas escondidas.

O principal problema em construir o Perceptron com multicamadas é que se desconhece como estender a regra de aprendizagem do Perceptron para trabalhar com várias camadas em

Figura 7 – Arquitetura Multilayer Perceptron




Fonte: Adaptada de Haykin (2009).

um único passo(SKANSI, 2018). Para isso, utiliza-se outros algoritmos como a Regra Delta e o Backpropagation ou retropropagação, por exemplo, cujo ponto principal é a atualização dos pesos por meio do gradiente descendente, abordado na seção seguinte.

### 2.2.3 Gradiente Descendente

O treinamento de uma rede neural envolve o conceito de otimização, que consiste em minimizar ou maximizar uma função $f(x)$ por meio da modificação do valor de $x$ (GOODFELLOW; BENGIO; COURVILLE, 2016a). Essa função é chamada de função objetivo e sua derivada é importante para minimizá-la, pois indica como modificar $x$ para obter uma pequena melhoria em $y$. Dessa forma, pode-se minimizar $f(x)$ movendo $x$ em pequenos passos com o sinal oposto da derivada, utilizando a técnica conhecida como gradiente descendente (CAUCHY, 1847).

Considerando que a definição do erro seja fornecida pelo erro médio quadrático, representada pela Equação 2.6; na qual, segundo Mitchell (1997), $D$ é o conjunto de exemplos do treinamento, $y_d$ é o alvo para o exemplo $d$ e $\hat{y}_d$ a saída prevista pela rede; o gradiente descendente é utilizado para ajustar os pesos $w$ iterativamente, com o objetivo de minimizar o erro da função.

$$E(w) = \frac{1}{2} \sum_{d \in D} (y_d - \hat{y}_d)^2 \tag{2.6}$$

Sendo assim, esse erro pode ser minimizado calculando a derivada de $E$ em relação a cada componente do vetor $w$ e $b$. Conforme Mitchell (1997), o gradiente de $E$ com relação $w$, matematicamente representado por $\bigtriangledown E(w)$, pode ser descrito como:

$$-\bigtriangledown E(w) = \left[ \frac{\partial E}{\partial w_0}, \quad \frac{\partial E}{\partial w_1}, \quad \ldots, \quad \frac{\partial E}{\partial w_n}, \quad \frac{\partial E}{\partial b_0}, \quad \frac{\partial E}{\partial b_1}, \quad \ldots \quad \frac{\partial E}{\partial b_n} \right] \tag{2.7}$$

O gradiente de $E$ tem como componentes as derivadas parciais, conforme mostrado na Equação 2.7. Existem três variantes do gradiente descendente que diferem na quantidade de dados utilizados para calcular o gradiente da função objetivo (RUDER, 2016). As três categorias são: Batch Gradient descent, Stochastic Gradient descent (Stochastic Gradient descent (SGD)) e Mini-batch Gradient descent.

De acordo com Santra, Hsieh e Lin (2021), o Batch Gradient descent calcula o $\bigtriangledown E$ para cada iteração e atualiza os parâmetros dos pesos. O gradiente por *batch* é representado pela Equação 2.8, que otimiza iterativamente as variáveis do modelo, a partir do conjunto de dados $D$. Conforme Zhang (2019), o termo $w^d$ denota o vetor atualizado através da iteração $d$.

$$w^d = w^{d-1} - \eta \bigtriangledown E(w^{(d-1)}; D) \tag{2.8}$$

Como é necessário calcular o gradiente para todo o conjunto de dados para realizar apenas uma atualização, o gradiente por *batch* pode ser muito lento e intratável para conjuntos de dados que não cabem na memória (RUDER, 2016). Em vez de utilizar o conjunto de treino completo, o algoritmo SGD atualiza as variáveis do modelo através do cálculo do gradiente da função de perda instância por instância (ZHANG, 2019).

Portanto, o SGD permite calcular o gradiente da função de custo com base em um único exemplo selecionado para a iteração, conforme explicado por Becher e Ponti (2021). Para atualizar os parâmetros, a Equação 2.9 é utilizada, onde cada instância $(x_i, y_i)$ pertence ao conjunto de dados $D$ e são processadas uma por vez.

$$w^d = w^{d-1} - \eta \bigtriangledown E(w^{(d-1)}; (x_i, y_i)) \tag{2.9}$$

O Mini-batch Gradient descent é uma combinação dos métodos SGD e Gradient Descent (GD), segundo Santra, Hsieh e Lin (2021). Nele, o conjunto de dados é dividido em diversos mini-lotes e cada mini-lote é utilizado para calcular o gradiente do erro e, assim, atualizar os parâmetros. A Equação 2.10 representa o *mini-batch* das instâncias de treinamento a partir do conjunto $D$, onde $B$ é o tamanho do mini-lote utilizado.

$$w^d = w^{d-1} - \eta E(w^{d-1}; B) \tag{2.10}$$

### 2.2.4 Backpropagation

Na rede neural de uma única camada, o processo de treinamento é relativamente simples, pois o erro (ou função de perda) pode ser calculado diretamente em função dos pesos, permitindo um cálculo fácil do gradiente (AGGARWAL, 2018). Nas redes *feedforward*, segundo Ramchoun *et al.* (2016), o algoritmo mais utilizado é o Backpropagation combinado com as técnicas de gradiente descendente.

O algoritmo de Backpropagation foi proposto por Rumelhart, Hinton e Williams (1986) e, segundo Nielsen (2018), possibilitou o uso de redes neurais para resolver problemas anteriormente considerados insolúveis. A partir de uma função de perda, como a Equação 2.6, por exemplo, o algoritmo procura atualizar os pesos e minimizar o erro, utilizando a derivada do gradiente descendente. O objetivo é aprender como o erro varia à medida que os pesos são alterados, tornando necessária a utilização da regra da cadeia do cálculo.

No centro do Backpropagation tem-se um cálculo para a derivada parcial usando $\frac{\partial E}{\partial w}$ da função de custo $E$ em relação a qualquer peso $w$ ( ou *bias* $b$) na rede (NIELSEN, 2018). Segundo Moons, Bankman e Verhelst (2019), o Backpropagation, aplica a regra da cadeia, seguindo uma ordem específica de operações como demonstrada pela Equação 2.11.

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \varphi_j} \frac{\partial \varphi_j}{\partial v_j} \frac{\partial v_j}{\partial w_{ij}} \tag{2.11}$$

Conforme a Equação 2.11, são calculadas as derivadas parciais da função de perda com relação aos pesos $w_{ij}$. Conforme Moons, Bankman e Verhelst (2019), $\varphi_j$ corresponde a uma função de ativação qualquer que é aplicada e $v_j$ refere-se a combinação linear computada durante o treinamento.

### 2.2.5 Funções de Ativação

O principal objetivo de qualquer rede neural é transformar os dados de entrada não linearmente separáveis em características abstratas linearmente separáveis utilizando hierarquia de camadas (DUBEY; SINGH; CHAUDHURI, 2022). Essas camadas são combinações de funções lineares e não lineares. Segundo Dubey, Singh e Chaudhuri (2022), as camadas mais comuns de não-linearidade são as funções de ativação, como: sigmoide, tangente hiperbólica, Rectified Linear Unit (ReLU), Leaky ReLU e Softmax.

Ao longo desta seção serão apresentados alguns tipos de funções de ativação, são elas que permitem a transformação não linear nos dados de entrada. Sem essa não linearidade, a rede não conseguiria resolver problemas complexos.

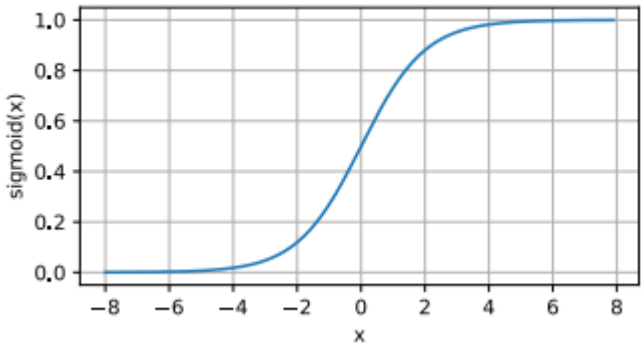
**Função Sigmoide Logística**: as funções sigmoides, cujos gráficos são curvas em "S", aparecem em uma grande variedade de contextos, tais como as funções de transferência utilizadas em muitas redes neurais (MENON *et al.*, 1996). Segundo Sellat, Bisoy e Priyadarshini (2022), essa função é amplamente utilizada, tendo aplicação principal para classificação binária, mapeando a soma ponderada das entradas para um intervalo entre $[0, 1]$.

Segundo Dubey, Singh e Chaudhuri (2021), a saída da função Sigmoide é saturada para entradas mais altas e mais baixas, matematicamente é expressa pela Equação 2.12 e representada

graficamente pela Figura 8.

$$\sigma(x) = \frac{1}{1+exp(-x))} \tag{2.12}$$

Figura 8 – Representação Gráfica da função Sigmoide




Fonte: Zhang *et al.* (2021).

**Função Tangente Hiperbólica**: de acordo com LeCun, Bengio e Hinton (2015), a função Tangente Hiperbólica conhecida como Tanh, é uma função mais suave, cujo centro é zero e apresenta intervalo que varia entre -1 a 1. A Equação 2.13 corresponde a essa função, que segundo Olgac e Karlik (2011), é definida como a razão entre as funções hiperbólicas seno e cosseno.

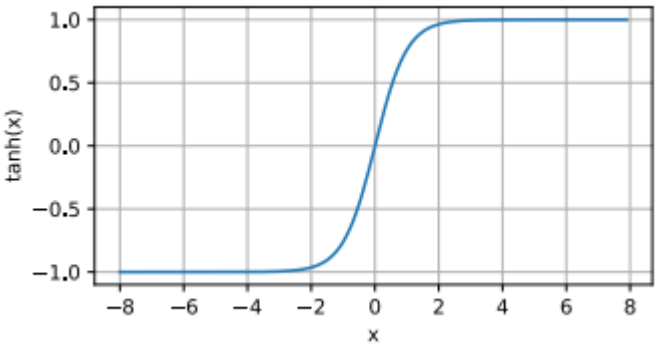
$$tanh(x) = \frac{1-exp(-2x)}{1+exp(-2x)} \tag{2.13}$$

O comportamento gráfico dessa função assemelha-se à Sigmoide, contudo é possível observar na Figura 9, de acordo com Zhang *et al.* (2021), que a Tanh apresenta uma simetria com relação a origem, observa-se também, que à medida que se aproxima de 0, a função se assemelha a uma transformação linear.

**Função ReLU**: a função ReLU (NAIR; HINTON, 2010), segundo Urban, Basalla e Smagt (2017), é linear para entradas positivas e zero para entradas negativas. Essa função de ativação, de acordo Krizhevsky, Sutskever e Hinton (2012), apresenta desempenho melhor que a Tanh, com aprendizado mais rápido. Conforme Nwankpa *et al.* (2018), a ReLU é representada pela Equação 2.14, realizando uma operação de limiar para cada elemento da entrada.

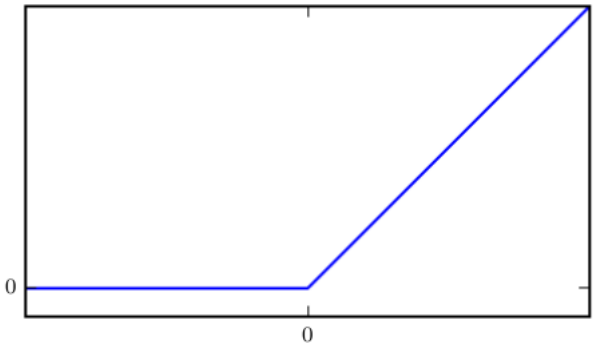
$$f(x) = max(0,x) = \begin{cases} x_i, & x_i \geq 0 \\ 0, & x_i < 0 \end{cases} \tag{2.14}$$

Figura 9 – Representação Gráfica da função Tangente




Fonte: Zhang *et al.* (2021).

Figura 10 – Representação Gráfica da função ReLU




Fonte: Goodfellow, Bengio e Courville (2016a).

Conforme Zhang *et al.* (2021), diferentemente das demais funções de ativação, todos os neurônios não são ativados simultaneamente. A entrada negativa, por exemplo, será convertida em zero e portando o neurônio não é ativado, como pode ser visualizado na Figura 10.

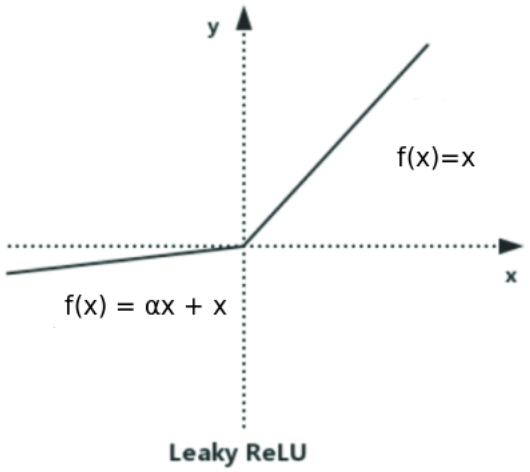
Segundo Goodfellow, Bengio e Courville (2016a), essa função é por padrão recomendada para utilização na maioria das redes neurais. Ela, assim como as citadas anteriormente, transforma entradas lineares em não lineares.

**Função Leaky ReLU**: a função Leaky ReLU similar a ReLU, tem como objetivo segundo Maas (2013), solucionar os problemas dos neurônios inativos da ReLU, de tal forma que os gradientes não sejam zeros. Conforme Dubey, Singh e Chaudhuri (2022), a função de ativação calcula o gradiente com um valor constante muito pequeno para o gradiente negativo, com $\alpha$ de 0.01. Essa função é representada pela Equação 2.15.

$$f(x) = \alpha x + x = \begin{cases} x, & x > 0 \\ \alpha x, & x \leq 0 \end{cases} \tag{2.15}$$

De acordo com Maas (2013), a Leaky ReLU tem um resultado idêntico quando compa-

Figura 11 – Representação Gráfica da função Leaky ReLU




Fonte: Adaptada de Xu *et al.* (2020).

rada com a ReLU padrão, com exceção dos gradientes não nulos, apresentando portanto, uma melhoria em esparsidade e dispersão, quando comparada com as funções padrão ReLU e Tanh. Segundo Xu *et al.* (2020), a saída da função tem um pequeno declive para a entrada negativa, como pode ser observado na Figura 11.

**Função Softmax**: a função Softmax é outro tipo de função de ativação utilizada na computação neural, responsável por calcular a distribuição de probabilidade a partir de um vetor de números reais (DUBEY; SINGH; CHAUDHURI, 2022). Segundo Sharma, Sharma e Athaiya (2017), a função Softmax é a combinação de múltiplas funções sigmoides, porém ao contrário dessa, pode ser utilizada para problemas de classificação multiclasse, atribuindo probabilidades.

A função Softmax é outro tipo de função de ativação amplamente utilizada na computação neural. Ela é responsável por calcular a distribuição de probabilidade a partir de um vetor de números reais (DUBEY; SINGH; CHAUDHURI, 2022). Ao contrário da função Sigmoide, que é comumente usada para problemas de classificação binária, atribuindo uma probabilidade à classe positiva e à classe negativa (como 1-p) (DUBEY; SINGH; CHAUDHURI, 2022). De acordo com Sharma, Sharma e Athaiya (2017), a função Softmax pode ser vista como uma extensão da função Sigmoide, combinando múltiplas funções sigmoides, mas sua característica principal é que ela atribui probabilidades para todas as classes, garantindo que a soma das probabilidades seja igual a 1 (NWANKPA *et al.*, 2018).

De acordo com Goodfellow, Bengio e Courville (2016a), pode ser expressa pela Equação 2.16.

$$f(v_i) = \frac{exp(v_i)}{\sum_j exp(v_j)} (i = 1, 2, ..., N). \tag{2.16}$$

Onde $v_1, v_2, ..., v_N$ são os valores de ativação dos neurônios da camada Softmax e o

valor $f(v_i)$ representa a probabilidade que a amostra pertence a i-ésima categoria (WANG *et al.*, 2018).

### 2.2.6 Funções de Custo

As funções de custo ou perda fazem parte da configuração primária no aprendizado de redes neurais. Essas funções são responsáveis por medir quão bem a rede ou modelo prevê suas saídas, uma vez que o objetivo é encontrar um conjunto de pesos que minimizem esse custo. Nesta seção, serão apresentadas as principais funções de perda para o aprendizado de máquina, o erro médio quadrático e a entropia cruzada.

**Mean Squared Error**: uma forma de medir o desempenho da rede neural é calcular o erro médio quadrático ou *mean squared error*. Considerando que $\hat{y}^{(teste)}$ sejam as previsões realizadas por um modelo no conjunto de teste $m$, por exemplo, então o cálculo para esse erro é dado por:

$$MSE_{teste} = \frac{1}{m}\sum_i (\hat{y}^{(teste)} - y^{(teste)})_i^2, \tag{2.17}$$

onde $y^{teste}$ corresponde a classe ou alvo esperado. Conforme Goodfellow, Bengio e Courville (2016a), para que um algoritmo de aprendizagem de máquina funcione como esperado, faz-se necessário melhorar os pesos de forma a reduzir o $MSE_{teste}$.

**Entropia Cruzada**: a partir da função Softmax, obtém-se a distribuição de probabilidade estimada, ou seja, $\hat{y}$, para cada entrada $x$ (ZHANG *et al.*, 2021). No aprendizado de máquina é importante verificar para um conjunto de dados, quão próxima é a saída prevista da saída correta. Sendo assim, a máxima verossimilhança escolhe os parâmetros que maximizam a probabilidade logarítimica das classes verdadeiras $y$, dados os valores observados $x$ (JURAFSKY; MARTIN, 2009).

Supondo que o conjunto de dados em que $x$ representa os dados e $y$ as classes, seja contido por $n$ exemplos; sendo $x_i$ um vetor de características e $y_i$ um vetor de rótulos, é possível comparar as estimativas com a realidade verificando quão provável as classes reais são de acordo com o modelo (ZHANG *et al.*, 2021). Sendo assim tem-se:

$$P(y|x) = \prod_{i=1}^{n} P(y_i|x_i). \tag{2.18}$$

Deste modo, dada as probabilidades condicionais estimadas, maximiza-se $P(y|x)$, que é equivalente a minimizar a probabilidade do *log-likelihood* (logaritmo da verossimilhança) negativo, ou seja, minimizar a diferença existente entre o que foi estimado e a probabilidade real. Segundo Zhang *et al.* (2021), para qualquer par de rótulo $y$ e predição $\hat{y}$ sobre uma quantidade $n$

de classes, a função de custo é definida por:

$$l(y,\hat{y}) = -\sum_{j=1}^{n} y_j log\hat{y}_j. \tag{2.19}$$

Essa função é denominada Entropia Cruzada ou Cross-Entropy, na Equação 2.19, cada probabilidade da classe prevista $\hat{y}_j$, é comparada com a saída desejada da classe (ZHANG *et al.*, 2021). Dessa maneira, a perda é calculada penalizando a probabilidade com base em quão distante está do valor esperado.

## 2.3 Redes Neurais profundas

Com o avanço da das redes neurais surgiram arquiteturas que utilizam um número maior de camadas e neurônios durante o treinamento. Sendo assim, nesta Seção são apresentadas algumas das principais arquiteturas para Aprendizado Profundo como as Redes Neurais Convolucionais, o Autoencoder, redes neurais Recorrentes, o Transformer para processamento de língua natural e o Vision Transformer que consiste em uma adaptação utilizando Transformer para visão computacional.

### 2.3.1 Redes Neurais Convolucionais

A Rede Neural Convolucional também conhecida como ConvNet (do inglês, Convolutional Neural Network), é um algoritmo de Aprendizado Profundo capaz de atribuir pesos e *bias* a filtros convolucionais que permite associar padrões a objetos ou aspectos em uma imagem, por exemplo, o espectrograma que é uma representação visual do áudio apresentada na Seção **??**. Ao contrário de uma rede Multilayer Perceptron, a ConvNet é capaz de capturar dependências espaciais de uma imagem por meio deeses filtros.

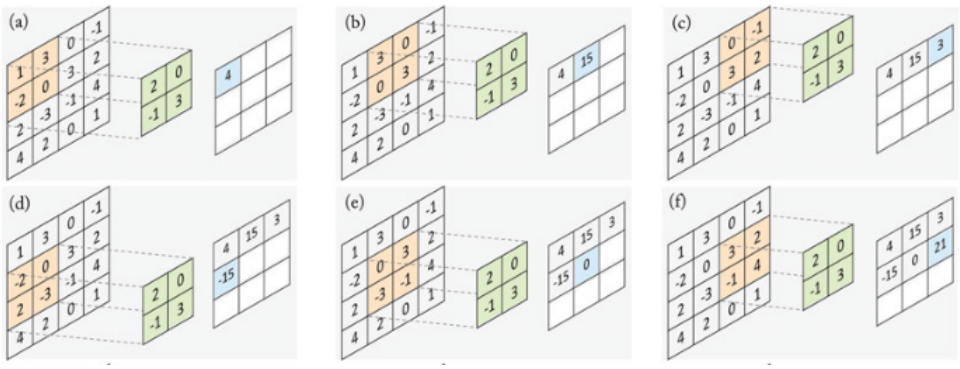
Em seu formato usado atualmente, a ConvNet foi proposta por Lecun e Bengio (1995), e em seu trabalho seminal Lecun, Bengio e Hinton (2015) utilizam dois tipos de camadas para realizar o aprendizado das características espaciais: camadas convolutivas e camadas de *pooling*. Nesta seção, serão abordadas essas camadas principais e o funcionamento desta rede.

As camadas convolucionais são os componentes mais importantes da rede neural convolucional. Conforme Khan *et al.* (2018), essas camadas compõem um conjunto de filtros também denominados *kernels* convolucionais. Esses *kernels* são utilizados na operação de convolução com uma determinada entrada para gerar mapas de características a partir de campos receptivos locais (PONTI *et al.*, 2017).

A entrada dessa rede é um tensor em 3D que apresenta canais, largura e altura. Os filtros são um conjunto de pesos e a operação de convolução caracteriza-se por deslizar o filtro através da entrada. Segundo Khan *et al.* (2018), considerando um mapa de características de entrada 2D

de tamanho 4x4 e um filtro de tamanho 2x2, o filtro desliza ao longo da largura e altura desse mapa, conforme representado na Figura 12.

Figura 12 – Operação de Convolução



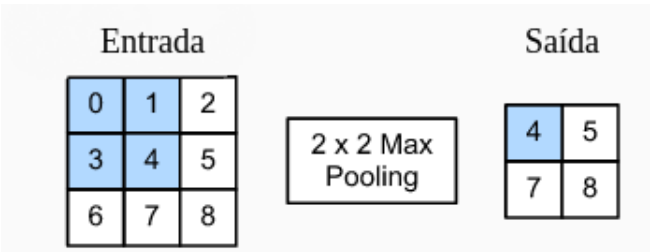
Fonte: Khan *et al.* (2018).

O filtro 2x2 da Figura 12 é multiplicado com a região de mesmo tamanho, representada pela cor laranja. Os valores resultantes dessa operação são somados para obter uma entrada correspondente no mapa de características de saída, em cada passo da convolução.

**Pooling**: As camadas de *pooling*, conforme O'Shea e Nash (2015), visam reduzir gradualmente a dimensão da representação e assim, reduzir ainda mais o número de parâmetros e a complexidade computacional do modelo. No contexto de processamento de imagens, pode ser considerado como semelhante à redução da resolução. Segundo Albawi, Mohammed e Al-Zawi (2017), o *maxpooling* é um dos métodos mais comuns de *pooling*, essa técnica divide a imagem em retângulos de sub-região, e apenas devolve o valor máximo dessa sub-região.

Como é possível observar na Figura 13, aplica-se um *maxpoling* $2 \times 2$, o filtro se move ao longo do mapa de características e armazena o maior valor em cada passo. De acordo com Venkatesan e Li (2017), o *maxpooling* é mais recomendável do que outros métodos, pois representa a resposta com melhor correspondência ao filtro, o que no contexto do processamento de imagem é o mais adequado.

Figura 13 – Operação de Pooling

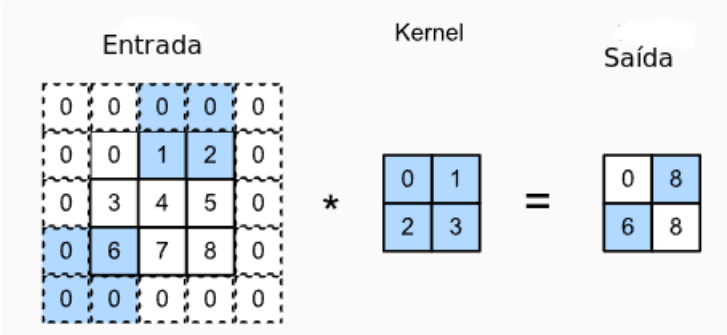



Fonte: Adaptada de Zhang *et al.* (2021).

**Padding e Stride**: Segundo Aggarwal (2018), a redução no tamanho da camada na operação convolutiva, não é desejável em geral, porque tende a perder alguma informação ao

longo da borda de características. Esse problema pode ser resolvido através do *padding* ou preenchimento, na Figura 14, por exemplo, utiliza-se o *zero-padding* que acrescenta *pixels* com valor zero, em torno dos limites do mapa de características.

Figura 14 – Padding e stride




Fonte: Adaptada de Zhang *et al.* (2021).

A operação de convolução é aplicada em diferentes regiões da imagem, a cada aplicação dessa operação a região é alterada pelo *stride*. Na Figura 14, por exemplo, o *stride* é 3x2, isso significa que o *kernel* 2x2 é deslocado três colunas verticalmente e duas colunas horizontalmente.

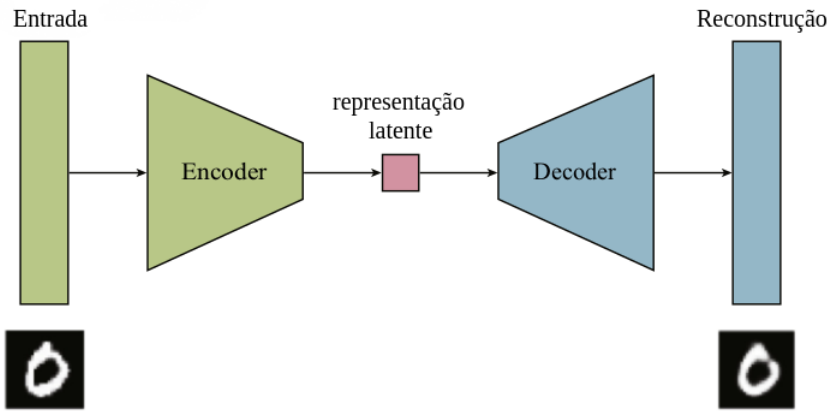
### 2.3.2 Autoencoder

O Autoencoder é uma rede neural que faz parte dos métodos de representação capazes de aprender características a partir de dados não etiquetados tipicamente por meio da tarefa de reconstrução Cavallari, Ribeiro e Ponti (2018). Métodos como esse, são concebidos para mapear os dados de entrada em um espaço latente restrito e depois utilizá-lo para produzir uma saída semelhante à entrada. Por isso o Autoencoder consiste em duas partes: um *encoder* e um *decoder* como representado na Figura 15, cada um com o seu próprio conjunto de parâmetros de aprendizagem.

Segundo Lopez Pinaya *et al.* (2020), a função do *encoder* é receber uma entrada $x$ e mapeá-la em um espaço latente, em que a entrada é representada por variáveis transformadas também chamadas de código. Por outro lado, o *decoder* recupera os dados codificados e aprende uma reconstrução da entrada. As aplicações de autoencoders abrangem desde dados não estruturados como imagens 2D ou 3D, vídeos e textos, e também em representação numérica. É possível utilizá-lo ainda na tarefa de fusão de atributos não estruturados e estruturados conforme o trabalho de Resende e Ponti (2022).

Analisando com uma abordagem matemática, de acordo com Goodfellow, Bengio e Courville (2016b), a rede possui uma camada oculta $h$ que descreve um código ou representação latente com base na entrada. O *encoder*, abrange a função codificadora $h = f(x)$ e o *decoder* que produz uma reconstrução $r = g(h)$ do que foi codificado. Conforme o autor ainda, esse modelo foi concebido para ser incapaz de aprender a copiar com perfeição. Para isso, são adicionadas

Figura 15 – Estrutura do Autoencoder




Fonte: Adaptada de Lopez Pinaya *et al.* (2020).

restrições na tarefa de cópia, de maneira tal que a representação latente $h$ assuma propriedades úteis dos dados de entrada.

Uma das formas de obter características úteis do Autoencoder é limitar $h$ a ter uma dimensão inferior a $x$. Um Autoencoder cuja dimensão do código $h$ é inferior a dimensão da entrada é denominado Undercomplete, conforme Goodfellow, Bengio e Courville (2016b), esse tipo de rede captura as características mais salientes dos dados de formação. O processo de aprendizagem consiste na minimização da função de perda, que pode ser representada pela Equação 2.20, em que $L$ é a função de custo, que penaliza $g(f(x))$ por ser diferente da entrada $x$.

$$L(x, g(f(x))) \tag{2.20}$$

De acordo com Bank, Koenigstein e Giryes (2020), no Undercomplete, é imposto um gargalo ou *bottleneck*, que refere-se a camada do código. Essa opção de gargalo também serve diretamente para se obter uma representação baixa da dimensão dos dados, para fins de compressão dos dados ou extração de características, por exemplo.

Outro tipo de Autoencoder é o Overcomplete, quando $h$ possui dimensões maiores que a entrada, ou seja, possui uma camada intermediária com alta dimensionalidade. Uma maneira de impedir a cópia dos dados, conforme Bank, Koenigstein e Giryes (2020), é aplicar a regularização, destacando-se dois tipos: Sparse Autoencoder e Denoising autoencoder.

**Sparse Autoencoder**: No Sparse Autoencoder, as restrições são adicionadas na camada oculta. Conforme Kumar, Nandi e Kala (2014), são aplicadas uma penalidade de esparsidade com o objetivo de responder a recursos estatísticos exclusivos do conjunto de dados, em vez de simplesmente se comportar como uma função identidade. Aplica-se normalmente a regularização L1, essa regularização segundo Goodfellow, Bengio e Courville (2016b), adiciona à função de custo ($L$) um termo ($\lambda$) multiplicado pelo valor absoluto da camada de código $h_i$ como descrito

na Equação 2.21. Essa regularização possibilita à seleção de características mais importantes, o que resulta em uma solução mais esparsa.

$$L(x, g(f(x))) + \lambda \sum_i |h_i| \tag{2.21}$$

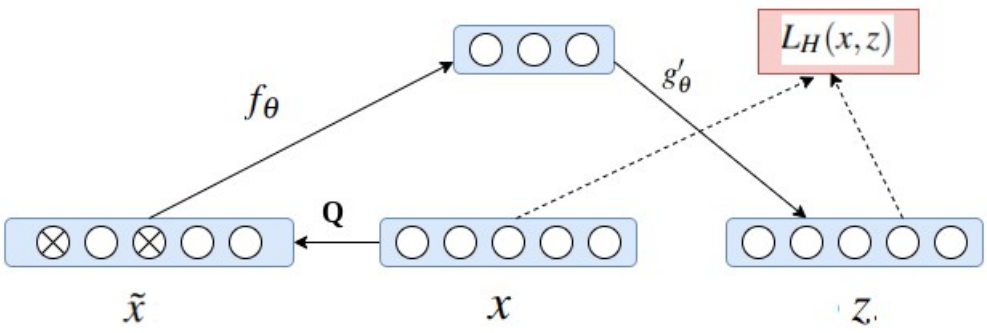
**Denoising Autoencoder**: O Denoising Autoencoder foi proposto com uma estratégia diferente das citadas anteriormente. Segundo Vincent *et al.* (2010), em vez de limitar a representação com restrições, foi proposta a alteração do critério de reconstrução para a aplicação das limpezas das entradas parcialmente ruidosas, $\tilde{x}$. Sendo assim, são adicionados ruídos às entradas e o trabalho do Denoising Autoencoder, portanto, é a reconstrução de uma entrada limpa a partir de dados ruidosos.

Portanto, essa versão ruidosa, $\tilde{x}$, é mapeada para uma representação apresentada pela Equação 2.22, em que $\varphi$ consiste na função de ativação e $\theta$ os parâmetros $W$ e $b$, em que $W$ é a matriz de pesos e $b$ o *bias*.

$$h = f_\theta(\tilde{x}) = \varphi(W\tilde{x} + b) \tag{2.22}$$

A partir da Equação 2.22, é reconstruída $z = g_\theta(h)' = \varphi(W'h + b')$. Os parâmetros $\theta$ e $\theta'$ são treinados para minimizar o erro médio de reconstrução sobre o conjunto de treinamento, ou seja, ter $z$ o mais próximo possível da entrada sem ruído (VINCENT *et al.*, 2010). A Figura 16, apresenta a arquitetura do Denoising Autoencoder, em que $\tilde{x}$ representa a entrada corrompida por $Q$, que consiste no mapeamento estocástico, da entrada $x$ para $\tilde{x}$. O *encoder* corresponde a $f_\theta$ e o *decoder* $g'_\theta$ que realiza a reconstrução de $x$, obtendo como resultado $z$. O erro da reconstrução é calculado por meio da função de custo $L_H(x, z)$.

Figura 16 – Arquitetura do Denoising Autoencoder




Fonte: Adaptada de Vincent *et al.* (2010).

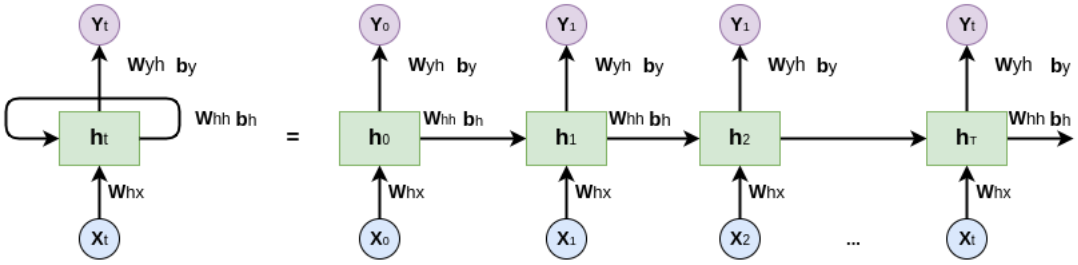
### 2.3.3 Redes Neurais Recorrentes

Uma das principais características das redes densamente conectadas, segundo Chollet (2017), é que elas não possuem capacidade de memorização do que foi aprendido. Cada entrada que lhes é mostrada é processada independentemente, não contendo um estado que seja mantido entre as entradas (CHOLLET, 2017).

Contudo, certos tipos de dados tais como séries temporais, texto e dados biológicos, contém dependências sequenciais entre os atributos (AGGARWAL, 2018), sendo assim, importante o uso de redes que sejam capazes de memorizar decisões passadas, como é o caso das RNNs (do inglês, Recurrent Neural Networks). De acordo com Sutskever, Martens e Hinton (2011), as RNNs são compostas por estados ocultos que funcionam de forma não linear. A estrutura desses estados, por sua vez, atuam como a memória da rede e o estado da camada oculta está condicionado ao seu estado anterior (MIKOLOV *et al.*, 2014). Essa estrutura permite as RNNs, armazenar, lembrar e processar sinais complexos referentes ao passado, por longos períodos de tempo (SALEHINEJAD *et al.*, 2017). Em síntese, essas redes neurais conseguem mapear uma sequência de entrada, como um sinal de áudio, para uma sequência de saída no período de tempo atual e prever a próxima sequência no período de tempo seguinte.

A RNN, conforme Haykin e Network (2004), é formada por neurônios artificiais com um ou mais laços de feedback, esses laços são ciclos recorrentes ao longo do tempo ou sequência (MIKOLOV *et al.*, 2010). Na Figura 17, é representada a rede neural recorrente clássica, na sua forma dobrada, em que todas as operações são tratadas de forma simultânea e compacta e na sua forma desdobrada, onde é apresentado explicitadamente as iterações para cada passo de tempo. Os estados das camadas ocultas, $h_0$ a $h_t$, estão conectadas ao longo do tempo $t$, por meio de ligações recorrentes(SALEHINEJAD *et al.*, 2017). As entradas consistem em $x_0, x_1, ..., x_t$.

Figura 17 – Rede Neural Recorrente




Fonte: Adaptada de Olah (2015).

De acordo com Sutskever, Martens e Hinton (2011), a rede calcula essa sequência de estados ocultos e uma sequência de saídas $y_0, ...y_t$ por meio das Equações 2.23 e 2.24. É importante destacar que as equações representadas utilizam a Função Tanh, contudo outras

funções de ativação são utilizadas, como a Sigmoide, por exemplo.

$$h_t = tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{2.23}$$

$$y_t = W_{yh}h_t + b_y \tag{2.24}$$

A Equação 2.23, é composta por $W_{h_x}$, a matriz dos pesos entre a entrada $x_t$ e a camada oculta. Segundo Lipton, Berkowitz e Elkan (2015), a matriz de pesos recorrentes refere-se a $W_{hh}$, entre a camada oculta e ela própria, $h_{t-1}$, que se refere ao estado anterior da rede e $b_h$ o *bias*. A função de ativação Tanh é aplicada aos elementos do vetor devolvendo valores entre -1 e 1. Na Equação 2.24, $y_t$ corresponde a saída da rede, cuja matriz de pesos é $W_{y_h}$ e encontra-se entre a camada oculta e a saída. Portanto, $h_t$ como representado na Equação 2.23, é o vetor da camada oculta gerado no tempo $t$.

É importante ressaltar que para redes destinadas à regressão, $y_t$ pode ser linear. No entanto em problemas de classificação, $y_t$ é não linear. Além disso, existem variações nas arquiteturas de redes neurais recorrentes que podem apresentar mais de uma camada recorrente e *feedforward*.

De acordo com Sutskever, Martens e Hinton (2011), a rede calcula uma sequência de estados ocultos $(h_0, ..., h_t)$ e uma sequência de saídas $(y_0, ..., y_t)$ através da aplicação das Equações 2.23 e 2.24. As redes neurais recorrentes não funcionam acrescentando novas camadas a uma rede *feedforward*, mas adicionando ligações recorrentes na camada oculta (SKANSI, 2018).

As redes neurais recorrentes clássicas contudo, apresentam problemas no desaparecimento do gradiente durante a retropropagação ou a explosão para valores maiores, conforme Aggarwal (2018). Esse tipo de instabilidade é o resultado direto da multiplicação sucessiva com a matriz de peso recorrente (AGGARWAL, 2018). Logo, a rede apresenta problemas na aprendizagem de sequências de longo prazo, sendo portanto necessário o uso de redes com portões, das quais a LSTM é apenas um exemplo.

### 2.3.4 Redes Recorrentes LSTM

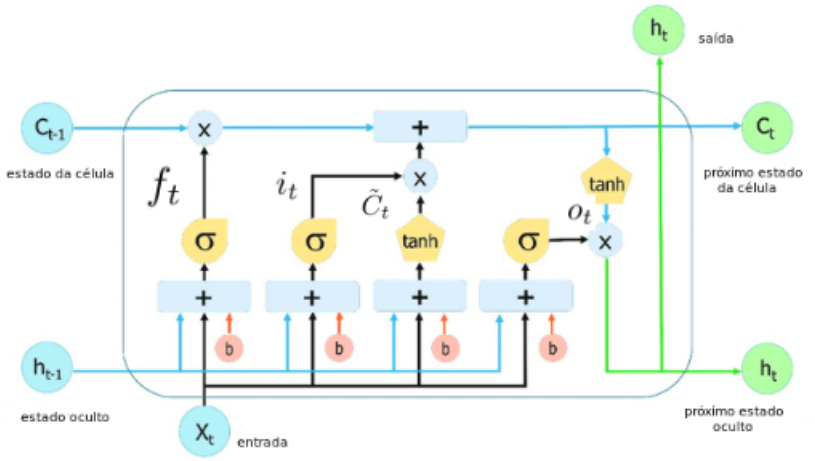
A rede LSTM (do inglês, Long Short-Term Memory), proposta por Schmidhuber, Hochreiter *et al.* (1997), é uma rede recorrente com memória de longo prazo e curto prazo, surgiu como uma adaptação da RNN tradicional. Conforme Greff *et al.* (2015), apresenta um modelo eficaz e expansível para problemas de aprendizagem relacionadas com dados sequenciais sendo amplamente utilizada para reconhecimento e geração de texto, tradução e reconhecimento de fala, por exemplo.

As redes LSTM dividem o problema de gestão de contexto em dois sub-problemas: retirar informação que já não é necessária no contexto e acrescentar informação suscetível de ser

necessária para posterior tomada de decisão (JURAFSKY; MARTIN, 2009).

Diferentemente da RNN clássica, a LSTM possui um módulo de repetição contendo quatro camadas, representada pela Figura 18. A arquitetura proposta por Gers e Schmidhuber (2000), apresenta o portão de esquecimento na célula. Segundo Yu *et al.* (2019), esse portão determina quais dos dados anteriores serão esquecidos e quais serão usados nas próximas etapas.

Figura 18 – Gates da rede LSTM




Fonte: Adaptada de Yan (2016).

A Figura 18 pode ser matematicamente expressa pelas Equações 2.25 a 2.30, em que $W_f$, $W_i$, $W_c$ e $W_o$ representam os pesos e $b_f$, $b_i$, $b_c$ e $b_o$ os *biases*. $W_f$ e $b_f$ representam o esquecimento, $W_i$ e $b_i$ compõem a entrada; $W_c$ e $b_c$ a memória e por fim, $W_o$ e $b_o$ a saída.

Na Equação 2.25, a porta de esquecimento $f_t$ , controlada pela função Sigmoide, decide quais informações da célula de memória anterior devem ser mantidas ou descartadas. Quando o valor $f_t$ está próximo de 1, as informações anteriores são preservadas, enquanto os valores próximos a 0 indicam que as informações antigas são esquecidas, como relatado por Yu *et al.* (2019).

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.25}$$

A próxima Equação 2.26, decide quais novas informações serão armazenadas no estado da célula $i$ - que representa um vetor de entrada do portão de ativação, de acordo com Arras *et al.* (2019).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.26}$$

Na Equação 2.27, uma camada *tanh* é usada para criar um vetor com novos valores

candidatos, $\tilde{C}$, que serão adicionados ao estado, como explicado por (MIKAMI, 2016).

$$\tilde{C}_t = tanh(W_c \cdot [h_{t-1}, xt + b_c]) \tag{2.27}$$

De acordo com Mikami (2016), na etapa de atualização da célula, é necessário multiplicar o vetor $f_t$ e adicioná-lo a $i_t * \tilde{C}_t$ para atualizar a célula referente ao estado anterior, $C_{t-1}$.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.28}$$

A Equação 2.29, descreve o portão de saída, $o_t$, que representa o que deve sair da célula de memória. Para isso, a função Sigmoide é aplicada novamente ao estado oculto anterior, $h_{t-1}$, e à entrada atual $x$, como explicado por Mikami (2016).

$$o_t = \sigma(W_o[h_{t-1}, x_t + b_o]) \tag{2.29}$$

Por fim, como descrito na Equação 2.30, o resultado obtido na Equação 2.29 é multiplicado com a função Tanh, que é aplicada à nova célula da memória. A partir das informações de saída, é possível prever qual será a próxima sequência.

$$h_t = o_t * tanh(C_t) \tag{2.30}$$

### 2.3.5 Redes Recorrentes GRU

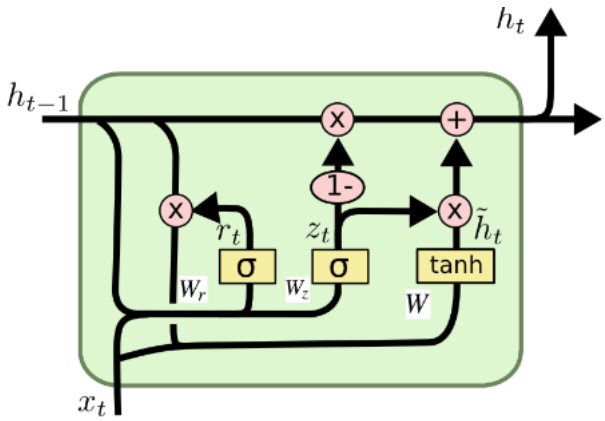
As redes LSTM introduzem um número considerável de parâmetros, tendo oito conjuntos de pesos a serem aprendidos para cada um dos quatro portões. A formação destes parâmetros adicionais impõe um custo elevado (JURAFSKY; MARTIN, 2009). Sendo assim, surge a rede GRU (do inglês, *Gated Recurrent Unit*), reduzindo o número de portões na arquitetura para dois, dispensando a utilização de um vetor de contexto separado. A arquitetura da GRU padrão pode ser descrita pelas Equações 2.31 a 2.34 e representada pela Figura 19.

$$r_t = \sigma(W_r * [h_{t-1}, x_t]) \tag{2.31}$$

$$z_t = \sigma(W_z * [h_{t-1}, x_t]) \tag{2.32}$$

De acordo com Wang *et al.* (2019), $r_t$ representa a saída do portão de reinicialização no momento $t$ e $z_t$ é o portão de atualização. $W_z$ é o peso entre a entrada e $h_{t-1}$ no portão de atualização, $W_r$ é o peso entre a entrada e o portão de reinicialização, neste caso, $h_{t-1}$ referente a Equação 2.31, é a unidade de saída da GRU padrão no tempo $t-1$.

Figura 19 – Gates da rede LSTM




Fonte: Adaptada de Olah (2015).

Na Equação 2.33, $\tilde{h}_t$ é o novo vetor com os valores candidatos criados com a camada *tanh* no tempo $t$, ou seja, o estado oculto que é posteriormente adicionado ao estado atual. $W$ consiste no peso entre a saída do portão de atualização e as entradas.

$$\tilde{h_t} = tanh(W * [r_t * h_{t-1}, x_t]) \tag{2.33}$$

Na Equação 2.34 informações redundantes são descartadas em $(1 - z_t * h_{t-1})$ e adicionadas ao produto $(z_t * \tilde{h_t})$, para formar o estado oculto $h_t$ (WANG *et al.*, 2019).

$$h_t = (1 - z_t * h_{t-1} + z_t * \tilde{h_t}) \tag{2.34}$$

De acordo com Jurafsky e Martin (2009), assim como a LSTM, a rede GRU utiliza uma função de ativação para bloquear ou permitir a passagem de informação. O objetivo do portão de reinicialização $r$ é decidir quais aspectos do estado oculto anterior são relevantes para o contexto atual e o que pode ser ignorado.

### 2.3.6 Mecanismo de Atenção

A atenção é uma função cognitiva complexa que é indispensável para os seres humanos (CORBETTA; SHULMAN, 2002). Nesta seção, será apresentado sobre o mecanismo de atenção do ponto de vista de aprendizado de máquina, a partir da qual podemos fazer analogias com essa capacidade cognitiva humana.

As redes neurais profundas utilizando os métodos descritos até a seção anterior apresentaram um importante desempenho em distintas tarefas de aprendizado de máquina. No entanto, uma das principais limitações dos métodos anteriores tem a ver com a capacidade de reter informação de curto e longo prazo em sequências como por exemplo sequências de texto (PONTI *et al.*, 2021). As abordagens Seq2seq (do inglês *sequence-to-sequence*), proposta inicialmente por

Sutskever, Vinyals e Le (2014), são responsáveis por mapear uma sequência para outra sequência. Surgiram primeiramente no contexto de processamento de língua natural e notadamente na resolução da tarefa de tradução automática. As primeiras arquiteturas, conforme Neubig (2017), empregam o uso do *encoder* e *decoder* e redes recorrentes. O *encoder* utiliza uma rede neural recorrente para codificar a sequência alvo em um vetor de contexto, também conhecido como *sentence embedding* (CASANOVA, 2022) e o *decoder*, recebe o vetor de contexto e o transforma na saída Weng (2018 apud CASANOVA, 2022).

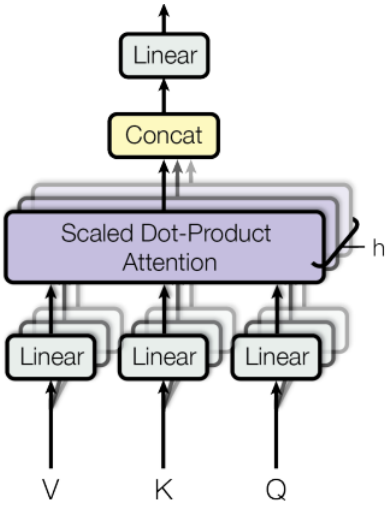
Contudo, o Seq2seq baseado em redes recorrentes apresenta desvantagens, dentre elas, segundo Jiang e Rijke (2018), está a falta de variabilidade dos modelos entre as respostas do sistema e o contexto contido no corpus original. A sequência de saída depende muito do contexto definido na camada oculta o que torna difícil para o modelo lidar com sentenças longas. Os mecanismos de atenção solucionam essas dificuldades, na tradução automática, por exemplo, com o trabalho concebido por Bahdanau, Cho e Bengio (2014). Com o mecanismo de atenção, ao invés da entrada ser comprimida, o decodificador (*decoder*) revisita a sequência de entrada em cada etapa, além disso, ele pode focalizar seletivamente em em partes específicas da entrada (ZHANG *et al.*, 2021). Isso é feito por meio do aprendizado de pesos específicos para cada *embedding* de entrada de forma que o modelo possa aprender a dar mais importância a entradas mais relevantes para a geração da próxima saída dado o contexto.

Em linhas gerais, a atenção permite enfatizar o que é relevante em detrimento do que é irrelevante. Em tarefas como a tradução automática, por exemplo, faz-se necessário atenção em palavras que sejam relevantes para uma tradução correta.

**Multi-head Attention**: A partir dos mecanismos de atenção o Transformer é capaz de capturar todos os relacionamentos entre os *tokens* de entrada e saída, sendo dominante no processamento de língua natural, conforme Zhang *et al.* (2021), no pré-treinamento de modelos. Segundo Vaswani *et al.* (2017), a função de atenção pode ser descrita como uma operação matemática que recebe três entradas: *query* (Q), *keys* (K) e *values* (V). Em Processamento de Língua Natual (PLN) essas entradas consistem em *embeddings*, que são representações latentes da entrada, que correspondem aos *tokens* de um dado texto. Essa operação realiza portanto, um mapeamento de uma consulta (*query*) e um conjunto de pares chave-valor (*key-value*) para uma saída. (VASWANI *et al.*, 2017). Segundo Hore e Chatterjee (2022), essas entradas são obtidas combinando os *embeddings* em uma matriz $X$, por exemplo, e multiplicando-a com as matrizes de peso $W_k$,$W_q$ e $W_v$.

Com base em Vaswani *et al.* (2017), foi proposta uma operação de atenção denominada Multi-head Attention. Em vez de ter apenas uma atenção com *keys*, *values* e *queries*, os autores propuseram projetar linearmente $h$ versões diferentes de *queries*, *keys* e *values* usando projeções lineares distintas. A função de atenção é então executada em paralelo para cada uma dessas versões projetadas, produzindo valores de saída independentes. Esses valores de saída são concatenados, como representado na Figura 20, e projetados para produzir os valores finais. A

Figura 20 – Multi-head Attention




Fonte: Vaswani *et al.* (2017).

Equação 2.35 é responsável por calcular a atenção para cada palavra, por exemplo. A função Scaled Dot-Product, calcula os produtos escalares da *query* com todas *keys*. Cada pontuação obtida é dividida por um fator escalar $\sqrt{d_k}$, em que $d_k$ representa a dimensão dos vetores da chave (VASWANI *et al.*, 2017).

Após as pontuações obtidas com a divisão pelo fator escalar ($\sqrt{d_k}$), é aplicada a função Softmax. Conforme Alammar (2018), essa pontuação determina o quanto cada palavra será expressa em uma determinada posição e por fim, ocorre a multiplicação com cada valor representado na equação por $V$. Ao final, as "cabeças de atenção" são concatenadas e multiplicadas por uma matriz de peso. )

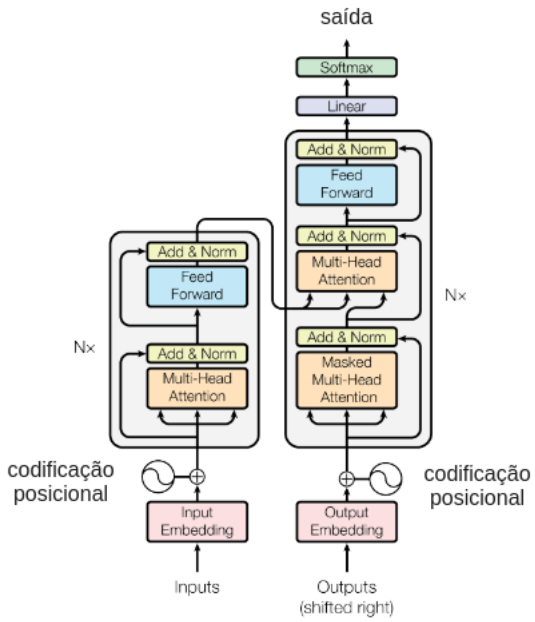
$$Attention(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.35}$$

### 2.3.7 Arquitetura do Transformer

Como explanado anteriormente, o Transformer, com o mecanismo de atenção, em processamento de língua natural consegue capturar relacionamentos entre os *tokens* de entrada e saída. Para que isso de fato aconteça, ele é formado por uma arquitetura composta de duas partes: *encoder* e *decoder*. De acordo com Vaswani *et al.* (2017), o *encoder* é composto por seis blocos idênticos, em que cada bloco contém duas camadas. A primeira subcamada apresenta o mecanismo Multi-head Attention e a segunda consiste em uma rede *feedforward*.

O *decoder* também apresenta camadas semelhantes ao *encoder* e apresenta uma camada adicional chamada Masked Multi-head Attention. Essa máscara garante que as previsões sejam com base nas saídas conhecidas, ou seja, cada posição dependa somente das posições anteriores. A razão para usar a versão mascarada de atenção Multi-head para os *embeddings* de saída é que

Figura 21 – Arquitetura do *encoder*




Fonte: Adaptada de Vaswani *et al.* (2017).

quando gerado o texto de saída, por exemplo, ainda não se tem as próximas palavras, tendo em vista, que elas ainda não foram geradas (GHOJOGH; GHODSI, 2020).

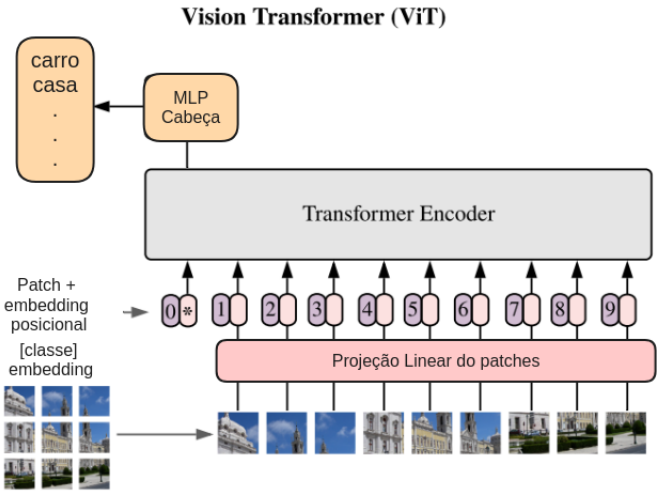
Na Figura 21, tem-se a arquitetura referente ao *encoder* do Transformer. Inicialmente os *inputs* são transformados em *embeddings* e, para incorporar informações de posições aos dados, é aplicado o *positional encoding* ao *embedding* de entrada (GHOJOGH; GHODSI, 2020). O *positional encoding* permite que o modelo aprenda relações entre as posições de cada *token*. A informação da posição tem um efeito importante na semântica da sequência, para isso Vaswani *et al.* (2017), propuseram o uso de funções senoidais.

Após o *positional encoding*, as entradas são passadas para o Multi-head Attention como explicitado anteriormente, aplicando o mecanismo de atenção $h$ vezes. Vaswani *et al.* (2017) anexaram uma conexão residual que corresponde a uma adição (*addition*) integrada com uma camada de normalização. A adição é inspirada nos blocos residuais propostos por He *et al.* (2016), que permitem que o gradiente flua em modelos com muitas camadas, adicionando a camada anterior a saída da camada atual. A camada de normalização (*layer normalization*) (BA; KIROS; HINTON, 2016), aplica uma normalização após cada camada. Portanto, a saída de cada subcamada é $CamadaNorm(x + Subcamada(x))$, onde $Subcamada(x)$ é a função implementada pela subcamada em si (VASWANI *et al.*, 2017). Por fim, após a adição e normalização citadas anteriormente, o resultado é passado para uma rede *feedforward*.

### 2.3.8 Vision Transformer

O Vision Transformer (ViT), representada pela Figura 22, é uma arquitetura proposta por Dosovitskiy *et al.* (2020a) que utiliza o *encoder* do modelo Transformer para visão computacional com grandes conjuntos de dados de imagens como o ImageNet (RUSSAKOVSKY *et al.*, 2015), CIFAR-100 (KRIZHEVSKY; HINTON *et al.*, 2009) e VTAB (ZHAI *et al.*, 1910). Inicialmente as imagens 2D são redimensionadas para uma sequência de *patches* 2D, $x_p \in \mathbb{R}^{Nx(P^2 \cdot C)}$, em que $P$ representa a resolução de cada *patch*, $C$ o número de canais e $N = HW/P^2$ o número de *patches*, em que H e W representam a resolução da imagem (DOSOVITSKIY *et al.*, 2020a).

Figura 22 – Visão Geral da arquitetura ViT




Fonte: Adaptada de Dosovitskiy *et al.* (2020a).

Cada um desses *patches* são transformados em *embeddings*, após a passagem pela camada densa e utilizados como entrada no *encoder* do Transformer (DOSOVITSKIY *et al.*, 2020a). Como mencionado na Seção 2.3.7, o *encoder* é composto por Multi-head Attention, camadas densas (MLP), camadas de normalização e conexões residuais, como apresentado na Figura 21. Assim como em modelos para processamento de língua natural, o ViT adiciona um *token* da classe no início da sequência de *patches*. Cada *patch* é tratado como um *token* que por sua vez, é linearizado para um sequência 1D. *Embeddings* de posição são adicionados, para codificar a posição de cada *patch* (DOSOVITSKIY *et al.*, 2020a). A representações da imagem obtidas com o modelo Transformer são utilizadas como entrada em uma MLP que é a "cabeça de classificação" com uma camada escondida. Essa "cabeça de classificação" é anexada à saída do último bloco do *encoder* do Transformer.

## 2.4 Modelos aplicados ao Processamento de áudio

Com o processamento de fala e os avanços na tarefa de reconhecimento automático da fala, síntese e classificação, modelos específicos para fala tem sido utilizados. Nesta Seção são

apresentadas dois modelos específicos, o Wav2vec 2.0 e o Audio Spectrogram Transformer que tem como base o Vision Transformer mencionado anteriormente.

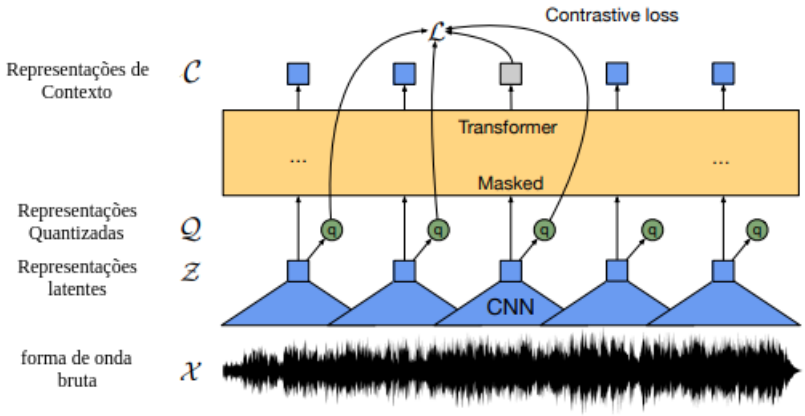
### 2.4.1 Wav2vec 2.0

Os modelos atuais do estado da arte para o reconhecimento da fala requerem grandes quantidades de dados de áudios transcritos para atingir um bom desempenho (AMODEI *et al.*, 2015). Contudo, segundo Baevski *et al.* (2020), em muitos cenários, os dados etiquetados são muito difíceis de serem obtidos. Com o surgimento do aprendizado auto-supervisionado, tornou-se possível a aprendizagem das representações gerais com dados não rotulados sobretudo na área de reconhecimento de fala.

Partindo dessa vertente, Baevski *et al.* (2020) desenvolveram o modelo Wav2vec 2.0 para o reconhecimento da fala por meio do áudio bruto. Esse modelo consiste principalmente em um *encoder* CNN, uma rede contextualizada Transformer e um módulo de quantização. A entrada do modelo é um sinal de áudio bruto e o codificador CNN aprende representações latentes da fala (XU *et al.*, 2021).

As primeiras camadas do modelo, são camadas convolucionais responsáveis por processarem os áudios brutos ($X$) para uma representação latente ($Z$). Após essas camadas , tem-se a rede de contexto, que segundo Fan *et al.* (2020), é composta por 12 blocos Transformer e, antes de serem enviadas as representações latentes, todos os passos de tempo são amostrados aleatoriamente como índices iniciais. Em linhas gerais, a rede seleciona um conjunto aleatório para processamento, em seguida esses dados são mascarados. O Transformer contextualiza essas representações mascaradas e gera representações de contexto ($C$) (FAN *et al.*, 2020), como representada na Figura 23.

Figura 23 – Arquitetura Wav2vec 2.0




Fonte: Adaptada de Baevski *et al.* (2020).

O modelo é composto também por um módulo de quantização que realiza a conversão de um espaço contínuo para um discreto. Foram discretizadas a saída do *encoder* de caracte-

rísticas em um conjunto finito de representações da fala por meio da quantização de produto. Essa quantização consiste em escolher representações quantizadas ($Q$) de vários *codebooks* e concatená-las (BAEVSKI *et al.*, 2020). Portanto, considerando uma quantidade de *codebooks* com entradas, são escolhidas uma entrada para cada *codebook* e depois são concatenados os vetores resultantes. Para a escolha da entrada correta para cada *codebook*, os autores utilizaram a função Gumbel Softmax (JANG; GU; POOLE, 2016)

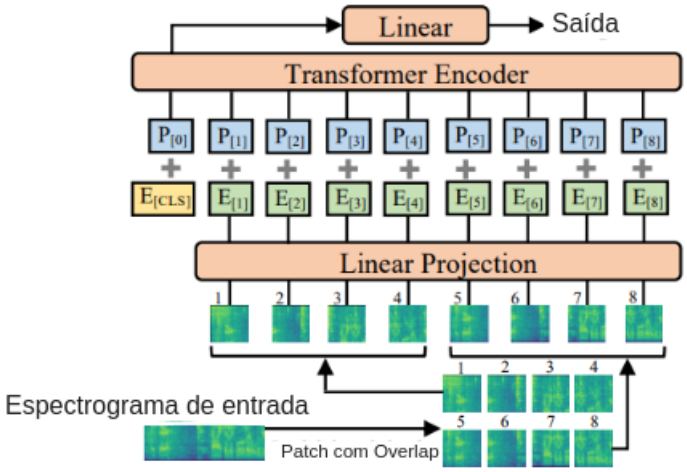
Para uma representação latente mascarada, o modelo precisa identificar a verdadeira representação quantizada da fala, sendo assim, os autores utilizaram a Contrastive *loss*, que maximiza a similaridade entre representações de dados em uma categoria, enquanto minimiza em categorias diferentes (PARK; AHMAD; HAIN, 2022). Sendo assim, essa função de perda computa o cosseno de similaridade entre as representações de contexto e representações quantizadas (BAEVSKI *et al.*, 2020), penalizando portanto, quando pares semelhantes são agrupados distantes. Outra função empregada foi a Diversity *loss* para incentivar o uso igual das entradas em cada *codebook*, garantindo a diversidade e evitando a redundância. Por fim, o modelo é ajustado para reconhecimento de fala adicionando uma projeção linear inicializada aleatoriamente no topo da rede de contexto em $C$ classes que representam o vocabulário da tarefa (BAEVSKI *et al.*, 2020).

### 2.4.2 Audio Spectrogram Transformer

O Audio Spectrogram Transformer (AST) (GONG; CHUNG; GLASS, 2021) foi desenvolvido com o propósito de aprender a associar espectrogramas de áudio com os rótulos correspondentes a esses áudios. Este modelo baseia-se no Vision Transformer (DOSOVITSKIY *et al.*, 2020b), mencionado na Seção anterior. No contexto do AST, o áudio é transformado em uma representação visual por meio de um processo que envolve segmentação em *frames* e realizada a conversão em coeficientes de filtro Mel com 128 dimensões. Durante o pré-processamento é aplicada a janela de Hamming de 25ms a cada 10ms. Isso resulta em um espectrograma de entrada com dimensões de $128 \times 100$t, em que "t" representa o tempo em segundos .

A imagem é subdividida em uma sequência de *patches*, conforme ilustrado na Figura 24. Cada *patch* tem dimensões de $16 \times 16$, com um deslocamento (*overlapping*) de 6 unidades entre eles. O número de *patches* (N) é determinado pela fórmula $N = 12(100-16)/10$. Cada *patch* é então transformado em um *embedding* de tamanho 768 usando uma camada linear, e é adicionado um *embedding* posicional. Um *token* especial de classificação [CLS] é inserido no início de cada sequência. Os *embeddings* resultantes são utilizados como entrada para o Transformer.

Figura 24 – Arquitetura Audio Spectrogram Transformer




Fonte: Adaptada de Gong, Chung e Glass (2021).

## 2.5 Trabalhos Relacionados

Nesta seção, são apresentados os trabalhos relacionados à classificação de variações linguísticas no Português do Brasil, começando pela abordagem de trabalhos em inglês. São discutidos os principais modelos utilizados pelos autores, *datasets* empregados, resultados significativos e conclusões alcançadas. Para o Português do Brasil, são detalhados quatro estudos na área de classificação de sotaques, com destaque para o trabalho pioneiro de Batista *et al.* (2018), Batista (2019), que empregou o conjunto de dados Braccent. Além disso, são apresentados os resultados mais relevantes e as conclusões obtidas pelos autores.

### 2.5.1 Trabalhos Relacionados em outros idiomas

A tarefa de classificação de variações linguísticas tem despertado considerável interesse científico, uma vez que, é um aspecto peculiar de um respectivo idioma e apresenta desafios de pesquisa interessantes. No trabalho realizado por Jain, Upreti e Jyothi (2018), foi proposta uma arquitetura multitarefa que aprende de maneira conjunta tanto classificação de sotaques para o inglês quanto um modelo acústico.

A arquitetura multitarefa proposta pelos autores é composta por três blocos. O primeiro bloco (A) corresponde a um sistema *baseline* que recebe as características acústicas dos áudios, como espectrograma mel e recursos que capturam informações do falante. O segundo bloco (B) consiste em uma rede treinada para classificar variações e apresenta camadas que são compartilhadas com primeiro bloco (A). O terceiro (C) é uma rede independente treinada para classificação. O *embeddings* do bloco C são utilizados para aumentar as características ou quantidade de entradas com variações linguísticas do bloco A.

Tanto para o desenvolvimento do modelo acústico quanto na identificação de variações

linguísticas no bloco C, Jain, Upreti e Jyothi (2018) escolheram uma Time Delay Neural Network (TDNN) (do inglês, Time Delay Neural Network) (PEDDINTI; POVEY; KHUDANPUR, 2015), uma rede *feedforward* que considera a sequência temporal dos dados. Na tarefa de classificação foram aplicados recursos intermediários aprendidos a partir das camadas TDNNs compartilhadas. O sistema *baseline* utilizado pelos autores foi uma TDNN *feedforward*. Para o treinamento foi utilizado o *dataset* do Common Voice[1] com variações do americano, inglês, canadense, galês e irlandês. Como teste os autores utilizaram o inglês falado na Nova Zelândia e alguns do sul asiático como a Índia, Paquistão e Sri Lanka.

O melhor resultado obtido para classificação foi 82.6%, para acurácia de validação. Os *embeddings* produzidos nessa etapa foram utilizados como entrada no bloco A, como citado anteriormente. Com essa técnica os autores obtiveram Word Error Rate (WER) (do inglês, *Word Error Rate*) de 50.9% para o sul asiático e 22.9% para as demais. Os resultados obtidos apresentaram desempenho superior em comparação ao *baseline*, com redução de 10% do WER no conjunto de teste para sotaques desconhecidos, concluindo que os recursos de *embeddings* aprendidos oferecem melhorias no desempenho (JAIN; UPRETI; JYOTHI, 2018).

Wang *et al.* (2021) propuseram um sistema de identificação de dialetos com arquitetura *end-to-end* a partir de um aprendizado por transferência, onde foi utilizado um modelo de ASR multilíngue. Primeiramente, eles treinaram um modelo ASR baseado em um Conformer (GULATI *et al.*, 2020), que consiste em um módulo para processamento de fala composto por camadas convolucionais e *feedforward*. Esse método foi escolhido pois permite a fusão dos recursos das diferentes camadas e reduz a dimensionalidade, foi utilizada também o Connectionist Temporal Classification (CTC)(GRAVES *et al.*, 2006), cuja intuição consiste em produzir uma única letra para cada frame de entrada, de modo que a saída tenha o mesmo comprimento da entrada, ou seja, essa função realiza um alinhamento automático (JURAFSKY; MARTIN, 2009). Em segundo lugar, os autores inicializaram o classificador baseado na arquitetura *end-to-end* com um codificador compartilhado com o modelo ASR multilíngue pré-treinado, citado anteriormente.

Os autores verificaram a aplicabilidade do modelo utilizando a tarefa proposta na competição Oriental Language Recognition Challenge 2020 (AP20-OLR) (LI *et al.*, 2020) para identificação de dialetos da língua oriental e também de outras línguas, catalão e grego, por exemplo. Foram utilizados mais de um *baseline* como o I-vector, que consiste em um vetor de dimensão reduzida usado para modelar a variabilidade do sinal de variações linguísticas (DEHAK *et al.*, 2011) e o X-vector. Nesse trabalho, os autores realizaram um comparativo com os demais grupos da competição. Em comparação com a equipe vencedora, a média da taxa de erro de classificação diminuiu 19.5%, ou seja, o grupo vencedor obteve 0.0783% e os autores conseguiram obter 0.0594% no conjunto de testes. Para o ASR conseguiram reduzir o Equal error rate (EER) (do inglês, *Equal error rate*) em 25.2% com relação ao vencedor (WANG *et al.*,

[1] https://commonvoice.mozilla.org/

2021). Os resultados foram satisfatórios e demonstraram que inicializar o classificador a partir de um modelo ASR *end-to-end* pré-treinado, possibilitou a melhora significativa para identificação de dialetos.

O trabalho proposto por Chitkara *et al.* (2022) aborda dois métodos para melhorar o desempenho dos sistemas ASR: *embeddings* pré-treinadas e funções de custo. Foi explorado o impacto desses métodos com diferentes arquiteturas e especificamente com dois idiomas, o inglês e o espanhol, considerando nativos e não nativos. A arquitetura proposta foi baseada em um aprendizado multitarefa ou *multi-task learning*, ou seja, o *encoder* aprendeu simultaneamente a tarefa principal (ASR) e a tarefa auxiliar (classificação de variação linguística), o que segundo os autores contribui para uma melhor generalização e desempenho em dados com variações linguísticas. As arquiteturas utilizadas foram o Transformer com base no trabalho de Likhomanenko *et al.* (2020) e o Wav2vec 2.0. Como *baseline*, foi utilizada uma combinação do Emformer (SHI *et al.*, 2021b), que é uma extensão do modelo Transformer, combinando camadas convolucionais e *self-attention*, com o Recurrent Neural Network Transducer (RNN-T) (LIU *et al.*, 2021), que consiste em uma rede recorrente com uma rede de transcrição.

Os autores adicionaram o classificador de sotaques ao modelo *baseline* (Enformer) na 14ª camada intermediária, para o espanhol, e 16ª camada para o inglês. Obtiveram uma melhoria de 2.4% na taxa de erro para o espanhol e 1.52% para o inglês, com relação ao *baseline* que não utilizava o Multi-task learning (MTL). Chitkara *et al.* (2022) procuravam responder se *embeddings* pré-treinadas poderiam modelar também diferentes sotaques. Sendo assim, os autores treinaram o modelo Enformer para o espanhol tendo como base o modelo pré-treinado em inglês, além disso utilizaram também o MTL. Com essas técnicas obtiveram melhoria média relativa de 7.29% na taxa de erro com relação ao *baseline*.

Com o Transformer, o uso do MTL obteve uma taxa de erro médio de 10.4% para o inglês e 28.1% para o espanhol da Argentina, porém não apresentaram melhoria para o Wav2vec 2.0, que obteve melhor desempenho sem o uso do MTL e ao aplicar o pré-treinamento, com taxa de erro médio de 25% para o espanhol argentino, por exemplo e 11.9% para o inglês, considerando as melhores taxas de erro. Chitkara *et al.* (2022) concluíram que a quantidade de dados disponíveis em inglês poderiam ser aproveitadas para construir sistemas ASR em outros idiomas e que *embeddings* pré-treinadas podem melhorar o desempenho dos modelos considerando falantes nativos, para o inglês e funcionam para falantes da Espanha e América Latina.

No trabalho proposto por Zuluaga *et al.* (2023), realizou a classificação de sotaques utilizando modelos multilingual englobando o inglês, italiano, alemão e espanhol, por meio de dois modelos: ECAPA-TDNN (do inglês, Emphasized Channel Attention, Propagation and Aggregation–TDNN) e Wav2vec 2.0 XLSR. O ECAPA-TDNN apresentou resultados estado da arte na tarefa de verificação de locutor, sendo assim os autores propuseram um treinamento utilizando SpecAugmentation(PARK *et al.*, 2019), uma técnica de aumento de dados, aplicado

diretamente às entradas de recursos de uma rede neural e outro experimento sem aumento de dados. Para o modelo Wav2vec 2.0 XLSR, foi realizado o *fine-tuning* utilizando o Common Accent, subconjunto obtido do Common Voice. Os resultados apontaram que o Wav2vec 2.0 XLSR obteve desempenho superior ao ECAPA-TDNN, com acurácia de 95.1%, enquanto o ECAPA-TDNN obteve 79%. Os autores observaram também uma melhoria na performance dos modelos ao aplicar o aumento de dados com SpecAugment. Por exemplo, o Wav2vec 2.0 XLSR obteve uma aumento de 2%, com 97% e o ECAPA obteve 89.7%. Para o italiano o modelo até 99% de acurácia utilizando o Wav2vec2 XLSR e o espanhol 68.5%.

Os autores concluíram que o aumento de dados é eficaz para o tratamento de dados desbalanceados, especialmente para sotaques com poucos recursos no mesmo idioma e que o Wav2vec 2.0 XLSR mantém um alto nível de acurácia.

Na Tabela 1 é apresentada uma síntese dos Trabalhos relacionados para outros idiomas.

| Autores | Modelos | Datasets |
|---|---|---|
| (JAIN; UPRETI; JYOTHI, 2018) | TDNN (*multi-task*) | Common Voice |
| (WANG *et al.*, 2021) | Conformer (*multi-task*) | AP20-OLR |
| (CHITKARA *et al.*, 2022) | Transformer e Wav2vec 2.0 | Datasets privados |
| (ZULUAGA *et al.*, 2023) | ECAPA-TDNN e Wav2vec 2.0 XLSR | Common Accent |

Tabela 1 – Trabalhos Relacionados em outros idiomas

### 2.5.2 Trabalhos Relacionados no Português Brasileiro

No português brasileiro, o trabalho pioneiro na área foi o de Batista (2019) que propôs um estudo das diferentes variedades linguísticas existentes com base em modelagens estatísticas e técnicas de aprendizado de máquina. A autora desenvolveu uma base denominada Braccent, devido a necessidade de construir um *dataset* que representasse as variações existentes no Brasil, contemplando sete sotaques: nortista, baiano, fluminense, mineiro, carioca, nordestino e sulista. Foi realizada uma captura online dessas falas, totalizando 1.757 amostras de áudio, entre 8 a 14 segundos. Além dessa base foi utilizada o *dataset* Ynoguti (YNOGUTI, 1999), que contém 1.600 amostras de áudio de 6 estados e a Corpus Forense do Português Brasileiro (CFPB) , com 411 amostras de áudios.

Batista (2019) utilizou três classificadores: Gaussian Mixture Model - Universal Background Model (GMM-UBM), I-vector e uma adaptação do GMM com o Suport Vector Machine (SVM). O GMM consiste em um modelo estatístico composto por uma mistura de distribuições gaussianas. São normalmente utilizados como modelo paramétrico da distribuição de probabilidade (REYNOLDS *et al.*, 2009). Esse modelo permite a descrição dos dados como distribuições gaussianas. Foram utilizados dois cenários para os experimentos: *closed-set*, validação no mesmo *dataset* onde ocorreu o treinamento, e *cross-dataset* ou validação externa, em que o teste ocorre em um *dataset* diferente do treinamento.

No cenário *closed-set* foi realizada a validação cruzada das amostras de treinamento e teste com as bases de dados do Braccent, Ynoguti e CFPB. Dentre os três *datasets* a base de dados Braccent foi a que obteve melhor desempenho para o *closed-set* com o modelo GMM-UBM com f1-*score* de 91% e taxa de erro de 19%. No resultado obtido com a fusão dos três *datasets*, o sistema GMM-SVM obteve os melhores resultados com F1-*score* de 84% e taxa de erro de 18%.

Para à validação externa, os resultados foram inferiores. No treinamento com o Braccent e teste realizado com o Ynoguti, o modelo GMM-SVM obteve melhores resultados com f1-*score* de 75% e taxa de erro de 20%. Para o treinamento com o Ynoguti e CFPB os resultados foram inferiores em comparação ao treinamento realizado com o Braccent. Segundo Batista (2019), as características fonéticas dos sotaques no Braccent apresentam maior representatividade das variações linguísticas, em relação às outras bases, o que permitiu um desempenho melhor desse *dataset* no cenário de validação externa. A autora conclui também a partir dos resultados obtidos para o treinamento realizado com o Ynoguti, que tais resultados contradizem a hipótese de que uma base de dados com baixo ruído de gravação e foneticamente balanceada é suficiente para o ajuste e um modelo acústico de reconhecimento de sotaques (BATISTA *et al.*, 2018; BATISTA, 2019). Além dessa questão, percebeu-se que alguns erros estavam polarizados em classes específicas, como o sotaque mineiro e o sulista. A possível explicação, segundo a autora, está relacionada a proximidade geográfica que pode interferir de alguma forma no sotaque de outra região.

A base de dados CFPB, obteve pior performance sobretudo quando utilizada para teste. Conforme a autora, realizar testes em uma base com pouco controle de gravação e amostras de fala espontânea podem ocasionar uma piora na performance. Em compensação, testes realizados com controle de gravação, com menos ruídos e mais clareza na pronúncia, contribuem para uma melhora no nível de reconhecimento de variações (BATISTA, 2019), como foi para a base Ynoguti. A autora ressalta a importância do treinamento com uma base de dados com variabilidade de fala e não apenas isenta de ruídos. Essa variabilidade possibilita que os sistemas modelem com maior precisão os traços linguísticos (BATISTA, 2019).

Com base nos *datasets* Braccent e Ynoguti, Tostes *et al.* (2021) aplicou o uso de arquiteturas diferentes para classificação de variações. Foram utilizadas redes convolucionais 1D e 2D, redes híbridas com camadas convolucionais, LSTM e também em sua versão bidirecional, BiLSTM. Para o desenvolvimento das arquiteturas, alguns hiperparâmetros como tipo e ordem das camadas, funções de ativação e número de neurônios foram selecionados manualmente, utilizando os conjuntos de treinamento e validação (TOSTES *et al.*, 2021). As camadas com LSTM ou BiLSTM, receberam como entrada os vetores de características obtidos nas camadas convolucionais. Posteriormente, os vetores obtidos nessas camadas recorrentes foram utilizados nas camadas totalmente conectadas.

Nos experimentos realizados com a base de dados Braccent, os melhores resultados foram obtidos com a CNN 1D com LSTM, com f1-*score* de 88%. No segundo experimento foi

utilizada a base Ynoguti para classificação de sotaque por estado, os resultados foram superiores para todas as arquiteturas utilizadas pelo autor, sobretudo para a CNN 1D com BiLSTM, com f1-*score* de 88% (TOSTES, 2022). O autor concluiu que as arquiteturas que obtiveram melhores resultados foram as que utilizavam faixas inteiras de frequências do espectrograma e que os vetores aprendidos podem contribuir para uma análise da fala e para o desenvolvimento de novas técnicas de classificação de sotaques (TOSTES, 2022).

No trabalho proposto por Lopes, Andrade e Komati (2021), é realizado um comparativo entre as ferramentas existentes para transcrição de áudios para as diferentes variações regionais. Foram selecionadas duas ferramentas, o Google Cloud Speech API [2] e Wit.ai [3], sendo utilizados cerca de 1.648 áudios da base Braccent, sendo que a cada áudio submetedio a API dessas ferramentas retornavam a transcrição. Para análise das ferramentas, foi utilizada a métrica de *Levenshtein* que define o menor número necessário de operações para transformar um texto transcrito para o original (LOPES; ANDRADE; KOMATI, 2021), em que quanto mais próximo de zero, melhor o resultado. Na análise dos resultados por variação linguística, os autores reportaram que o Google Cloud Speech API obteve melhor desempenho nas variações nordestina e baiana e piores resultados para nortista e carioca e que o Wit.ai obteve boa performance para todas as variações linguísticas exceto a carioca. Esses resultados levaram os autores a concluir que os sistemas com resultado inferior em certas variações (como a carioca), provavelmente não possuem em seu treinamento exemplos das variações linguísticas com baixo desempenho.

Posteriormente, Almeida (2022) comparou os resultados de Lopes, Andrade e Komati (2021) e Batista *et al.* (2018), Batista (2019) utilizando tanto a Regressão Logística Multiclasse e aplicando *transfer learning* no Wav2vec 2 Base com o conjunto de dados do Braccent. Os resultados apontaram que o Wav2vec 2.0 alcançou precisão geral de 69% e F1-score de 38%, enquanto a Regressão Logística obteve apenas precisão de 39%. Os autores também realizaram uma análise de gênero, mas não obtiveram diferenças significativas quanto ao desempenho do modelo Wav2vec 2 Base. O autor enfatizou a importância de avaliar esses modelos com outros conjuntos de dados, sendo necessário também aplicar experimentos com modelos pré-treinados para o Português do Brasil.

No trabalho do Milan (2023), foi realizada uma ampliação do *dataset* Braccent, por meio da coleta de áudios utilizando uma aplicação web. Foram utilizados conjuntos de frases para realização das gravações e adicionada também informações sobre os interlocutores como idade, gênero, local de nascimento, local onde reside, tempo de residência e local onde os pais nasceram. Essa nova base é denominada Sotaque Brasileiro[4].

O autor utilizou aprendizado não-supervisionado com o algoritmo K-means, para fazer os agrupamentos dos espectrogramas, uma vez que, procurou identificar por meio do agrupamento,

[2] https://cloud.google.com/speech-to-text
[3] https://wit.ai/
[4] <https://github.com/sotaque-brasileiro/sotaque-brasileiro>

regiões geográficas associadas a grupos distintos. Após a definição do número de classes, foi realizado o treinamento para as 5 classes obtidas com o K-means, utilizando o KNN e arquitetura de redes neurais. O treinamento supervisionado com KNN foi obtido F1-score de 73% e com a CNN customizada 78% no conjunto reservado para inferência. O autor concluiu que não ficou clara a separação geográfica para as classes, uma vez que, a partir das porcentagens classificadas para cada grupo, assumiu-se um sotaque muito similar aos estados do Rio Grande do Sul, São Paulo e Ceará (MILAN, 2023), o que em termos linguísticos tem muita diferença .

Conforme o autor os algoritmos de classificação foram satisfatórios para amostras desconhecidas, sobretudo com o uso de redes neurais convolucionais e alguns fatores como o balanceamento e o aumento de dados sem modificações que alterem significativamente o áudio original, podem contribuir para uma eventual melhoria (MILAN, 2023).

Na Tabela 2 é apresentada uma síntese dos Trabalhos relacionados para o Português Brasileiro.

| **Autores** | **Modelos** | **Dataset** |
|---|---|---|
| (BATISTA, 2019) | GMM-UBM e SVM | Braccent |
| (TOSTES *et al.*, 2021) | CNNs e LSTM | Braccent |
| (ALMEIDA, 2022) | Wav2vec 2 Base | Braccent |
| (MILAN, 2023) | KNN e CNNs | Sotaque Brasileiro |

Tabela 2 – Trabalhos Relacionados para classificação de sotaques no Português Brasileiro

CAPÍTULO

# 3

# MATERIAIS E MÉTODO

No decorrer deste capítulo é apresentado os materiais e métodos utilizado para classificação automática de regionalismos no Português Brasileiro . Na Seção 3.1 são descritos os *datasets* de fala espontânea e não espontânea. Na Seção 3.2 estão descritos os materiais utilizados para desenvolvimento. Posteriormente, na Seção 3.3, são apresentados o protocolo experimental, em que estão descritos o pré-processamento utilizado no decorrer deste trabalho.

Na Subseção 3.3.1, os subconjuntos obtidos após pré-processamento, contendo nove variações linguísticas para o Spotify Podcast, 3 para o CORAA ASR e 6 sotaques para o Braccent. Para classificação dos regionalismos foram utilizados quatro modelos: CNN 1D LSTM, CNN2D, Audio Spectrogram Transformer e Wav2vec 2.0 XLSR, presentes na Subseção 3.3.3. São apresentadas a arquitetura e hiperparâmetros utilizados no treinamento para cada modelo. Na Subseção 3.3.4, constam a divisão dos *datasets* durante o treinamento.

Por fim, no decorrer da Subseção 3.3.5, tem-se a descrição das métricas para avaliação no cenário *closed-set*, em que a avaliação ocorre no mesmo *dataset* do treinamento e *cross-dataset*, avaliação em outro *dataset*. Para melhor compreensão acerca das características aprendidas pelo modelo, foi utilizado o Grad-CAM, apresentado na Subseção 3.3.6.

## 3.1 Datasets

Nesta seção, são descritos os conjuntos de dados utilizados neste estudo: Spotify Podcasts, CORAA ASR e Braccent. O Spotify Podcasts é composto por uma variedade de *podcasts* em português de Portugal e do Brasil. Para este trabalho foram utilizados apenas *podcasts* referentes ao Português do Brasil. O CORAA ASR, voltado para o português brasileiro, reúne diversos conjuntos de dados em um único corpus e, assim como o Spotify Podcasts, é caracterizado por conter fala espontânea. Por outro lado, o Braccent, que visa representar as variações do português brasileiro, é composto por fala preparada, ou seja, gravadas a partir de sentenças lidas, contendo

7 regionalismos do Português Brasileiro.

### 3.1.1 Braccent

O Braccent é um dataset desenvolvido por Batista (2019) através de uma aplicação Web. Para confecção desse *dataset* cada locutor realizou a pronúncia de 16 sequências de frases foneticamente balanceadas com transcrição fonética por meio do programa Praat[1], em um total aproximado de 3 minutos de fala (BATISTA, 2019). O *dataset* é composto por 1.743 amostras de fala preparada e cada gravação é identificada por gênero e sotaque regional (BATISTA, 2019). As informações dos sotaques e número de gravações são apresentadas na Tabela 3.

Tabela 3 – Descrição do número de gravações das regiões presentes no dataset Braccent

| Sotaques | Número de Gravações | Feminino | Masculino |
|---|---|---|---|
| Nortista | 27 | 8 | 19 |
| Baiano | 183 | 103 | 80 |
| Fluminense | 114 | 63 | 51 |
| Mineiro | 148 | 63 | 85 |
| Carioca | 82 | 47 | 35 |
| Nordestino | 344 | 153 | 191 |
| Sulista | 845 | 435 | 410 |

Fonte: Adaptada de Batista (2019).

### 3.1.2 Spotify Podcasts

O *dataset* Spotify contém *podcasts* que apresentam cerca de 123 mil episódios em português do Brasil e português de Portugal, conforme descrito por Tanaka *et al.* (2022). Essa base foi obtida mediante licença para uso acadêmico. A média de palavras da transcrição para o português brasileiro é de aproximadamente 9.539 palavras, como detalhado na Tabela 4. Para esse trabalho, como mencionado anteriormente, não foram utilizados *podcasts* com sotaque de Português de Portugal, apenas brasileiro.

Tabela 4 – Descrição dos *Podcasts* do Português Brasileiro e de Portugal

| | |
|---|---|
| Número de *Podcasts* | 16.131 |
| Número de Episódios | 123.054 |
| Duração Média dos Episódios | $\sim$ 37.3 min. |
| Palavras Transcritas em Média por Episódio | $\sim$ 9.539 |

Fonte: Adaptada de Tanaka *et al.* (2022).

Os episódios transcritos contendo português do Brasil totalizam 114.387. Para cada áudio, armazenado no formato ogg, tem-se a transcrição e o metadado que contém as respectivas

[1] <https://www.fon.hum.uva.nl/praat/>

informações: URI para o *podcast*, nome do *podcast*, descrição, idioma, *link* RSS do *podcast*, descrição do episódio, duração, prefixo referente ao nome do *podcast*, nome do episódio.

Dentre os gêneros mais utilizados estão *podcasts* sobre negócios com 26.915 mil, educação com 23.541 mil e sobre esportes com 13.467 mil episódios. O *dataset* contempla também um arquivo em formato XML para cada programa de *podcast* que apresenta informações de metadados mais específicas como autor, sobre o que se trata o episódio, *links* dos áudios. A localidade onde foi gravado, informação de interesse para essa pesquisa, está presente em parte dos episódios e foi usada, conforme descrito posteriormente, para montagem do datasets utilizados nos experimentos.

Para os propósitos dessa dissertação, organizamos um subconjunto desse *dataset* para realizar experimentos englobando diferentes gêneros. Os critérios de inclusão utilizados foram primeiramente possuir informação sobre a região, que foram identificados por meio da aplicação de filtros no arquivo de metadados.Cada episódio referente a um *podcast* apresenta uma transcrição completa em formato JSON e no mesmo arquivo uma transcrição minutada por palavra, o que indica que provavelmente foi realizada de maneira automática.

### 3.1.3 CORAA

O CORAA ASR (*Automatic Speech Recognition*) (JÚNIOR *et al.*, 2021) é um conjunto de dados públicos para reconhecimento automático de fala para o idioma português brasileiro. Contém cerca de 290,77 horas de áudios e suas respectivas transcrições. Esse *dataset* é uma composição de áudios de outros 5 projetos:

- ALIP (GONÇALVES, 2019)
- C-ORAL Brasil (RASO; MELLO, 2012)
- NURC-Recife (JR *et al.*, 2016)
- SP-2010 (MELLO; PETTORINO; RASO, 2012)
- TEDx (SALESKY *et al.*, 2021)

Os áudios do CORAA foram validados por anotadores e transcritos para utilização do ASR. No Quadro 1, são apresentadas informações específicas de cada *dataset* utilizado na composição do CORAA, como os gêneros dos áudios, variedade linguística, quantidade em horas, por exemplo.

## 3.2 Hardware e Software

Nesta seção estão as especificações técnicas de hardware e bibliotecas utilizadas. Para realização dos experimentos foi utilizada a GPU NVIDIA Titan RTX com 128 GB RAM. Os

Quadro 1 – Informações Gerais do Dataset CORAA

| Corpus | Gêneros | Variedade linguística | Horas |
|---|---|---|---|
| ALIP | Entrevistas, Diálogos | São Paulo | 78 |
| C-ORAL Brasil I | Diálogos, Conversações | Minas Gerais | 21.13 |
| NURC Recife | Diálogos, Entrevistas, Conferência | Recife | 279 |
| SP2010 | Conversações,Entrevistas, Leituras | São Paulo Capital | 65 |
| TEDx Português | Conversações de Palco | vários regionalismos | 249 |

Fonte: Adaptada de Júnior *et al.* (2021).

experimentos foram executados no container do Docker (versão 19.03.14) utilizando o Cuda (versão 11.8) . A seguir são apresentadas informações sobre as bibliotecas:

- Librosa (versão 0.10.0) [2]: pacote Python para análise de áudio e música, realizando o pré-processamento do áudio;
- Pydub (versão 0.25.1) [3]: pacote Python para manipulação de áudios, assim como o Librosa.
- Python (versão 3.8.10) [4]: é uma linguagem de programação de alto nível que suporta diversos paradigmas de programação.
- Pytorch (versão 2.1.1+cu118)[5]: consiste em uma interface de programação em Python para aprendizado de máquina e auxilia no desenvolvimento de modelos.
- Tensorboard (versão 2.13.0) [6] : ferramenta para análise dos modelos desenvolvidos por meio de imagens, gráficos, histogramas.
- Torchaudio (versão 2.1.1+cu118)[7]: biblioteca para processamento de áudio e sinal utilizando Pytorch.

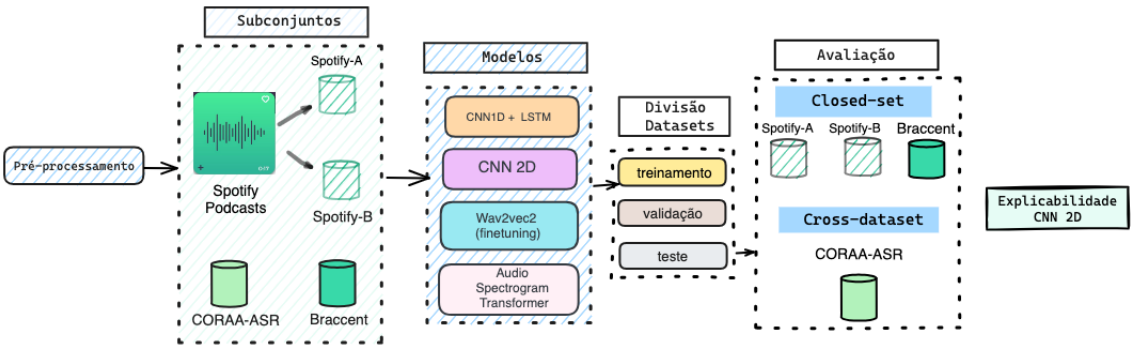
## 3.3 Método

Nesta seção, serão detalhados os métodos utilizados. O fluxograma representado pela Figura 25 apresenta uma sumarização das etapas propostas para o desenvolvimento deste trabalho.

O protocolo geral para investigar **classificadores** para variedades linguísticas no Brasil consistiu na divisão dos subconjuntos, etapas de pré-processamento, desenvolvimento dos classificadores, análise dos resultados e comparação com a literatura.

[2] https://librosa.org/doc/latest/index.html
[3] https://github.com/jiaaro/pydub
[4] https://www.python.org/
[5] https://pytorch.org/
[6] https://www.tensorflow.org/tensorboard?hl=pt-br
[7] https://pytorch.org/audio/stable/index.html

Figura 25 – Fluxograma Geral




Fonte: Elaborada pelo autor.

### 3.3.1 Subconjunto Spotify Podcasts

Foram organizados dois subconjuntos para avaliação de dois cenários: (i) com número limitado de locutores (Spotify-A) e (ii) com uma quantidade maior de falantes (Spotify-B). Para o primeiro subconjunto, Spotify-A, foram selecionados manualmente os episódios de podcasts que apresentassem sotaque brasileiro com base nos dados geográficos do locutor no arquivo metadados, como por exemplo, “Rádio Manaus” e “Puc Minas” ou ainda, expressões idiomáticas indicativas de um sotaque específico (por exemplo, “Bah”, “Oxe”) na descrição do episódio.

O campo de descrição foi utilizado para confirmar a localização do locutor, dado que em alguns casos incluía nomes dos convidados. Para esse subconjunto foram escolhidos podcasts com no máximo dois locutores. É importante ressaltar também que as divisões das variações linguísticas nos respectivos subconjuntos não passaram por validação de um linguista.

#### *Spotify-A*

No subconjunto Spotify-A foram utilizadas 9 variações linguísticas, considerando a divisão por estado. Nesse subconjunto não foi realizada a diarização, os áudios foram cortados em segmentos entre 5 10 segundos. Foi aplicada a remoção de fundo musical utilizando a biblioteca Spleeter (mencionada na Subseção 3.3.2) e remoção dos cinco primeiros e cinco últimos segmentos pois normalmente continham vinhetas de abertura e introdução repetidas.

Nos episódios de *podcasts* que envolviam dois locutores, os metadados forneciam o nome dos locutores juntamente com sua localidade. Quando essas informações não estavam disponíveis nos metadados, procedeu-se com uma verificação manual. Isso envolveu a audição do episódio em que o locutor se apresentava, identificando verbalmente sua localidade de origem. Essa abordagem foi adotada para garantir a precisão na atribuição das localidades aos locutores nos metadados do *podcast*.

Com as etapas de pré-processamento citadas, foi obtido inicialmente um subconjunto com 7.782 segmentos de áudios. Os números por estado são especificados na Tabela 5. Na Figura

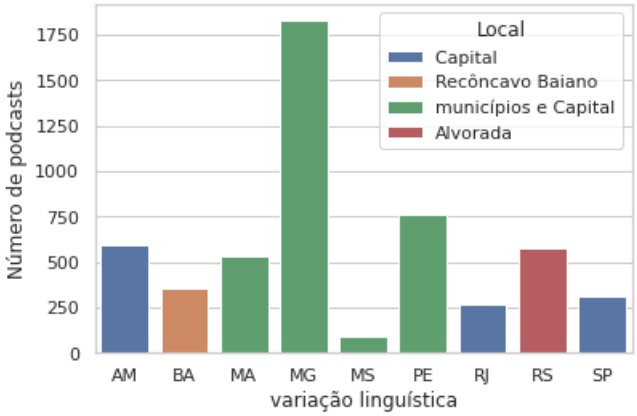
26, é apresentado graficamente os números citados anteriormente por localidade.

| **Variação** | **segmentos** | **Horas** | **Locutores** |
|---|---|---|---|
| AM | 487 | $\sim 1$ | $2-3$ |
| BA | 625 | $\sim 1$ | 1 |
| MA | 1,326 | $\sim 3$ | $2-3$ |
| MS | 109 | $\sim 0.3$ | $1-2$ |
| MG | 2,461 | $\sim 5$ | $2-3$ |
| PE | 1,624 | $\sim 4$ | $1-3$ |
| RJ | 284 | $\sim 0.8$ | $1-3$ |
| RS | 402 | $\sim 1$ | $1-2$ |
| sp-SP | 464 | $\sim 1$ | $1-3$ |
| Total | 7,782 | $\sim 17$ | $\sim 23$ |

Tabela 5 – Informação sobre o subconjunto Spotify-A por estado

No estado de Minas Gerais, os *podcasts* coletados abrangem falantes de vários municípios do estado, sobretudo os que contém as filiais da PUC Minas e também da capital, Belo Horizonte. Assim como do Maranhão e Pernambuco que abrangem não somente a capital mas também demais localidades do estado.

Figura 26 – Quantidade de *podcasts* por variação linguística e localização - Spotify-A




Fonte: Elaborada pelo autor.

### *Spotify-B*

Com o objetivo de aumentar a quantidade de locutores por estado, foi realizada uma nova consulta no metadados do *dataset* do Spotify. Foram selecionados buscas por palavras chaves, semelhante ao realizado no Spotify-A. No entanto, dessa vez, para cada *podcast* selecionado os áudios foram analisados manualmente, verificando se os locutores pertenciam ao estado em questão.

Para o Spotify-B, foi realizada a redução das classes para 5, utilizando os estados que apresentavam maior quantidade de locutores. Cada segmento de áudio foi obtido mediante

diarização, utilizando Whisper[8] (RADFORD *et al.*, 2022) e Pyannote [9]como está melhor descrito na Subseção 3.3.2. É importante mencionar que a ferramenta utilizada não é totalmente precisa na contagem de locutores. Em casos onde as falas de um locutor são mais curtas, elas são ocasionalmente atribuídas erroneamente a outro locutor com falas mais longas. Detalhes sobre o subconjunto Spotify-B podem ser encontrados no repositório do GitHub [10].

| Variação | segmentos | Horas | Locutores |
|---|---|---|---|
| BA | 4.667 | $\sim$ 13.5 | $\sim$ 25 |
| MG | 3.618 | $\sim$ 11.9 | $\sim$ 60 |
| PE | 14.008 | $\sim$ 48.23 | 102 |
| RJ | 2.979 | $\sim$ 8.38 | $\sim$ 50 |
| sp-SP | 11.906 | $\sim$ 30.88 | 85 |
| Total | 37.178 | $\sim$ 112.89 | $\sim$ 322 |

Tabela 6 – Informação sobre o subconjunto Spotify-B por estado

### 3.3.2 Pré-processamento dos Datasets

Considerando os *datasets*: subconjunto Spotify-A, subconjunto Spotify-B, CORAA e Braccent, descreveremos o pré-processamento utilizado. Observamos que para cada *dataset* foi utilizado um procedimento diferente devido às suas características e segundo os objetivos da pesquisa.

1. **Conversão do áudio e reamostragem** - reamostrar os áudios do *dataset* do Spotify para 16kHZ e conversão para formato `.wav`. Para os *datasets* CORAA ASR e Braccent não foi necessário reamostragem pois já estão nessa taxa de amostragem.

2. **Remoção de música e silêncio** - remover a música de fundo dos episódios dos podcasts referentes ao dataset do Spotify com a biblioteca Spleeter[11] que inclui modelos pré-treinados. Após essa etapa foi aplicada a remoção do silêncio, em pausas longas aplicando um algoritmo de detecção de atividade vocal.

3. **Corte dos áudios em segmentos de 5 a 10 segundos (Spotify-A)** - Para o subconjunto A do Spotify, realizou-se o recorte dos áudios para uma duração entre 5 a 10 segundos. Esta faixa de tempo foi escolhida para garantir a inclusão de pelo menos uma sentença completa proferida pelo locutor. Durante esse processo de recorte, não houve uma distinção entre os diferentes locutores presentes nos áudios, o que levou à obtenção de segmentos que podem incluir mais de um locutor e apresentar interrupções nas falas.

[8] <https://github.com/openai/whisper>
[9] <https://github.com/pyannote/pyannote-audio>
[10] <https://github.com/aryamtos/spotify-subset/>
[11] <https://github.com/deezer/spleeter>

4. **Diarização, Transcrição e Corte (Spotify-B)** - separação dos áudios dos episódios por locutor utilizando Whisper e Pyannote, e posterior corte levando em consideração a identificação dos locutores.

5. **Cálculo do espectrograma** - aplicação da STFT (Short-Time Fourier Transform), com tamanho da janela igual a 3000 e passo igual a 2000 para gerar entradas para o modelo CNN1D LSTM (TOSTES *et al.*, 2021; TOSTES, 2022). Para o AST o áudio de entrada passou por um pré-processamento onde foi convertido em coeficientes de filtro Mel com 128 dimensões. Esses coeficientes foram calculados utilizando uma janela de Hamming de 25ms a cada 10ms, resultando em um espectrograma de dimensões 128x100t, onde "t" representa a duração do áudio em segundos.

### 3.3.3 Modelos

Nesta subseção estão descritos os modelos utilizados, detalhes e informações sobre as arquiteturas. As arquiteturas utilizadas foram: CNN 1D LSTM, Wav2vec 2.0 e Audio Spectrogram Transformer. A Tabela 7 apresenta uma síntese dos hiperparâmetros utilizados no decorrer dos experimentos para os respectivos modelos. Notar que para o Wav2Vec2.0 há dois valores para *learning rate* usados nas duas etapas do treinamento: transferência de aprendizado e *finetuning*, conforme descrito posteriormente.

Tabela 7 – Hiperparâmetros de otimização utilizados para os modelos (CNN 1D LSTM, Wav2vec2 e AST)

| Modelo | Otimizador | *learning rate* | Batch size | *dropout* |
|---|---|---|---|---|
| CNN1D LSTM | Adam | $1e-5$ | 16 | 0.4 |
| CNN2D | Adam | 1e-5 | 32 | – |
| Wav2vec 2.0 | AdamW | $6e-5$ e $3e-5$ (finetuning) | 4 | 0.375 |
| AST | AdamW | $1e-5$ | 8 | 0.375 |

Fonte: Elaborada pelo autor.

#### *CNN 1D LSTM*

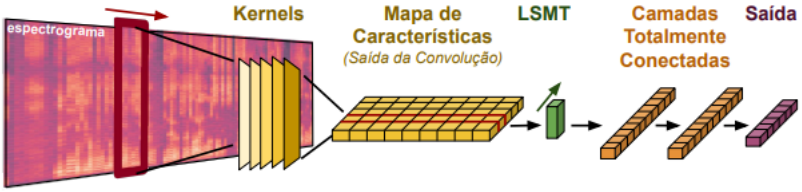
Com base nos principais trabalhos relacionados à classificação de variações linguísticas do Português do Brasil, utilizamos um modelo CNN 1D LSTM conforme descrito por Tostes *et al.* (2021). Esta arquitetura apresentou bons resultados na classificação de sotaques considerando fala por meio de sentenças lidas com o conjunto de dados Braccent e reportada em trabalhos anteriores.

A entrada deste modelo consiste em uma faixa de frequência do espectrograma. Na Figura 27, a primeira camada é composta por uma convolução 1D com 32 filtros, aplicada nos espectrogramas de entrada. Em seguida, uma camada de *dropout* é empregada para minimizar o

*overfitting*. O vetor de características resultante da etapa de convolução é então utilizado como entrada para a LSTM.

Após a LSTM, são empregadas duas camadas *fully connected*, com função de ativação Tanh e 128 neurônios. Por fim, a última camada desta arquitetura corresponde à de classificação. A quantidade de neurônios nesta camada foi ajustada conforme a quantidade de variações consideradas em cada *dataset*.

Figura 27 – Arquitetura CNN 1D + LSTM




Fonte: Adaptada de Tostes *et al.* (2021).

Para o treinamento, foram utilizados os seguintes hiperparâmetros: uma taxa de *dropout* de 0.4, a função de perda Entropia Cruzada, o otimizador Adam e uma taxa de aprendizado de $1 \times 10^{-5}$, escolhida com base na análise da perda no conjunto de desenvolvimento durante o treinamento. O treinamento foi conduzido ao longo de 100 épocas, com parada antecipada e uma paciência de 50 épocas. Além disso, o modelo foi treinado cinco vezes, utilizando cinco sementes diferentes com os valores: 123, 42, 101, 1 e 5, resultando em inicializações diferentes.

### *CNN2D*

Com o objetivo de analisar a classificação não somente em faixas de frequência, mas também ao longo do tempo, foi utilizada a CNN 2D com entrada um espectrograma de tamanho $227 \times 227 \times 3$. Os espectrogramas foram obtidos como descrito na Subseção 3.3.2. Na Tabela 8 são apresentadas informações sobre a arquitetura utilizada, a qual foi definida de forma empírica com experimentos incrementais partindo de uma arquitetura mais compacta para uma com mais camadas e parâmetros até atingir um ponto de saturação no conjunto de desenvolvimento.

Foram utilizados para o treinamento taxa de aprendizado $1e-5$, otimizador Adam, epsilon $1e-8$. A taxa de aprendizado é ajustado dinamicamente durante o treinamento com o *scheduler*. Foi utilizado o *CosineAnnealingWarmRestarts* que diminui a taxa de aprendizado de acordo com a função cosseno. Os hiperparâmetros utilizados para a função foram:

- $T_0$ ( número de epocas para a primeira reinicialização) : 1
- $T_{mult}$ (fator multiplicado a $T_0$ ): 2
- $eta_{min}$ (taxa de aprendizado mínima): $1e-5 * 0.01$

| Tipo | Tamanho do filtro/stride/padding | Tamanho da entrada |
|---|---|---|
| Conv2d+BatchNorm+ReLU | K=(5,5), padding=1 | $16 \times 225 \times 225$ |
| Maxpool2d | K=(4,4), stride=2 | $16 \times 111 \times 111$ |
| Conv2d+BatchNorm+ReLU | K=(5,5), padding=1 | $16 \times 109 \times 109$ |
| Maxpool2d | K=(4,4), stride=2 | $16 \times 53 \times 53$ |
| Conv2d+BatchNorm+ReLU | K=(5,5), padding=1 | $32 \times 51 \times 51$ |
| Maxpool2d | K=(4,4), stride=2 | $32 \times 24 \times 24$ |
| Conv2d+BatchNorm+ReLU | K=(5,5), padding=1 | $64 \times 22 \times 22$ |
| Maxpool2d | K=(4,4), stride=2 | $64 \times 10 \times 10$ |
| Flatten | - | 6400 |
| Linear+ReLU | - | $6400, 2048$ |
| Linear+ReLU | - | $2048, 1024$ |
| Linear+ReLU | - | $1024, 512$ |
| Softmax | - | |

Tabela 8 – Arquitetura da CNN 2D

O treinamento foi realizado durante 100 épocas com paciência de 50 épocas utilizando a função de perda Entropia Cruzada. O repositório para a CNN2D encontra-se disponível no GitHub[12]

### *Wav2vec 2.0 XLSR-53*

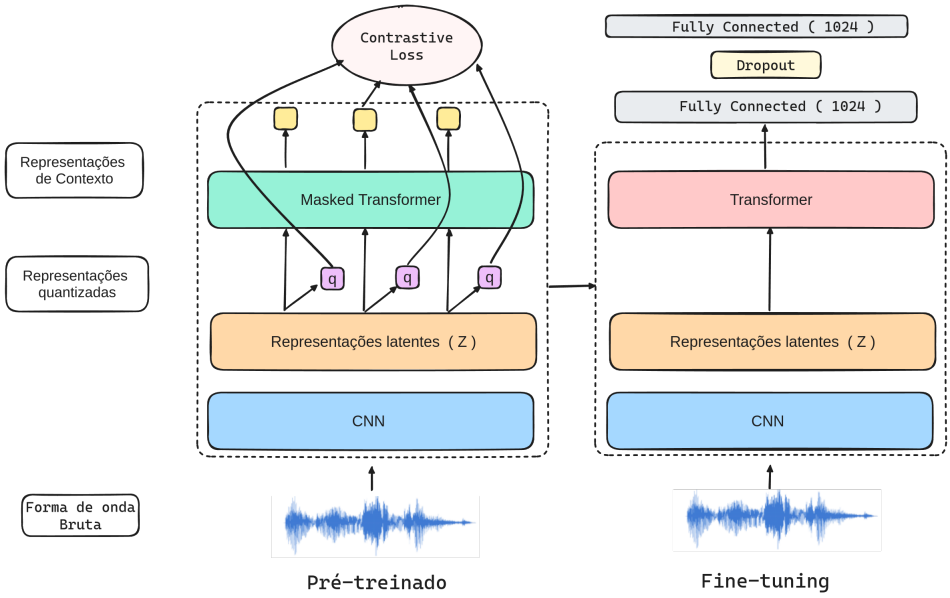
O Wav2vec 2.0 XLSR-53 consiste em um modelo multilingual e foi utilizado em trabalhos anteriores para a tarefa de ASR no português do Brasil e na classificação de sotaques em outros idiomas como apresentado nos trabalhos relacionados. Sendo assim, foi realizado o *fine-tuning* desse modelo, a partir do modelo pré-treinado disponível no HuggingFace[13] e o repositório com o código utilizado encontra-se disponível no GitHub[14]. Para a classificação, como representada na Figura 28, foram utilizadas: Função Tanh na saída do modelo base, uma camada densa contendo 1024 neurônios, *dropout* de 0.375, seguido por outra camada densa com a mesma quantidade de neurônios anterior e a camada de classificação.

Conforme Conneau *et al.* (2019), na primeira parte do treinamento foi realizado o congelamento do *feature encoder*. Os hiperparâmetros utilizados foram: otimizador AdamW, com beta1 0.9 e beta2 0.999, epsilon de $1e-8$, taxa de aprendizado $3e-5$. Inicialmente o modelo foi treinado durante 5 épocas com taxa de aprendizado de $3e-5$, para a classificação binária com batch size de tamanho 4. Após essa etapa foi realizado o *fine-tuning* do modelo com taxa de aprendizado de $3e-5$ durante 5 épocas.

Para o cenário multiclasse foi utilizada taxa de aprendizado de $3e-5$ e o treinamento realizado durante 10 épocas. Após o treinamento realizou-se um novo ajuste utilizando os pesos (*checkpoints*) obtidos no treinamento inicial. Foi utilizado um aquecimento da taxa de

[12] <https://github.com/aryamtos/accent-classification-audio-cnns/>
[13] <https://huggingface.co/facebook/wav2vec2-large-xlsr-53>
[14] <https://github.com/aryamtos/wav2vec2-brazilian-regionalism-classification>

Figura 28 – Fine-tuning Wav2vec 2.0 XLSR




Fonte: Elaborada pelo autor.

aprendizado de 0 a $3e-5$ nas primeiras duas épocas e depois foi aplicado o decaimento linear. Foram considerados também 2 seeds diferentes: 42 e 101.

*Audio Spectrogram Transformer*

O Audio Spectrogram Transformer é um modelo utilizado em tarefas de classificação, tendo como base o ViT, como mencionado em capítulos anteriores. Esse modelo pré-treinado está disponível apenas para o idioma inglês. Foi utilizado o modelo pré-treinado também disponibilizado no HuggingFace[15] e aplicado o *transfer learning* para classificação de sotaques. Inicialmente foi realizada a normalização dos áudios de entrada para obtenção das estatísticas necessárias, o código desse modelo encontra-se no GitHub[16] com base no repositório original do modelo. Durante o treinamento foi realizado o congelamento do modelo base e os hiperparâmetros foram ajustados conforme o tipo de classificação. Para a classificação binária foi utilizada a taxa de aprendizado de $1e-5$, *batch size* tamanho 8, otimizador Adam, com beta1 0.9 e beta2 0.999. Para o cenário multiclasse foi utilizada taxa de aprendizado entre $2e-5$ e $5e-5$. Em ambos os treinamentos foram utilizadas 20 epocas.

### 3.3.4 Divisão dos Datasets

Para o treinamento dos modelos para que não houvesse contaminação foi realizada a separação de podcasts específicos apenas para teste (e não presentes no treinamento) para o Spotify, sendo que as amostras do CORAA ASR referente as classes PE, SP e MG são

[15] <https://huggingface.co/MIT/ast-finetuned-audioset-10-10-0.4593>
[16] <https://github.com/aryamtos/ast-brazilian-portuguese>

utilizadas apenas para teste. Em alguns casos, devido a questões de balanceamento de locutor e segmentos de áudios, optou-se por reduzir a quantidade de amostras por classe no treinamento e validação. Para todos os experimentos realizados o conjunto de testes permaneceu fixo tanto para o subconjunto do Spotify-B quanto o subconjunto do CORAA ASR. É importante ressaltar também que para essa tarefa de classificação é necessário a utilização de locutores distintos entre conjunto de treinamento, desenvolvimento e teste para que não ocorra contaminação de locutores, portanto não foi considerada a técnica de validação cruzada para divisão do conjunto de dados.

Devido ao desbalanceamento entre as classes, foi utilizada também durante o treinamento dos modelos uma função de custo ponderada ou *weighted loss* que é uma modificação da função de custo padrão por meio de pesos. Esses pesos foram utilizados com o objetivo de atribuir uma maior penalidade às classificações incorretas da classe com menor quantidade de amostras. Sendo assim, foi atribuído um peso maior a classe minoritária e menor para a classe majoritária. O peso para cada classe foi calculado conforme a Equação 3.1, em que utiliza-se o número de amostras de uma classe específica ( $amostras$) e o número total de amostras no conjunto de dados ($total$).

$$1 - (\frac{amostras}{total}) \tag{3.1}$$

Nas subseções seguintes estão descritas as quantidades específicas utilizadas para treinamento, validação e teste.

### *Divisão Subconjunto - Spotify-B*

Para a classificação binária entre as classes PE e SP foram utilizados as quantidades de áudios apresentadas na Tabela 9. Foram utilizados no total 50 locutores para cada classe no conjunto de treinamento, para o desenvolvimento 16 locutores de São Paulo (SP) e 25 do Recife (PE). Para teste foram distribuídos 11 locutores para cada uma das classes.

| **Variação** | **Treinamento** | Dev | **Teste** | **Locutores** |
|---|---|---|---|---|
| `PE` | 8.161 | 2.304 | 534 | $\sim 86$ |
| `sp-SP` | 7.998 | 2.353 | 500 | $\sim 77$ |
| Total | 10.465 | 4.657 | 1034 | $\sim 163$ |

Tabela 9 – Divisão do Spotify-B para classificação binária no cenário com muitos locutores

Nos experimentos realizados utilizando uma quantidade um pouco menor de locutores com o objetivo de avaliar o desempenho dos modelos Wav2vec 2.0 e o Audio Spectrogram Transformer foram utilizados de 9 a 11 locutores no treinamento. Na Tabela 10, são apresentadas as quantidades específicas utilizadas após a redução de locutores. No conjunto de desenvolvimento foram utilizados 5 locutores para cada classe e o conjunto de teste permaneceu fixo, como mencionado anteriormente.

| Variação | Treinamento | Dev | Teste | Locutores |
|---|---|---|---|---|
| PE | 1.135 | 339 | 534 | $\sim 27$ |
| sp-SP | 945 | 96 | 500 | $\sim 27$ |
| Total | 2.080 | 435 | 1034 | $\sim 54$ |

Tabela 10 – Divisão do Spotify-B para classificação binária reduzindo a quantidade de locutores durante o treinamento

Durante a classificação multiclasse (PE, MG, SP) devido ao desbalanceamento entre as classes, foi reduzida a quantidade de segmentos de áudio e também de locutores. Foram utilizados 25 locutores para o treinamento, 10 locutores para o desenvolvimento e 11 para teste. Na Tabela 11 são apresentadas as demais informações por classe.

| Variação | Treinamento | Dev | Teste | Locutores |
|---|---|---|---|---|
| PE | 484 | 146 | 534 | $\sim 46$ |
| sp-SP | 494 | 180 | 500 | $\sim 46$ |
| MG | 484 | 120 | 287 | $\sim 46$ |
| Total | 1.462 | 446 | 1034 | $\sim 138$ |

Tabela 11 – Divisão do Spotify-B para classificação multiclasse balanceando a quantidade de locutores e segmentos de áudios entre as classes.

### *Divisão Subconjunto - CORAA-ASR*

O dataset CORAA-ASR foi utilizado no cenário *cross-dataset* (avaliação com dataset diferente do treinamento) como teste. Portanto, foram selecionadas amostras de áudios para as três classes: PE, SP e MG. A Tabela 12 contém informações para cada classe utilizada, devido a incerteza na quantidade de locutores, essa informação não foi reportada.

| Variação | Segmentos | Horas |
|---|---|---|
| PE | 353 | $\sim 0.9$ |
| sp-SP | 371 | $\sim 1$ |
| MG | 351 | $\sim 0.6$ |
| Total | 1.075 | $\sim 2.5$ |

Tabela 12 – Divisão do CORAA-ASR para validação no cenário *cross-dataset*

### *Divisão Subconjunto - Braccent*

Para o *dataset* Braccent foram utilizadas 6 classes: Baiano, Carioca, Fluminense, Nordestino, Mineiro e Sulista. O sotaque Nortista não foi considerado pois continha baixo volume de áudios disponíveis, dificultando a geração dos subconjuntos de treinamento, desenvolvimento e teste. Na Tabela 13 estão descritas as quantidades de áudios utilizados para treinamento, desenvolvimento e teste e a quantidade de locutores distintos para cada regionalismo.

| Variação | Treinamento | Dev | Teste | Locutores |
|---|---|---|---|---|
| Baiano | 94 | 53 | 27 | $\sim 14$ |
| Carioca | 47 | 16 | 18 | $\sim 6$ |
| Fluminense | 59 | 38 | 12 | $\sim 10$ |
| Nordestino | 126 | 100 | 58 | $\sim 24$ |
| Mineiro | 88 | 30 | 24 | $\sim 12$ |
| Sulista | 225 | 128 | 97 | $\sim 38$ |
| Total | 639 | 365 | 236 | $\sim 104$ |

Tabela 13 – Divisão do Braccent para classificação multiclasse contendo informações sobre quantidade de áudios e locutores distintos por regionalismo

A divisão foi realizada com o objetivo que não houvesse contaminação de locutores e balanceando a quantidade de locutores entre treinamento, validação e teste. As informações sobre a quantidade de locutores utilizada para cada classe é descrita na Tabela 14.

| Variação | Treinamento | Dev | Teste |
|---|---|---|---|
| Baiano | 6 | 4 | 4 |
| Carioca | 3 | 1 | 2 |
| Fluminense | 4 | 4 | 2 |
| Nordestino | 9 | 8 | 7 |
| Mineiro | 6 | 3 | 3 |
| Sulista | 23 | 9 | 6 |
| Total | 51 | 29 | 24 |

Tabela 14 – Quantidade de locutores para treinamento, validação e teste por regionalismo.

### 3.3.5 Avaliação

Para a avaliação do modelo de detecção dos regionalismos, foram utilizadas como métricas a precisão (Equação 3.2), taxa de revocação (Equação 3.3) e o f1-score (Equação 3.4) de cada uma das classes. As siglas $VP$ representa verdadeiro positivo, $FP$ falso positivo e $FN$ falso negativo. Dessas métricas, o f1-score é considerado um dos mais importantes pois além de equilibrar as duas métricas, precisão e revocação, é útil em casos que possam existir uma diferença significativa no número de exemplos de cada classe, fornecendo uma visão geral da performance do modelo.

$$\frac{VP}{VP+FP} \tag{3.2}$$

$$\frac{VP}{VP+FN} \tag{3.3}$$

$$2 * \frac{(prec * recall)}{(prec + recall)}, \tag{3.4}$$

onde *prec* é a precisão.

Para o treinamento foram utilizadas as divisões dos conjuntos de dados com base descrito na Subseção 3.3.4 e protocolos conforme o modelo utilizado ( Subseção 3.3.3 ). Foram utilizados dois cenários para avaliação dos modelos: *Closed-set* e *cross-dataset*. No cenário *closed-set* o modelo foi avaliado no mesmo *dataset* do treinamento, no *cross-dataset*, a avaliação ocorreu em um *dataset* diferente, para este trabalho foi utilizado o CORAA ASR. Para o treinamento realizado com o *dataset* Spotify a validação externa ocorreu com o *dataset* CORAA ASR. Para o dataset Braccent foi realizada a avaliação apenas no cenário *closed-set*.

### 3.3.6 Explicabilidade

Para uma melhor compreensão dos padrões aprendidos pela CNN2D, foi utilizado o Grad-CAM (do inglês, Gradient-weighted Class Activation Mapping) (SELVARAJU *et al.*, 2017), uma técnica que destaca regiões importantes na imagem que contribuem para uma determinada classificação. Essa abordagem utiliza informações do gradiente, que consiste nas características aprendidas na última camada convolucional. Portanto foram utilizadas as seguintes etapas:

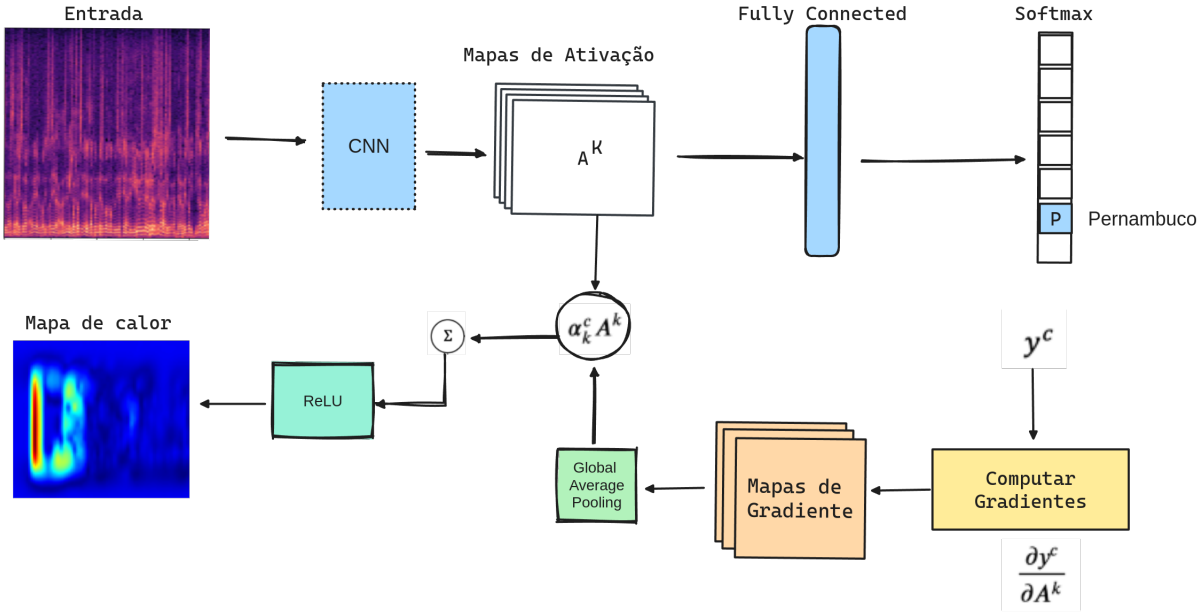
- Cálculo do gradiente: cada mapa de características da última camada da CNN captura informações do espectrograma. Sendo assim, os gradientes da classe referente aos mapas de características são utilizados para computar uma pontuação de importância para destacar as regiões relevantes na imagem de entrada. Na Figura 29 o cálculos dos gradientes é representado matematicamente por $\frac{\partial y^c}{\partial A^k}$. O $y^c$ corresponde a pontuação prevista antes de passar pela Softmax e $A^k$, um mapa de características bidimensional.

- Calcular valores alfa: nessa etapa é aplicado o Global Average Pooling dos gradientes obtidos para obtenção dos valores alfa, que são valores de importância. Na Figura 29 os valores alfa são representados por $\alpha_k^c$.

- Gerar mapas de calor: após a obtenção dos valores alfa, é realizada uma soma ponderada dos mapas de ativação. Os pesos correspondem aos valores alfa obtidos. Na Figura 29 é representada por $\sum_k \alpha_k^c A^k$.

Após as etapas descritas, a função ReLU é aplicada à soma ponderada. Isso resulta na obtenção de um mapa de calor, o qual foi ampliado para o mesmo tamanho da imagem e sobreposto sobre ela. Além de permitir um conhecimento sobre quais frequências foram utilizadas, essa técnica permite saber se a rede aprendeu algum padrão espúrio.

Figura 29 – Representação do mapa de calor gerado pelo Grad-CAM




Fonte: Adaptada de Selvaraju *et al.* (2017).

CAPÍTULO

# 4

# RESULTADOS E DISCUSSÃO

Neste capítulo, são apresentados os resultados obtidos com os seguintes modelos: CNN 1D LSTM, CNN2D, Wav2vec 2.0 e Audio Spectrogram Transformer. Cada um desses modelos foram avaliados nos conjuntos de dados descritos no Capítulo 3, utilizando como métrica principal o F1-score, utilizada em estudos anteriores para classificação de variações linguísticas no Português do Brasil.

Para avaliar os modelos, foram seguidos os protocolos definidos no capítulo anterior, com modificações nos hiperparâmetros conforme o modelo e o conjunto de dados. Para a CNN 1D LSTM, treinada do zero, foram utilizadas quantidades de épocas maiores do que nos demais modelos. Com o modelo pré-treinado Wav2vec 2.0 XLSR multilingual, foi realizado o *fine-tuning* dos parâmetros, enquanto para o modelo Audio Spectrogram Transformer pré-treinado, foi aplicado o *transfer learning*.

Em termos dos dados utilizados, foram considerados dois cenários: um primeiro cenário com poucos locutores, utilizando o subconjunto Spotify-A, e um segundo cenário com muitos locutores, utilizando o subconjunto Spotify-B. Cada um desses cenários foi avaliado tanto no *closed-set* quanto no *cross-dataset*, utilizando o CORAA ASR. Com o conjunto de dados Braccent, foi realizada apenas a avaliação no cenário *closed-set*. Conforme mencionado anteriormente, as divisões nos conjuntos de treinamento, desenvolvimento e teste garantiram que estes conjuntos fossem compostos por diferentes locutores.

Um detalhamento mais completo dos valores obtidos para as métricas de precisão e revocação para cada um dos cenários e as os mapas de calor após aplicar o Grad-CAM, são apresentadas no decorrer deste capítulo.

## 4.1 Cenário com poucos locutores

### 4.1.1 Closed-set: Spotify-A

No experimento inicial realizado, foram utilizadas 9 variações linguísticas do Spotify-A utilizando o modelo CNN1D LSTM. Foi realizada uma divisão do subconjunto em 70% treinamento, 15% validação e 15% teste. O objetivo dessa divisão foi avaliar o quanto a contaminação de locutores entre as divisões podem trazer resultados que não correspondem ao aprendizado de características das variações. O F1-score obtido com essa divisão foi de 96%. Em contraste, no outro cenário em que para cada uma das divisões tinha-se locutores distintos, o F1-score foi de 24% para as 9 classes.

Apresentamos propositalmente os dois cenários para demonstrar que o problema em questão torna-se mais difícil quando não possuímos exemplos no treinamento que possam fornecer padrões espúrios para que o modelo aprenda como: o locutor, os dispositivos utilizados na captura, o ruído de fundo, e outras características da gravação. Como desejamos estudar a classificação dos regionalismos por modelos agnósticos a essas características, esse resultado evidencia os desafios de generalização relacionados ao problema.

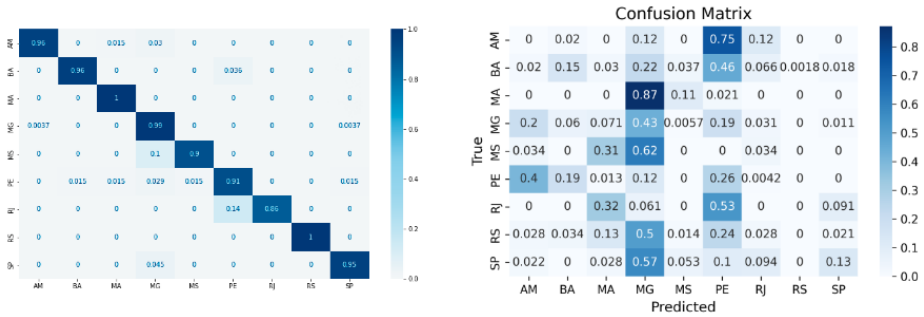



Figura 30 – Matriz de Confusão para o cenário closed-dataset no Spotify-A com contaminação e sem contaminação de locutores. (a): contaminação de locutores,(b): sem contaminação

Na Tabela 15, é apresentado o resultado geral da classificação binária (PE, SP) para as 5 sementes utilizadas. A precisão para PE é relativamente alta ($83 \pm 9\%$), indicando que a maioria das classificações positivas estão corretas. No entanto, embora a classificação positiva seja precisa, muitas amostras classificadas como PE não pertenciam a essa classe. Por outro lado, para a classe SP, observa-se uma alta taxa de revocação, o que significa que a maioria das amostras é corretamente identificada . Contudo, com baixa precisão($26 \pm 2\%$). O F1-score geral ($34 \pm 11\%$) apresentado na Tabela 15 indica um desequilíbrio entre a taxa de revocação e a precisão.

Na Tabela 16 é apresentado os resultados para o cenário com 3-classes (MG, PE, SP). O modelo obteve maior quantidade de acerto para a classe PE com F1-score de $70 \pm 8\%$. Os resultados apresentados na Figura 30(b) e na Tabela 16 mostram um enviesamento para classes

| classe | Precisão | Revocação | F1-score |
|---|---|---|---|
| PE | $83 \pm 9\%$ | $20 \pm 20\%$ | $28 \pm 24$ |
| SP | $26 \pm 2\%$ | $87 \pm 17\%$ | $40 \pm 4$ |
| F1-score Geral | | | $34 \pm 11\%$ |

Tabela 15 – Cenário Closed-set – Classificação binária: média e desvio padrão das avaliações no Spotify-A utilizando o modelo CNN1D-LSTM entre as classes PE, SP)

com um maior número de amostras de áudio como MG e PE e poucas previsões precisas para a classe SP.

| classe | Precisão | Revocação | F1-score |
|---|---|---|---|
| MG | $40 \pm 9\%$ | $46 \pm 27\%$ | $40 \pm 18$ |
| PE | $60 \pm 13\%$ | $89 \pm 9\%$ | $70 \pm 8$ |
| SP | $89 \pm 2\%$ | $26 \pm 2\%$ | $40 \pm 2$ |
| F1-score Geral | | | $50 \pm 8\%$ |

Tabela 16 – Cenário Closed-set – Multiclasse: média e desvio padrão das avaliações no Spotify-A utilizando o modelo CNN1D-LSTM entre as classes MG, PE, SP

A partir dos valores das métricas obtidas, percebe-se que os resultados favoreceram a classe PE provavelmente devido à sua maior quantidade de áudios. Em contraste, a classe SP apresentou um F1-score de aproximadamente 40%. Observa-se também um desvio padrão relativamente alto em ambas as classificações, indicando que para certas inicializações o modelo não conseguiu convergir utilizando poucos locutores.

Os resultados apontam também que o modelo não consegue aprender características dos sotaques e provavelmente aprende padrões espúrios durante o treinamento ou simplesmente decora o locutor que está no treinamento e portanto consegue classificar bem a variação linguística se o mesmo locutor estiver na validação, como foi possível observar no experimento considerando o vazamento de locutores entre treinamento e teste.

### 4.1.2 Closed-set: Braccent

Para os experimentos com o Braccent utilizamos os modelos AST e CNN2D, cujos resultados são apresentados nas Tabelas 17 e 18, respectivamente. Mesmo com a tentativa de balancear a quantidade de locutores na divisão do dataset e minimizar o impacto do desbalanceamento na quantidade de áudios atribuindo pesos as classes durante o treinamento, os resultados não foram satisfatórios.

Em ambos modelos, a classe Sulista, a qual tem o maior número de áudios para treinamento, obteve um F1-score maior que os demais regionalismos. É notável que a classe Baiana apesar de ter 94 áudios de treinamento em comparação a classe mineira que possuía 88 áudios, teve um resultado muito inferior (o AST, Tabela 17, foi capaz de gerar predições para a classe Mineiro). Isso provavelmente se deve ao fato dos áudios rotulados como Baiano conterem uma

quantidade abrangente de variações provenientes de distintas regiões, mas que, em decorrência da divisão dos sotaques por Nascentes (1953), foram atribuídas a essa única classe.

Possíveis razões para a dificuldade em alcançar bons resultados incluem a baixa variedade de dados, ou seja, poucos locutores distintos, e também o baixo volume na quantidade de horas disponíveis nos regionalismos, sobretudo, mas não exclusivamente, nas classes Carioca e Fluminense.

| classe | Precisão | Revocação | F1-score |
|---|---|---|---|
| Baiano | 0.04% | 0.07% | 0.05% |
| Carioca | - | - | - |
| Fluminense | - | - | - |
| Mineiro | 24% | 33% | 28% |
| Nordestino | 24% | 60% | 34% |
| Sulista | 69% | 39% | 50% |
| F1-score Geral | | | 18% |

Tabela 17 – Resultado para o dataset Braccent utilizando o modelo Audio Spectrogram Transformer (AST) no cenário multiclasse

| classe | Precisão | Revocação | F1-score |
|---|---|---|---|
| Baiano | 0.10% | 0.04% | 0.05% |
| Carioca | - | - | - |
| Fluminense | - | - | - |
| Mineiro | - | - | - |
| Nordestino | 45% | 55% | 50% |
| Sulista | 47% | 76% | 58% |
| F1-score Geral | | | 19% |

Tabela 18 – Resultado para o dataset Braccent utilizando o modelo CNN2D no cenário multiclasse

### 4.1.3 Cross-dataset: Spotify-A e CORAA ASR

Para o cenário cross-dataset, foi realizado o treinamento com todo o *dataset* Spotify-A e avaliação com o CORAA ASR, para as classe MG,PE e SP. Na Tabela 19, são apresentados os resultados para a classificação binária entre PE e SP. O modelo classificou erroneamente PE como SP, com a maioria dos resultados concentrados nessa classe e obteve F1-score geral de $38 \pm 3\%$.

| classe | Precisão | Revocação | F1-score |
|---|---|---|---|
| PE | $35 \pm 10\%$ | $10 \pm 3\%$ | $15 \pm 5$ |
| SP | $48 \pm 2\%$ | $81 \pm 6\%$ | $60 \pm 3$ |
| F1-score Geral | | | $38 \pm 3\%$ |

Tabela 19 – Cenário *Cross-dataset* – Classificação Binária: média e desvio padrão das avaliações utilizando o modelo CNN1D-LSTM entre as classes PE e SP com o CORAA ASR

Para três classes, o modelo confunde PE com MG e vice-versa com F1-score geral de $30 \pm 1\%$. Na Tabela 20, percebe-se que não houve nenhum acerto para a classe PE. Enquanto para a classe SP o modelo consegue classificar com desempenho razoável ($77 \pm 6\%$).

| classe | Precisão | Recall | F1-score |
|---|---|---|---|
| MG | $11 \pm 4\%$ | $12 \pm 5\%$ | $11 \pm 4$ |
| PE | – | – | – |
| SP | $77 \pm 3\%$ | $79 \pm 13\%$ | $77 \pm 6$ |
| F1-score Geral | | | $30 \pm 1\%$ |

Tabela 20 – Cenário *Cross-dataset* – Classificação Multiclasse: média e desvio padrão das avaliações utilizando o modelo CNN1D-LSTM entre as classes MG, PE e SP com o CORAA ASR

## 4.2 Cenário com muitos Locutores

### 4.2.1 Closed-set: Spotify-B

No cenário com muitos locutores, foram avaliados quatro modelos: CNN1D LSTM, CNN2D, Wav2vec2 e Audio Spectrogram Transformer. Para os resultados obtidos com a CNN1D LSTM durante a classificação binária utilizando uma maior quantidade de locutores (51 locutores - PE, 52 locutores - SP), foi possível observar um melhor equilíbrio entre a precisão e a revocação para as classes (PE, SP). Embora a precisão para a classe PE tenha sido ligeiramente inferior, como apresentado na Figura 31, com $61 \pm 3\%$, observou-se um aumento significativo na taxa de revocação ($58 \pm 7\%$) comparado a resultados anteriores ($20 \pm 20\%$) ao utilizar o Spotify-A.

Para o sotaque de São Paulo (SP), uma precisão de $57 \pm 3\%$ foi registrada, enquanto o F1-score geral para a CNN1D LSTM foi de $59 \pm 2\%$. Isso indica um melhor equilíbrio entre precisão e recall em comparação com os resultados apresentados na Tabela 16 para o Spotify-A (menos locutores). É importante destacar que os resultados com um maior número de locutores apresentaram um desvio padrão menor em comparação com o cenário com poucos locutores.

| classe | CNN 1D LSTM | CNN 2D | Wav2vec2 | AST | Wav2vec2* |
|---|---|---|---|---|---|
| **Closed-set** | | | | | |
| PE | $59 \pm 4\%$ | $54 \pm 0.06$ | $83 \pm 0.02\%$ | 55% | 72% |
| SP | $58 \pm 4\%$ | $69 \pm 0.02\%$ | $84 \pm 0.01\%$ | 25% | 55% |
| F1-score geral | $59 \pm 2\%$ | $61 \pm 0.04\%$ | 83% | 40% | 63% |

Tabela 21 – Cenário Closed-set - Classificação Binária : F1-score dos modelos para classificação binária ( PE,SP). O símbolo * indica que foi reduzida a quantidade de locutores.

A CNN2D demonstrou um desempenho considerável em comparação com a CNN1D LSTM, exibindo uma variação menor entre os resultados para as diferentes classes. Especificamente para a classe SP, o modelo aprendeu características mais distintivas e obteve uma vantagem significativa, conforme apresentado na Tabela 21, apresentando um F1-score geral de

$61 \pm 0.04\%$. Esse desempenho foi alcançado mesmo com uma quantidade menor de imagens de espectrograma (484 para PE e 494 para SP). Percebe-se que a CNN2D foi capaz de capturar melhor as relações espaciais entre as frequências e o tempo, o que é crucial para identificar padrões relevantes para a classificação de variações linguísticas. Por outro lado, ao utilizar a LSTM, a compreensão do contexto temporal da fala, como a evolução das frequências ao longo do tempo e a duração dos fonemas, pode ser mais adequada. A CNN2D como pode ser observado na Figura 31, obteve uma precisão de $78 \pm 0.03\%$ para a classe PE enquanto para a classe SP uma alta taxa de revocação de $87 \pm 0.02\%$ (Figura 32).

Na Figura 31 e 32, é evidente o contraste entre a precisão e a revocação nos modelos CNN1D LSTM, CNN2D e Wav2vec 2.0. Observa-se que a precisão para a classe PE é mais alta, enquanto a revocação para a classe SP é mais significativa nos resultados. Além disso, a precisão menor, especialmente nos modelos CNN1D LSTM e CNN2D, indica uma taxa mais alta de falsos positivos para SP. Essas diferenças sugerem nuances na capacidade dos modelos em identificar corretamente as amostras de cada classe, destacando a necessidade de considerar tanto a precisão quanto a revocação ao avaliar o desempenho dos modelos.

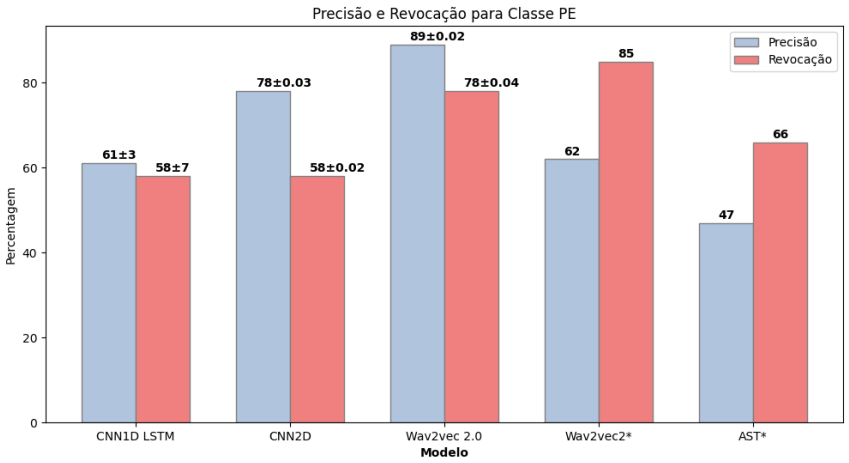



Figura 31 – Cenário Closed-set Spotify-B classificação binária para a classe PE

Os resultados da classificação binária usando o modelo Wav2vec 2.0 XLSR multilingual apresentaram desempenho superior em relação ao modelo CNN1D LSTM. No cenário *closed-set*, o F1-score geral atingiu 83%. Foi notável também a redução significativa no desvio padrão, havendo menos variações entre os valores das classes correspondentes, conforme apresentado na Tabela 21. Comparativamente aos demais modelos, o Wav2vec 2.0 XLSR foi capaz de capturar representações contextualmente relevantes que são cruciais para a distinção entre os sotaques de Pernambuco e São Paulo. Além disso, o modelo demonstrou a capacidade de aprender características mais complexas, características intrínsecas a cada sotaque.

Com o objetivo de investigar o impacto da variação na quantidade de locutores durante o treinamento, foi realizada uma redução na quantidade de locutores, utilizando de 9 a 11

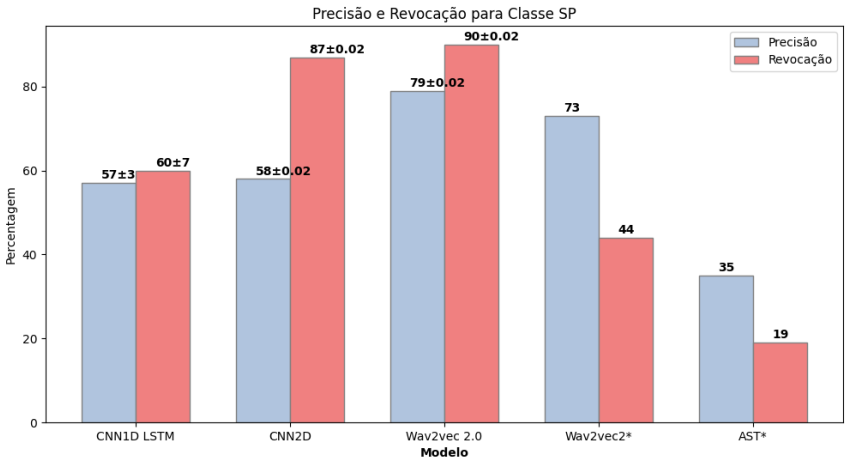



Figura 32 – Cenário Closed-set Spotify-B classificação binária para a classe SP

locutores para as classes PE e SP no treinamento, enquanto 5 locutores foram empregados para o conjunto de desenvolvimento. Observou-se que, com essa redução de locutores e segmentos de áudio durante o treinamento (1135 para PE e 945 para SP), houve uma queda significativa no desempenho do modelo, resultando em um F1-score de 63%. Observa-se portanto, que a adição de variabilidade de locutores e não somente de quantidade de trechos ou segmentos de áudio contribuem para um melhor desempenho dos modelos utilizados.

No caso do modelo Audio Spectrogram Transformer (AST), como mencionado anteriormente, realizou-se o *transfer learning* ao longo de 20 épocas, com a utilização de 1135 áudios para a classe PE e 945 para SP. O modelo alcançou um F1-score de apenas 24% no conjunto de teste. Observou-se a necessidade de uma quantidade maior de épocas de treinamento e segmentos de áudio por classe para melhorar o desempenho do modelo. Além disso, em comparação com o Wav2vec2* ( utilizando uma quantidade menor de áudios), mesmo com uma quantidade semelhante de áudios, o AST apresentou um desempenho significativamente inferior. Isso se deve, em parte, ao fato de o modelo AST ser pré-treinado apenas para o idioma inglês. É importante mencionar também que em comparação ao artigo original desse modelo, não foram utilizadas técnicas de aumento de dados e mixagem de duas amostras de áudio, o que pode ter contribuído também para redução do desempenho, tendo em vista, que essas técnicas podem auxiliar na capacidade de generalização dos modelos.

Outro fator a ser considerado é que, embora o AST também faça uso do Transformer e seja capaz de capturar dependências nos espectrogramas, a conversão do áudio para espectrograma e a divisão em *patches* podem resultar em alguma perda de informação temporal, o que pode afetar a capacidade do modelo de compreender o contexto completo referente a determinado sotaque. Além disso, o modelo parece ter dificuldade em generalizar na classificação binária. Por outro lado, os resultados obtidos com o Wav2vec 2.0 XLSR, apontam que o modelo consegue capturar nuances e detalhes mais específicos do sinal de áudio bruto, o que pode contribuir para

uma melhor performance do modelo.

No cenário multiclasse com três classes (PE, SP e MG), os resultados de todos os modelos indicaram uma dificuldade na classificação com mais de duas classes. Conforme apresentado na Tabela 22, os modelos conseguiram resultados para a classe SP, mas tiveram poucos acertos para a classe MG. Notavelmente, a CNN 2D mostrou uma taxa de F1-score considerável (71%) para a classe PE em comparação com as outras classes, indicando que o modelo conseguiu aprender características distintivas dos espectrogramas para essa classe, com taxa de revocação de 77% ( Figura 40 no Apêndice). Apesar da classe MG ser mais desafiadora, a CNN 2D conseguiu capturar algumas características, mesmo com um desempenho inferior (39%) em comparação com os demais modelos, como a CNN1D LSTM, Wav2vec2 e AST.

| classe | CNN 1D LSTM | CNN 2D | Wav2vec2 | AST |
|---|---|---|---|---|
| PE | - | 71% | 24% | 47% |
| SP | 55% | 35% | 52% | 27% |
| MG | 12% | 39% | - | - |
| F1-score Geral | 22% | 48% | 25% | 24% |

Tabela 22 – Cenário Closed-set - Multiclasse: F1-score dos modelos para cenário multiclasse ( PE,SP, MG )

A partir dos resultados reportados para três classes, percebemos que diferentemente do trabalho do Tostes (2022) para fala preparada, a CNN1D LSTM não obteve êxito na tarefa de classificação para fala espontânea. Como é possível observar na Tabela 22 e as métricas de precisão e revocação (Figura 40 no Apêndice), o modelo não consegue distinguir a classe PE. Isso evidencia os desafios na classificação da fala espontânea, especialmente em cenários com mais de duas classes. Além disso, observa-se uma dificuldade na classificação de amostras para o sotaque mineiro, tanto utilizando o Wav2vec2 quanto o AST. Uma das hipóteses para essa dificuldade é a grande variedade de dialetos presentes dentro do próprio estado de Minas Gerais. Além disso, como Minas Gerais faz fronteira com vários outros estados brasileiros, é possível que o sotaque mineiro seja influenciado por essas regiões vizinhas, o que pode complicar ainda mais sua classificação.

Conforme os resultados apresentados, os modelos provavelmente estão aprendendo padrões espúrios que não se referem de fato às características dos sotaques em questão. É possível observar também que a tarefa é desafiadora, mesmo com os dados balanceados seja na quantidade de segmentos de áudios ou variedade de locutor. Portanto para uma melhor investigação sobre o que de fato os modelos estão aprendendo, é necessário uma análise por parte de um linguista acerca dos padrões de entonação e ritmo da fala do sotaque, além de considerar nos metadados dos arquivos de áudio, informações sobre variações sociolinguísticas presentes no áudio por exemplo, como idade, gênero, nível educacional que influenciam o sotaque.

### 4.2.2 Cross-dataset: Spotify-B e CORAA ASR

No cenário *cross-dataset*, os resultados utilizando o modelo CNN1D LSTM confirmaram as conclusões apontadas por Batista (2019), indicando que o modelo apresenta dificuldades para generalizar em outros conjuntos de dados. Conforme mostrado na Tabela 23, as predições do modelo favoreceram a classe PE, porém com uma baixa taxa de precisão ($50 \pm 1\%$) como pode ser observado na Figura 33. Por outro lado, a classe SP apresentou uma melhor precisão, porém com uma taxa de revocação baixa ($11 \pm 4\%$).

É possível observar que no cenário *cross-dataset* a maioria dos modelos tem um comportamento oposto ao *closed-set*. Como pode ser observado na Figura 33 e 34, no cenário *cross-dataset* os modelos apresentam uma taxa de revocação maior para a classe PE e uma precisão mais acentuada para a classe SP. Provavelmente isso é decorrente das diferenças entre os conjuntos de dados, como características acústicas, distribuição dos sotaques, por exemplo, mas é necessário uma melhor investigação para compreender o que realmente os modelos estão aprendendo.

Comparando com o cenário com menor quantidade de falantes (Spotify-A), observou-se uma melhoria significativa na capacidade de detecção para a classe PE, com uma taxa de revocação consideravelmente mais elevada ($96 \pm 4\%$) em comparação com o Spotify-A ($10 \pm 3\%$). Isso sugere que o modelo identifica corretamente o sotaque PE.

| classe | CNN 1D LSTM | CNN 2D | Wav2vec2 | AST |
|---|---|---|---|---|
| PE | $66 \pm 9\%$ | $50 \pm 21\%$ | 80% | 32% |
| SP | $19 \pm 6\%$ | $21 \pm 0.07\%$ | 70% | 48% |
| F1-score Geral | $43 \pm 3\%$ | $35 \pm 14\%$ | 75% | 40% |

Tabela 23 – Cenário Cross-dataset - Classificação Binária: F1-score dos modelos para classificação binária ( PE, SP)

O F1-score geral para a CNN1D LSTM apresentou uma melhora para 43%, conforme mostrado na Tabela 23. Essa melhoria é atribuída ao aumento na precisão e revocação do sotaque PE. No entanto, a classificação de diferentes variações ainda se mostra um desafio no cenário *cross-dataset*, mesmo com o aumento na quantidade de áudios e locutores. Por outro lado, para a CNN2D, os resultados mostram uma queda de desempenho do modelo, que confunde a classe PE com SP e vice-versa, resultando em um F1-score geral de $35 \pm 14\%$.

Os resultados no modelo Wav2vec2 XLSR multilingual indicaram que modelos pré-treinados com áudios em pt-BR alcançam desempenho considerável na classificação binária de sotaques, obtendo um F1-score de 75% no CORAA-ASR, conforme mostrado na Tabela 23. Na Figura 33 e 34, observa-se um comportamento, oposto também entre precisão e revocação das classes. A alta precisão para a classe SP (99%), sugere que o Wav2vec identifica bem as características para esse sotaque utilizando o CORAA ASR, porém a baixa revocação indica que o modelo tem dificuldade em capturar todas as amostras verdadeiras de SP, provavelmente

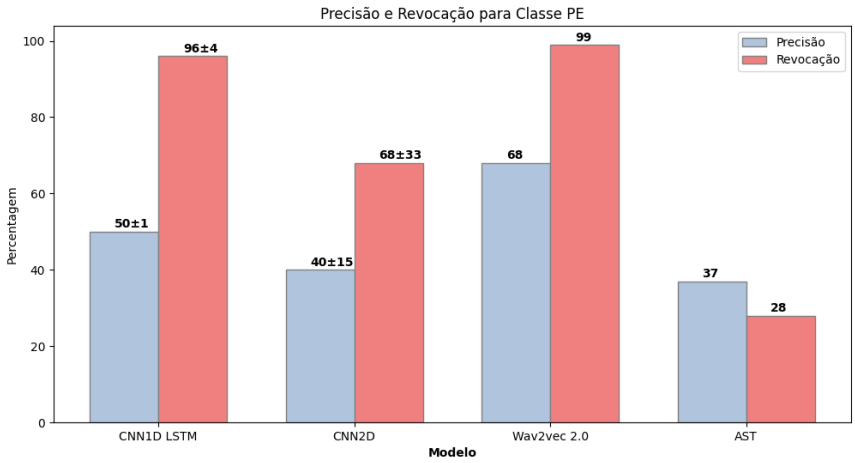



Figura 33 – Cenário Cross-dataset Spotify-B classificação binária - Classe PE

esses resultados podem estar relacionados as diferenças na complexidade dos sotaques presentes nos *datasets*. Por outro lado, o modelo Audio Spectrogram Transformer obteve um desempenho inferior aos demais modelos, com um F1-score de 40% e apresenta um comportamento diferente dos demais. Enquanto nos demais modelos para o cenário multiclasse a taxa de revocação para a classe PE é acentuada no *cross-dataset*, o AST apresenta uma uma taxa um pouco maior para precisão e o contrário ocorre para a classe SP.

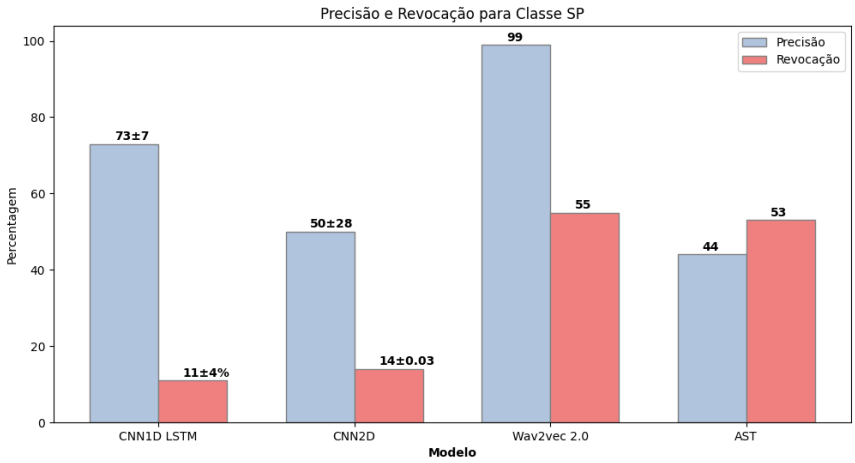



Figura 34 – Cenário Cross-dataset Spotify-B classificação binária - Classe SP

No cenário multiclasse (PE, SP, MG), conforme apresentado na Tabela 24, o modelo Wav2vec2 demonstrou uma melhor classificação para a classe PE, enquanto a CNN1D LSTM obteve melhores resultados para a classe SP. O F1-score de 68% alcançado pelo modelo Wav2vec2 na classificação do sotaque PE indica que ele foi capaz de capturar características distintivas dessa classe. O modelo apresenta dificuldades para classificar as demais classes, sobretudo a classe MG ( Figura 42 no Apêndice), em que não obtém nenhum acerto. Percebe-se também que

ao avaliar o modelo em um outro *dataset*, a capacidade de generalização reduz significativamente, tendo em vista também as características acústicas distintas entre o Spotify Podcasts e CORAA ASR, como mencionado anteriormente.

Para o sotaque SP, a CNN1D LSTM teve um desempenho notavelmente superior, alcançando 52% de F1-score em comparação com a CNN2D, por exemplo, o que sugere uma melhor capacidade de capturar nuances específicos do sotaque paulista. No entanto, para a classe MG, nenhum modelo se destacou significativamente, indicando um desafio para os modelos na classificação desse sotaque.

Em desempenho geral, percebe-se que o Wav2vec2 foi o que obteve F1-score maior com 32% . Mesmo aplicando técnicas de balanceamento entre as classes, atribuindo pesos específicos durante o treinamento, percebe-se que é necessário uma quantidade mais robusta de áudios para cada uma das classes. Uma alternativa para o modelo multilingual seria aplicar técnicas de aumento de dados diretamente nas entradas, como apresentado no trabalho do Zuluaga *et al.* (2023), que pode contribuir para uma melhora no desempenho do modelo. Para o AST, faz-se necessário um estudo mais detalhado sobre as características particulares que o modelo está aprendendo do espectrograma para cada classe e assim realizar ajustes no treinamento específico para cada regionalismo.

| classe | CNN 1D LSTM | CNN 2D | Wav2vec2 | AST |
|---|---|---|---|---|
| PE | – | – | 68% | 39% |
| SP | 52% | 17% | 26% | – |
| MG | 13% | 14% | – | 33% |
| F1-score Geral | 22% | 10% | 32% | 24% |

Tabela 24 – Cenário Cross-dataset - Multiclasse : F1-score dos modelos para cenário multiclasse ( PE, SP, MG)

A partir dos resultados apresentados para o cenário *closed-set* e *cross-dataset* observa-se que as dificuldades para classificação não somente ocorrem quando são utilizados outros *datasets* para validação, como apontado pela Batista (2019), mas também em um mesmo *dataset*, sobretudo quando não ocorre a presença de um mesmo locutor entre conjunto de treinamento, validação e teste. É importante mencionar também os desafios para processamento de fala espontânea, desde características acústicas dos segmentos de áudios entre os *datasets* a características distintas dos sotaques brasileiros.

É possível observar também que, mesmo com a fala preparada, os resultados reportados com o Braccent indicam a necessidade de uma maior quantidade de áudios representativos para cada sotaque. Além disso, deve-se considerar uma divisão mais precisa por parte dos linguistas. É igualmente importante avaliar a relevância da validação cruzada, dado que pode ocorrer vazamento de locutores entre os *folds*, o que distancia os resultados de um cenário realista.

Os resultados apresentados reforçam o desafio de classificar sotaques de um mesmo

país. A maioria dos trabalhos na literatura, como o estudo de Chitkara *et al.* (2022), utiliza variações de idiomas como inglês e espanhol de diferentes países, e não apenas de um único país. Isso destaca a complexidade adicional ao se tentar classificar sotaques brasileiros, como neste trabalho.

## 4.3 Resultados Explicabilidade

Para uma melhor compreensão sobre as características consideradas pela rede convolucional 2D durante a classificação foi utilizado o Grad-CAM.Esse modelo foi escolhido, pois trabalho com dados em formato bidimensional, enquanto a CNN1D LSTM combina convoluções unidimensionais que podem ser menos visualmente intuitva.

Na Figura 35, são apresentadas as predições corretas para a classe PE e SP utilizando o conjunto de testes do Spotify-B.Na Figura 35(a) que representa a classe PE, foram obtidos padrões de ativação em áreas distintas. Na Figura 35(b), referente a classe SP, o Grad-CAM mostra cores mais quentes no início, indicando que essas regiões foram mais influentes para a decisão do modelo. É possível observar que para a classe PE, o modelo apresenta alguma confiança na classificação dos outros sotaques (SP e MG), embora com menor probabilidade em comparação a classe PE. Isso denota que o modelo também leva em consideração essas regiões para classificação das demais classes.

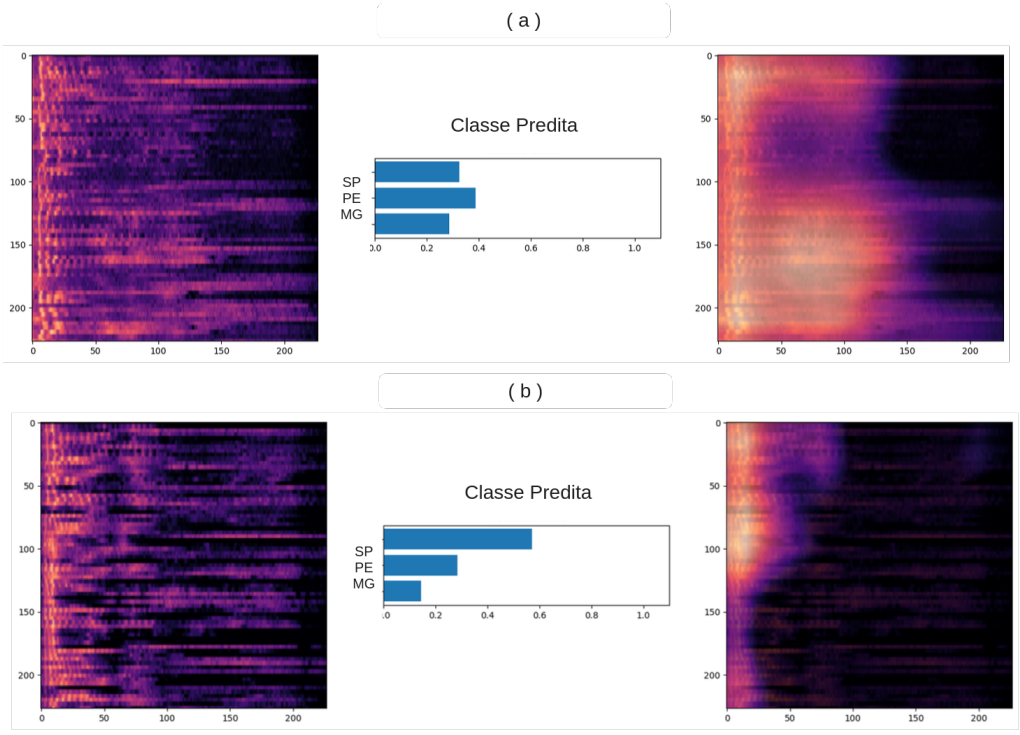



Figura 35 – Resultado de explicabilidade de dois exemplos de teste do Spotify-B classificados corretamente pela CNN2D, incluindo o espectrograma de entrada, os scores gerados pelo modelo e a seguir os mapas de calor gerados pelo Grad-CAM para exemplos das classes PE (a) e SP (b)

No espectrograma, as cores mais escuras (próximas ao preto) geralmente indicam ausência ou níveis mais baixos de energia em determinadas frequências. Com o Grad-CAM, foi possível observar que o modelo não leva em consideração essas regiões, sugerindo, portanto, que ele não aprende informações irrelevantes ou que não contribuem para a distinção dos sotaques. Na Figura 36, são apresentadas as conclusões mencionadas anteriormente, mostrando padrões de ativação distintos para a classe PE. Na Figura 37, para a classe SP pode ser observado que o mapa de calor se concentra nas frequências menores do espectrograma.

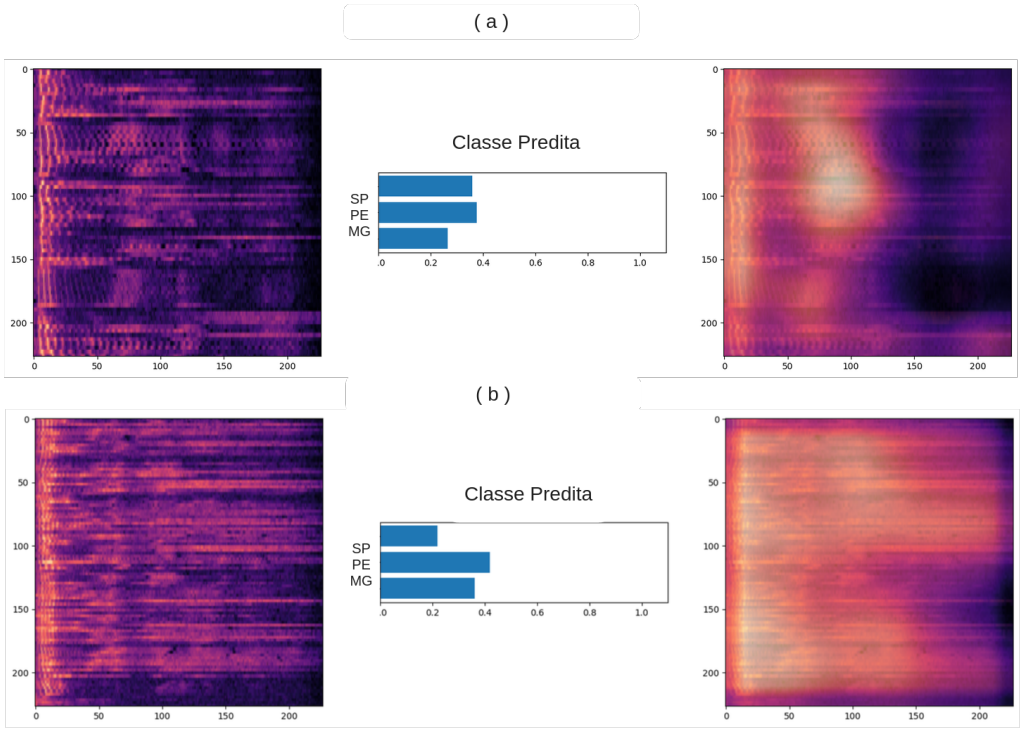



Figura 36 – Predicões Corretas Classe PE Spotify-B: Mapas de calor obtidos utilizando o Grad-CAM para as predições corretas da classe PE no conjunto de teste do Spotify-B

Para as predições erradas, como apresentado na Figura 38(a), foi possível perceber que o modelo utilizou as primeiras faixas de frequência para classificar a classe PE como SP. Para o subconjunto do CORAA ASR, o resultado da CNN2D foi apenas de 11%, como reportado anteriormente. Assim, a maioria das predições foi incorreta. Na Figura 38(b), observa-se que o modelo confunde SP com PE e vice-versa.

O modelo erra praticamente todas as predições para a classe MG, conforme mostrado na Figura 39(b). Observa-se que o modelo identificou que espectrogramas mais completos, com menor quantidade de áreas de baixa energia, correspondem à classe PE. Por outro lado, espectrogramas com maior concentração nas primeiras faixas ou frequências mais baixas são associados à classe SP.

Conforme descrito, percebe-se que para a classe PE, o modelo analisa diferentes áreas do espectrograma, enquanto para SP considera frequências mais baixas. Para classificações corretas o Grad-CAM concentrou-se em áreas mais específicas enquanto para predições erradas foi

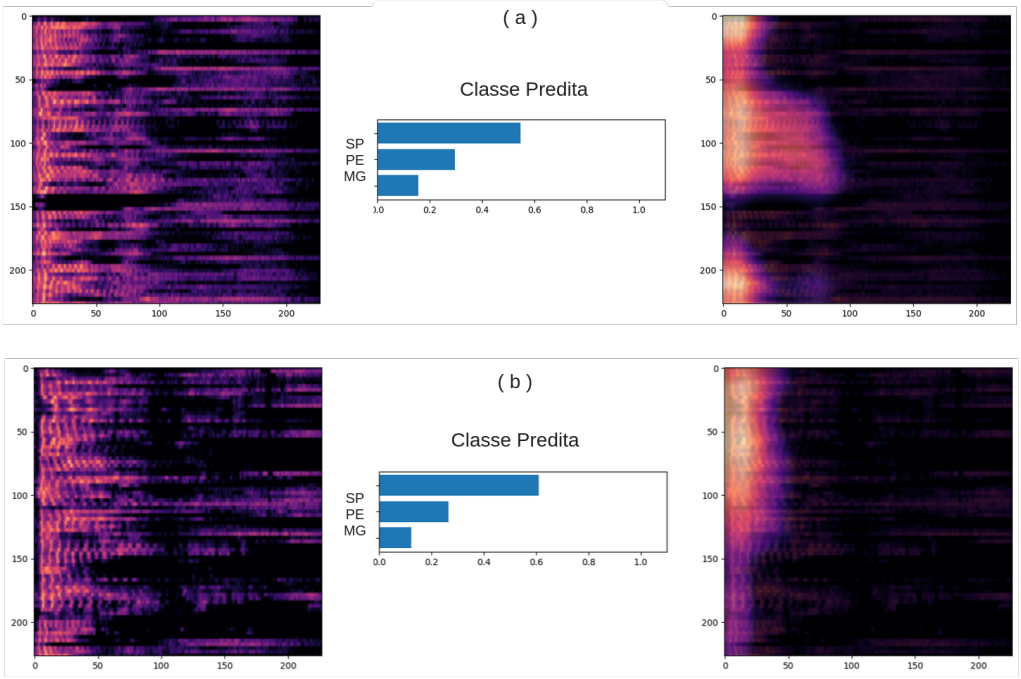



Figura 37 – Predicões Corretas Classe SP Spotify-B:Mapas de calor obtidos utilizando o Grad-CAM para as predições corretas da classe SP no conjunto de teste do Spotify-B

possível observar mapas de calor gerados em áreas distintas ou ao longo das faixas frequência e do tempo. É crucial destacar que, mesmo com os mapas de calor, para a tarefa de classificação de sotaques, é necessário realizar uma análise dessas variações linguísticas considerando diferentes frequências e intensidades presentes, que podem ser características da fala por estado, como pronúncias específicas associadas à classe PE, por parte de um linguista. Analisar a energia em diferentes bandas de frequência, por exemplo, pode auxiliar na identificação das características específicas de um sotaque.

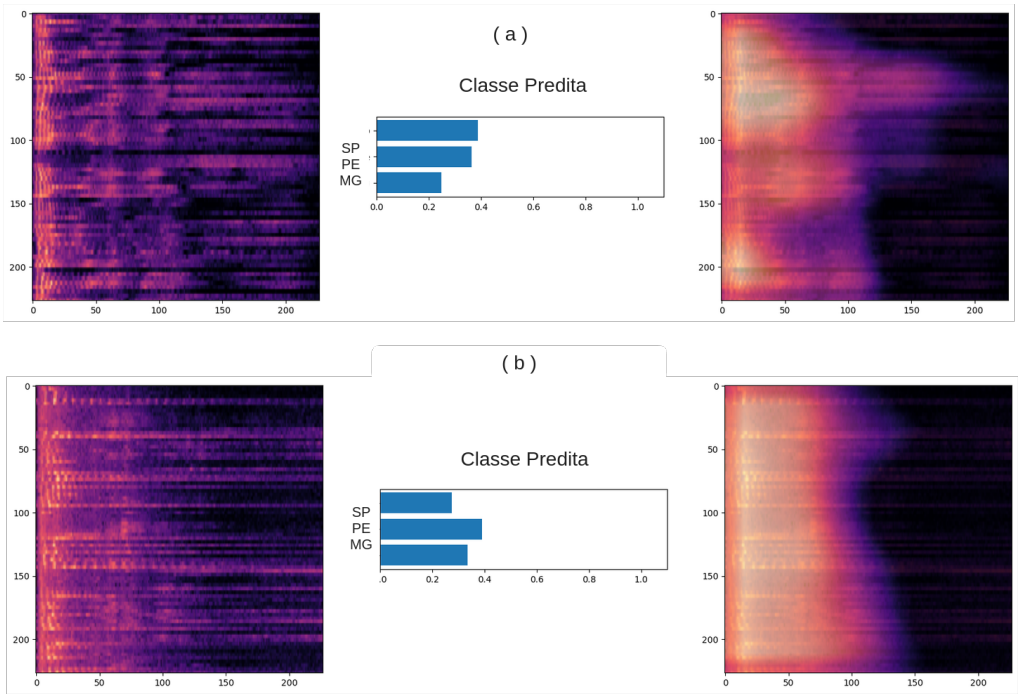



Figura 38 – Grad-CAM classe PE: Predição errada para a classe PE utilizando imagens do subconjunto de teste do Spotify-B

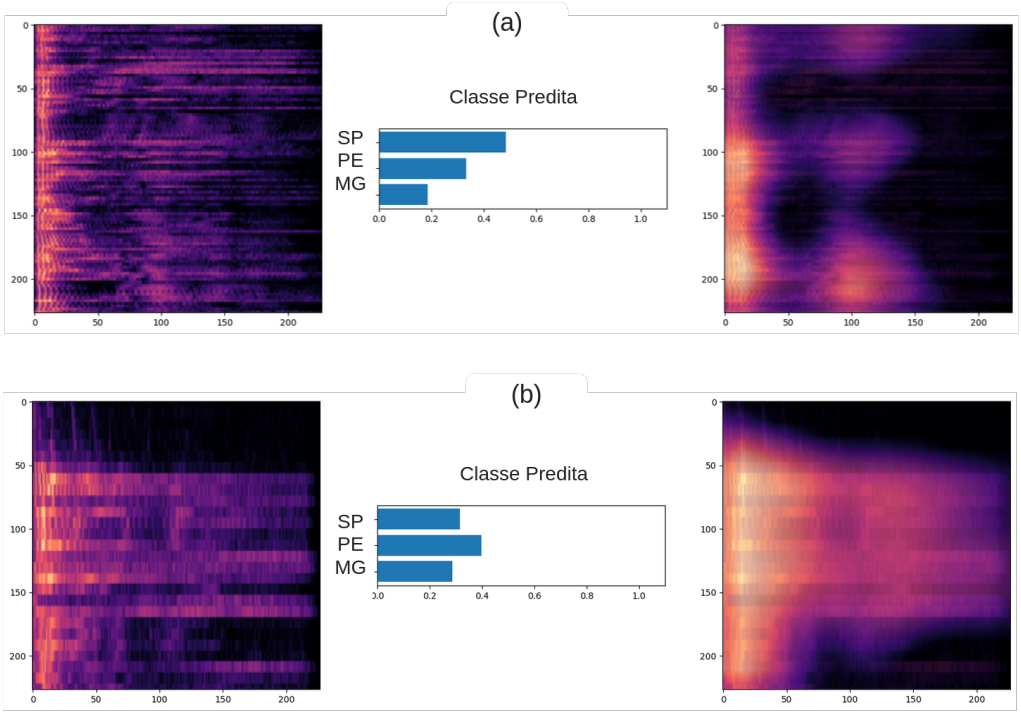



Figura 39 – Predicões Incorretas para o CORAA ASR e Spotify-B. (a) O modelo classificou PE como SP – Spotify-B. (b) O modelo classificou MG como PE – CORAA ASR

CAPÍTULO

# 5

# CONCLUSÃO E TRABALHOS FUTUROS

A classificação de sotaques para o Português Brasileiro é uma tarefa desafiadora e que pode auxiliar na área de reconhecimento automático de fala. Partindo do objetivo inicial em projetar e avaliar modelos, para este trabalho foram utilizadas abordagens distintas para a tarefa de classificação, englobando Redes Neurais Convolucionais 1D com LSTM, CNN2D, Wav2vec2 XLSR e Audio Spectrogram Transformer.

Os resultados apontaram que os modelos apresentaram dificuldades para distinguir as variações linguísticas quando os locutores com os quais realizamos o teste não estão presentes no treinamento, mesmo no cenário *closed-set*. Resultados superiores são obtidos apenas com partições aleatórias ou *cross-validation* em que há presença de segmentos de um mesmo locutor nos conjuntos de treinamento, desenvolvimento e teste.

No cenário com poucos locutores utilizando o Spotify-A, os resultados demonstraram que a presença de 1 a 3 locutores por sotaque não é suficiente para uma boa classificação com F1-score baixo, mesmo no cenário *closed-set*. O mesmo ocorreu no *dataset* Braccent, o qual contém áudios com fala não espontânea e desbalanceamento tanto na quantidade de locutores quanto no volume de áudios entre as classes. Com mais locutores (Spotify-B), os resultados apontaram uma melhora significativa dos modelos e menor desvio padrão. Para a classificação binária, aplicar *fine-tuning* em modelos pré-treinados como Wav2vec2 XLSR foi mais eficaz e permitiu aprender aspectos e características mais complexas referentes aos sotaques de Pernambuco e São Paulo, tanto no cenário *closed-set* quanto no *cross-dataset*, conforme apresentado na Tabela 23.

Em termos dos modelos utilizados, nossos resultados mostram que a CNN2D pode ser mais eficaz para aprender características espaciais (frequência e tempo) em comparação a CNN1D LSTM. Contudo, treinar modelos do zero, como CNN1D LSTM e CNN2D, requer uma quantidade maior de épocas de treinamento e também uma quantidade maior de dados. Ainda, o modelo Audio Spectrogram Transformer não apresentou boa transferência de aprendizado, sendo necessário mais estudos para entender como melhor utilizá-lo. Percebeu-se, também que

para a tarefa de classificação funcionar de forma eficiente tanto no cenário *closed-set* quanto no cenário *cross-dataset* é necessário um ajuste constante de hiperparâmetros de otimização conforme o modelo a ser utilizado.

Com a explicabilidade utilizando o Grad-CAM na CNN 2D, é possível concluir que o modelo utilizou as primeiras faixas de frequências dos espectrograma para classificar SP e diferentes áreas para classificação do sotaque Pernambucano. Além disso, em geral os exemplos nos quais a rede tem maior incerteza apresentam mapas de importância espalhados por todo o sinal. Isso pode levar a crer que os casos de sucesso necessita o aprendizado padrões espaço-temporais pontuais. Para confirmar essa interpretação é necessário uma análise do espectrograma com ferramentas para verificar melhor aspectos prosódicos da fala com auxílio de linguistas.

O *dataset* Spotify Podcasts possui bom potencial para ser usado como forma de investigar a tarefa de classificação de sotaques devido a diversidade e quantidade de locutores disponível. No entanto, apresenta desafios, pois além da presença de sobreposições de fala, tem-se o fundo musical e faz-se necessário um pré-processamento adequado, como mencionado em capítulos anteriores.

A hipótese inicial para este trabalho é que um modelo com mecanismos de atenção com parâmetros pré-treinados conseguisse generalizar para outros *datasets*, contudo os resultados apontaram que para a classificação multiclasse a hipótese não é verdadeira sendo necessária ainda o emprego de outras técnicas ou abordagens e uma maior quantidade de dados para englobar as características de um sotaque específico. Os resultados obtidos no decorrer deste trabalho mostraram que o a classificação de variações linguísticas ainda apresenta desafios não somente em cenários *cross-dataset* mas também *closed-set* e é um problema em aberto para classificação multiclasse (MATOS *et al.*, 2024).

As variações linguísticas consistem em padrões de pronúncia e fala características de lugares distintos dentro de uma mesma língua materna e devido aos nuances distintos que cada variedade possui, identificá-las de forma automática é ainda um desafio. Ao desenvolver classificadores de sotaques, notamos a importância de validar em cenários mais próximos do real, assumindo partições de treinamento e teste sem repetição de locutor e validação *cross-dataset*, além do uso de modelos pré-treinados. O volume, diversidade e balanceamento dos diferentes sotaques também tem um papel fundamental na qualidade dos modelos treinados.

Assim, como trabalhos futuros, tem-se ampliar o conjunto de dados com uma maior diversidade de locutores e outras variedades linguísticas aliada à análise dos segmentos de áudios por parte de um linguista. Aplicar técnicas de aumento de dados no modelo do Audio Spectrogram Transformer e Wav2vec2, por exemplo, podem auxiliar no melhoria dos resultados. Além disso, o aprendizado contrastivo com uso de modelos pré-treinados podem facilitar na distinção dos sotaques, bem como, o uso de *embeddings* de locutores como entrada em um modelo. Ainda, o uso de melhores técnicas de interpretabilidade para analisar características da fala para cada sotaque em específico é também um tema com potencial para futuros estudos.

# REFERÊNCIAS

AGGARWAL, C. C. **Neural Networks and Deep Learning**: A textbook. Cham: Springer, 2018. 497 p. ISBN 978-3-319-94462-3. Citado nas páginas 37, 40, 47, 51 e 52.

ALAMMAR, J. **The Illustrated Transformer**. 2018. Disponível em: <https://jalammar.github.io/illustrated-transformer/>. Citado na página 57.

ALBAWI, S.; MOHAMMED, T. A.; AL-ZAWI, S. Understanding of a convolutional neural network. In: **2017 International Conference on Engineering and Technology (ICET)**. [S.l.: s.n.], 2017. p. 1–6. Citado na página 47.

ALCAIM, A.; SOLEWICZ, J. A.; MORAES, J. Antônio de. Freqüência de ocorrência dos fones e listas de frases foneticamente balanceadas no português falado no rio de janeirolt;br /38;gt;doi: 10.14209/jcis.1992.2. **Journal of Communication and Information Systems**, v. 7, n. 1, Jun. 2015. Disponível em: <https://jcis.sbrt.org.br/jcis/article/view/166>. Citado na página 24.

ALMEIDA, D. R. de. Comparação entre modelos com diferentes abordagens para classificação de sotaques brasileiros. Universidade Federal de Campina Grande, 2022. Citado nas páginas 67 e 68.

AMODEI, D.; ANUBHAI, R.; BATTENBERG, E.; CASE, C.; CASPER, J.; CATANZARO, B.; CHEN, J.; CHRZANOWSKI, M.; COATES, A.; DIAMOS, G.; ELSEN, E.; ENGEL, J. H.; FAN, L.; FOUGNER, C.; HAN, T.; HANNUN, A. Y.; JUN, B.; LEGRESLEY, P.; LIN, L.; NARANG, S.; NG, A. Y.; OZAIR, S.; PRENGER, R.; RAIMAN, J.; SATHEESH, S.; SEETAPUN, D.; SENGUPTA, S.; WANG, Y.; WANG, Z.; WANG, C.; XIAO, B.; YOGATAMA, D.; ZHAN, J.; ZHU, Z. Deep speech 2: End-to-end speech recognition in english and mandarin. **CoRR**, abs/1512.02595, 2015. Disponível em: <http://arxiv.org/abs/1512.02595>. Citado na página 60.

ARRAS, L.; ARJONA-MEDINA, J.; WIDRICH, M.; MONTAVON, G.; GILLHOFER, M.; MÜLLER, K.-R.; HOCHREITER, S.; SAMEK, W. Explaining and interpreting lstms. In: **Explainable ai: Interpreting, explaining and visualizing deep learning**. [S.l.]: Springer, 2019. p. 211–238. Citado na página 53.

BA, J. L.; KIROS, J. R.; HINTON, G. E. Layer normalization. **arXiv preprint arXiv:1607.06450**, 2016. Citado na página 58.

BAEVSKI, A.; ZHOU, Y.; MOHAMED, A.; AULI, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. **Advances in neural information processing systems**, v. 33, p. 12449–12460, 2020. Citado nas páginas 60 e 61.

BAHDANAU, D.; CHO, K.; BENGIO, Y. Neural machine translation by jointly learning to align and translate. **arXiv preprint arXiv:1409.0473**, 2014. Citado na página 56.

BANK, D.; KOENIGSTEIN, N.; GIRYES, R. Autoencoders. **CoRR**, abs/2003.05991, 2020. Disponível em: <https://arxiv.org/abs/2003.05991>. Citado na página 49.

BATISTA, N. *et al.* Detecçao automática de sotaques regionais brasileiros: A importância da validaçao cross-datasets. **Anais do XXXVI Simpósio Brasileiro de Telecomunicaçoes e Processamento de Sinais (SBrT)**, p. 939–944, 2018. Citado nas páginas 62, 66 e 67.

BATISTA, N. A. R. **Estudo sobre Identificação Automática de Sotaques Regionais Brasileiros baseada em Modelagens Estatísticas e Técnicas de Aprendizado de Máquina**. 138 p. Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas, 2019. Citado nas páginas 24, 25, 33, 62, 65, 66, 67, 68, 70, 93 e 95.

BAUM, L. E.; EAGON, J. A. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. 1967. Citado na página 23.

BECHER, A.; PONTI, M. Optimization matters: Guidelines to improve representation learning with deep networks. In: **Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional**. Porto Alegre, RS, Brasil: SBC, 2021. p. 595–606. ISSN 2763-9061. Disponível em: <https://sol.sbc.org.br/index.php/eniac/article/view/18287>. Citado na página 40.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. [S.l.]: Springer, 2006. Citado na página 38.

CALLOU, D.; LEITE, Y. **Inicia < o ^ fonŽtica e ^ fonologia**. [S.l.]: Zahar, 1990. Citado na página 33.

CASANOVA, E. **Síntese de fala aplicada à geração de conjunto de dados para reconhecimento automático de fala**. Tese (Doutorado) — USP-Instituto de Ciências Matemáticas e de Computação, São Carlos, 2022. Citado na página 56.

CASANOVA, E.; WEBER, J.; SHULBY, C. D.; JUNIOR, A. C.; GÖLGE, E.; PONTI, M. A. Yourtts: Towards zero-shot multi-speaker tts and zero-shot voice conversion for everyone. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2022. p. 2709–2720. Citado na página 23.

CAUCHY, A. Methode generale pour la resolution des systemes d'equations simultanees. **C.R. Acad. Sci. Paris**, v. 25, p. 536–538, 1847. Disponível em: <https://cir.nii.ac.jp/crid/1573387450834953216>. Citado na página 39.

CAVALLARI, G. B.; RIBEIRO, L. S.; PONTI, M. A. Unsupervised representation learning using convolutional and stacked auto-encoders: a domain and cross-domain feature space analysis. In: IEEE. **2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)**. [S.l.], 2018. p. 440–446. Citado na página 48.

CHAN, W.; JAITLY, N.; LE, Q.; VINYALS, O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In: **2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2016. p. 4960–4964. Citado na página 24.

CHITKARA, P.; RIVIERE, M.; COPET, J.; ZHANG, F.; SARAF, Y. Pushing the performances of asr models on english and spanish accents. **arXiv preprint arXiv:2212.12048**, 2022. Citado nas páginas 64, 65 e 96.

CHOLLET, F. **Deep Learning with Python**. [S.l.]: Manning, 2017. ISBN 9781617294433. Citado na página 51.

CHRISTENSEN, M. G. **Introduction to audio processing**. [S.l.]: Springer, 2019. Citado na página 29.

CONNEAU, A.; KHANDELWAL, K.; GOYAL, N.; CHAUDHARY, V.; WENZEK, G.; GUZMÁN, F.; GRAVE, E.; OTT, M.; ZETTLEMOYER, L.; STOYANOV, V. Unsupervised cross-lingual representation learning at scale. **arXiv preprint arXiv:1911.02116**, 2019. Citado na página 78.

CORBETTA, M.; SHULMAN, G. L. Control of goal-directed and stimulus-driven attention in the brain. **Nature reviews neuroscience**, Nature Publishing Group, v. 3, n. 3, p. 201–215, 2002. Citado na página 55.

DEHAK, N.; KENNY, P.; DEHAK, R.; DUMOUCHEL, P.; OUELLET, P. Front-end factor analysis for speaker verification. **Audio, Speech, and Language Processing, IEEE Transactions on**, v. 19, p. 788 – 798, 06 2011. Citado na página 63.

DEMPSTER, J. Chapter six - signal analysis and measurement. In: **The Laboratory Computer**. London: Academic Press, 2001, (Biological Techniques Series). p. 136–171. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780122095511500398>. Citado na página 32.

DENG, K.; CAO, S.; MA, L. Improving accent identification and accented speech recognition under a framework of self-supervised learning. **arXiv preprint arXiv:2109.07349**, 2021. Citado na página 24.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S. *et al.* An image is worth 16x16 words: Transformers for image recognition at scale. **arXiv preprint arXiv:2010.11929**, 2020. Citado na página 59.

DOSOVITSKIY, A.; BEYER, L.; KOLESNIKOV, A.; WEISSENBORN, D.; ZHAI, X.; UNTERTHINER, T.; DEHGHANI, M.; MINDERER, M.; HEIGOLD, G.; GELLY, S.; USZKOREIT, J.; HOULSBY, N. An image is worth 16x16 words: Transformers for image recognition at scale. **CoRR**, abs/2010.11929, 2020. Disponível em: <https://arxiv.org/abs/2010.11929>. Citado na página 61.

DUBEY, S. R.; SINGH, S. K.; CHAUDHURI, B. B. A comprehensive survey and performance analysis of activation functions in deep learning. **CoRR**, abs/2109.14545, 2021. Disponível em: <https://arxiv.org/abs/2109.14545>. Citado na página 41.

____. Activation functions in deep learning: A comprehensive survey and benchmark. **Neurocomputing**, Elsevier, 2022. Citado nas páginas 41, 43 e 44.

EMRESOY, M. K.; EL-JAROUDI, A. Iterative instantaneous frequency estimation and adaptive matched spectrogram. **Signal Processing**, v. 64, n. 2, p. 157–165, 1998. ISSN 0165-1684. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0165168497001837>. Citado na página 32.

FAN, Z.; LI, M.; ZHOU, S.; XU, B. Exploring wav2vec 2.0 on speaker verification and language identification. **arXiv preprint arXiv:2012.06185**, 2020. Citado na página 60.

GARDNER, M.; DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. **Atmospheric Environment**, v. 32, n. 14, p. 2627–2636, 1998. ISSN 1352-2310. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1352231097004470>. Citado na página 38.

GERS, F. A.; SCHMIDHUBER, J. Recurrent nets that time and count. In: IEEE. **Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium**. [S.l.], 2000. v. 3, p. 189–194. Citado na página 53.

GHOJOGH, B.; GHODSI, A. Attention mechanism, transformers, bert, and gpt: Tutorial and survey. OSF Preprints, 2020. Citado na página 58.

GIANNAKOPOULOS, T.; PIKRAKIS, A. **Introduction to audio analysis: a MATLAB® approach**. [S.l.]: Academic Press, 2014. Citado na página 29.

GONÇALVES, S. C. L. Projeto alip (amostra linguística do interior paulista) e banco de dados iboruna: 10 anos de contribuição com a descrição do português brasileiro. **Estudos Linguísticos (São Paulo. 1978)**, v. 48, n. 1, p. 276–297, 2019. Citado na página 71.

GONG, Y.; CHUNG, Y.-A.; GLASS, J. Ast: Audio spectrogram transformer. **arXiv preprint arXiv:2104.01778**, 2021. Citado nas páginas 61 e 62.

GOODFELLOW, I. J.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, MA, USA: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado nas páginas 37, 39, 43, 44 e 45.

_____. **Deep Learning**. Cambridge, MA, USA: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado nas páginas 48 e 49.

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**. [S.l.: s.n.], 2006. p. 369–376. Citado na página 63.

GREFF, K.; SRIVASTAVA, R. K.; KOUTNíK, J.; STEUNEBRINK, B. R.; SCHMIDHUBER, J. **LSTM: A Search Space Odyssey**. 2015. Cite arxiv:1503.04069Comment: 12 pages, 6 figures. Disponível em: <http://arxiv.org/abs/1503.04069>. Citado na página 52.

GRIS, L. R.; CASANOVA, E.; OLIVEIRA, F.; SOARES, A.; CANDIDO-JUNIOR, A. Desenvolvimento de um modelo de reconhecimento de voz para o português brasileiro com poucos dados utilizando o wav2vec 2.0. In: **Anais do XV Brazilian e-Science Workshop**. Porto Alegre, RS, Brasil: SBC, 2021. p. 129–136. ISSN 2763-8774. Citado na página 24.

GULATI, A.; QIN, J.; CHIU, C.-C.; PARMAR, N.; ZHANG, Y.; YU, J.; HAN, W.; WANG, S.; ZHANG, Z.; WU, Y. *et al.* Conformer: Convolution-augmented transformer for speech recognition. **arXiv preprint arXiv:2005.08100**, 2020. Citado na página 63.

HÄMÄLÄINEN, M.; ALNAJJAR, K.; PARTANEN, N.; RUETER, J. Finnish dialect identification: The effect of audio and text. **CoRR**, abs/2111.03800, 2021. Disponível em: <https://arxiv.org/abs/2111.03800>. Citado na página 24.

HAYKIN, S.; NETWORK, N. A comprehensive foundation. **Neural networks**, v. 2, n. 2004, p. 41, 2004. Citado na página 51.

HAYKIN, S. S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009. Citado nas páginas 36, 37, 38 e 39.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778. Citado na página 58.

HOAI, N. B. K.; OH, H.; YI, H. Traffic density classification using sound datasets: An empirical study on traffic flow at asymmetric roads. **IEEE Access**, PP, p. 1–1, 07 2020. Citado na página 32.

HORE, P.; CHATTERJEE, S. **A Comprehensive Guide to Attention Mechanism in Deep Learning for Everyone**. Analytics Vidhya, 2022. Disponível em: <https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/>. Citado na página 56.

ILARI, R.; BASSO, R. O português da gente: a língua que estudamos, a língua que falamos. **(No Title)**, v. 2, p. 167–168, 2009. Citado nas páginas 34 e 35.

JAIN, A.; UPRETI, M.; JYOTHI, P. Improved accented speech recognition using accent embeddings and multi-task learning. In: **Interspeech**. [S.l.: s.n.], 2018. p. 2454–2458. Citado nas páginas 24, 62, 63 e 65.

JANG, E.; GU, S.; POOLE, B. Categorical reparameterization with gumbel-softmax. **arXiv preprint arXiv:1611.01144**, 2016. Citado na página 61.

JIANG, S.; RIJKE, M. de. Why are sequence-to-sequence models so dull. **EMNLP 2018**, p. 81, 2018. Citado na página 56.

JR, M. O. *et al.* Nurc digital um protocolo para a digitalização, anotação, arquivamento e disseminação do material do projeto da norma urbana linguística culta (nurc). **CHIMERA: Revista de Corpus de Lenguas Romances y Estudios Lingüísticos**, v. 3, n. 2, p. 149–174, 2016. Citado na página 71.

JÚNIOR, A. C.; CASANOVA, E.; SOARES, A. da S.; 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. **CoRR**, abs/2110.15731, 2021. Disponível em: <https://arxiv.org/abs/2110.15731>. Citado nas páginas 71 e 72.

JURAFSKY, D.; MARTIN, J. H. **Speech and language processing**. 2. ed., [pearson international edition]. ed. London [u.a.]: Prentice Hall, Pearson Education International, 2009. 1024 S. p. (Prentice Hall series in artificial intelligence). ISBN 0-13-504196-1, 978-0-13-504196-3. Disponível em: <http://aleph.bib.uni-mannheim.de/F/?func=find-b&request=285413791&find_code=020&adjacent=N&local_base=MAN01PUBLIC&x=0&y=0>. Citado nas páginas 30, 31, 45, 53, 54, 55 e 63.

KARIM, A. Text to speech using mel-spectrogram with deep learning algorithms. **Periodicals of Engineering and Natural Sciences (PEN)**, v. 10, p. 380–386, 07 2022. Citado na página 31.

KHAN, S.; RAHMANI, H.; SHAH, S.; BENNAMOUN, M. A guide to convolutional neural networks for computer vision. **Synthesis Lectures on Computer Vision**, v. 8, p. 1–207, 02 2018. Citado nas páginas 46 e 47.

KHODZHAEV, Z. Spectrogram - practical guide. 02 2022. Citado nas páginas 31 e 32.

KOULIDIS, A. **Modular Testing Facility for Downhole Sensor Evaluation**. Tese (Doutorado), 06 2017. Citado na página 30.

KRIZHEVSKY, A.; HINTON, G. *et al.* Learning multiple layers of features from tiny images. Toronto, ON, Canada, 2009. Citado na página 59.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F.; BURGES, C. J. C.; BOTTOU, L.; WEINBERGER, K. Q. (Ed.). **Advances in Neural Information Processing Systems 25**. Curran Associates, Inc., 2012. p. 1097–1105. Disponível em: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>. Citado na página 42.

KUMAR, V.; NANDI, G. C.; KALA, R. Static hand gesture recognition using stacked denoising sparse autoencoders. In: IEEE. **2014 seventh international conference on contemporary computing (IC3)**. [S.l.], 2014. p. 99–104. Citado na página 49.

LACERDA, T. B.; MIRANDA, P.; CÂMARA, A.; FURTADO, A. P. C. Deep learning and mel-spectrograms for physica violence detection in audio. In: SBC. **Anais do XVIII Encontro Nacional de Inteligência Artificial e Computacional**. [S.l.], 2021. p. 268–279. Citado na página 32.

LECUN, Y.; BENGIO, Y. Convolutional networks for images, speech, and time-series. In: ____. **The handbook of brain theory and neural networks**. [S.l.]: MIT Press, 1995. Citado na página 46.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 521, n. 7553, p. 436–444, maio 2015. ISSN 00280836. Disponível em: <http://dx.doi.org/10.1038/nature14539>. Citado na página 42.

____. Deep learning. **Nature Cell Biology**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, maio 2015. ISSN 1465-7392. Funding Information: Acknowledgements The authors would like to thank the Natural Sciences and Engineering Research Council of Canada, the Canadian Institute For Advanced Research (CIFAR), the National Science Foundation and Office of Naval Research for support. Y.L. and Y.B. are CIFAR fellows. Publisher Copyright: © 2015 Macmillan Publishers Limited. All rights reserved. Citado na página 46.

LI, Z.; ZHAO, M.; HONG, Q.; LI, L.; TANG, Z.; WANG, D.; SONG, L.; YANG, C. Ap20-olr challenge: Three tasks and their baselines. In: IEEE. **2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)**. [S.l.], 2020. p. 550–555. Citado na página 63.

LIKHOMANENKO, T.; XU, Q.; PRATAP, V.; TOMASELLO, P.; KAHN, J.; AVIDOV, G.; COLLOBERT, R.; SYNNAEVE, G. Rethinking evaluation in asr: Are our models robust enough? **arXiv preprint arXiv:2010.11745**, 2020. Citado na página 64.

LIPTON, Z. C.; BERKOWITZ, J.; ELKAN, C. A critical review of recurrent neural networks for sequence learning. **arXiv preprint arXiv:1506.00019**, 2015. Citado na página 52.

LIU, C.; ZHANG, F.; LE, D.; KIM, S.; SARAF, Y.; ZWEIG, G. Improving rnn transducer based asr with auxiliary tasks. In: **2021 IEEE Spoken Language Technology Workshop (SLT)**. [S.l.: s.n.], 2021. p. 172–179. Citado na página 64.

LOPES, T. V. R.; ANDRADE, J. O.; KOMATI, K. S. Comparação de serviços em nuvem para transcrição de fala na língua portuguesa em áudios com sotaques regionais brasileiros. In: SBC. **Anais da IX Escola Regional de Informática de Goiás**. [S.l.], 2021. p. 96–109. Citado nas páginas 24, 25 e 67.

Lopez Pinaya, W. H.; VIEIRA, S.; GARCIA-DIAS, R.; MECHELLI, A. Chapter 11 - autoencoders. In: MECHELLI, A.; VIEIRA, S. (Ed.). **Machine Learning**. Academic Press, 2020. p. 193–208. ISBN 978-0-12-815739-8. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780128157398000110>. Citado nas páginas 48 e 49.

MAAS, A. L. Rectifier nonlinearities improve neural network acoustic models. In: . [S.l.: s.n.], 2013. Citado na página 43.

MADRUGA, M. R. **Fonética e Fonologia da Língua Portuguesa - Processos fonológicos no Português Brasileiro**. [S.l.]: r Editora e Distribuidora Educacional S.A, 2017. Citado na página 33.

MATEUS, M. H. M. Estudando a melodia da fala: traços prosódicos e constituintes prosódicos. **Palavras-Revista da Associação de Professores de Português**, v. 28, p. 79–98, 2004. Citado na página 34.

MATOS, A.; ARAÚJO, G.; JúNIOR, A. C.; PONTI, M. Accent classification is challenging but pre-training helps: a case study with novel Brazilian Portuguese datasets. In: **Proceedings of the 16th International Conference on Computational Processing of Portuguese**. Santiago de Compostela, Galicia/Spain: Association for Computational Lingustics, 2024. p. 364–373. Disponível em: <https://aclanthology.org/2024.propor-1.37>. Citado na página 102.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115–133, 1943. Citado na página 36.

MCLOUGHLIN, I. **Applied Speech and Audio Processing: With Matlab Examples**. [S.l.]: Cambridge University Press, 2009. Citado nas páginas 29 e 31.

MELLO, H.; PETTORINO, M.; RASO, T. **Proceedings of the VIIth GSCP International Conference. Speech and Corpora**. [S.l.]: Firenze University Press, 2012. Citado na página 71.

MELLO, R. F.; PONTI, M. **Machine Learning: A Practical Approach on the Statistical Learning Theory**. Cham: Springer, 2018. Citado na página 38.

MENON, A.; MEHROTRA, K.; MOHAN, C. K.; RANKA, S. Characterization of a class of sigmoid functions with applications to neural networks. **Neural Networks**, v. 9, n. 5, p. 819–835, 1996. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/0893608095001077>. Citado na página 41.

MERGU, R. R.; DIXIT, S. K. A new paradigm for plotting spectrogram. **Journal of Information Systems and Communication**, Bioinfo Publications, v. 3, n. 1, p. 158, 2012. Citado na página 32.

MIKAMI, A. Long short-term memory recurrent neural network architectures for generating music and japanese lyrics. **Comput. Sci. Dep**, p. 1–27, 2016. Citado na página 54.

MIKOLOV, T.; JOULIN, A.; CHOPRA, S.; MATHIEU, M.; RANZATO, M. Learning longer memory in recurrent neural networks. **arXiv preprint arXiv:1412.7753**, 2014. Citado na página 51.

MIKOLOV, T.; KARAFIÁT, M.; BURGET, L.; CERNOCKỲ, J.; KHUDANPUR, S. Recurrent neural network based language model. In: MAKUHARI. **Interspeech**. [S.l.], 2010. v. 2, n. 3, p. 1045–1048. Citado na página 51.

MILAN, G. G. **Identificação de Sotaques Regionais Brasileiros baseada em Inteligência Computacional**. 63 p. — Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2023. Citado nas páginas 67 e 68.

MITCHELL, T. M. **Machine learning**. [S.l.]: McGraw-hill New York, 1997. v. 1. Citado nas páginas 37 e 39.

MOONS, B.; BANKMAN, D.; VERHELST, M. Embedded deep neural networks: Algorithms, architectures and circuits for always-on neural network processing. In: _____. [S.l.: s.n.], 2019. p. 1–31. ISBN 978-3-319-99222-8. Citado na página 41.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th International Conference on International Conference on Machine Learning**. Madison, WI, USA: Omnipress, 2010. (ICML'10), p. 807–814. ISBN 9781605589077. Citado na página 42.

NASCENTES, A. Études dialectologiques du brésil. **ORBIS-Bulletin International de Documentat ion Linguistique, Louvain**, v. 2, n. 2, p. 438–444, 1953. Citado nas páginas 35 e 88.

NEUBIG, G. Neural machine translation and sequence-to-sequence models: A tutorial. **arXiv preprint arXiv:1703.01619**, 2017. Citado na página 56.

NIELSEN, M. A. misc, **Neural Networks and Deep Learning**. Determination Press, 2018. Disponível em: <http://neuralnetworksanddeeplearning.com/>. Citado na página 41.

NWANKPA, C.; IJOMAH, W.; GACHAGAN, A.; MARSHALL, S. Activation functions: Comparison of trends in practice and research for deep learning. **CoRR**, abs/1811.03378, 2018. Disponível em: <http://arxiv.org/abs/1811.03378>. Citado nas páginas 42 e 44.

OLAH, C. **Understanding LSTM networks**. colah's blog, 2015. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>. Citado nas páginas 51 e 55.

OLGAC, A.; KARLIK, B. Performance analysis of various activation functions in generalized mlp architectures of neural networks. **International Journal of Artificial Intelligence And Expert Systems**, v. 1, p. 111–122, 02 2011. Citado na página 42.

OPPENHEIM, A. V. Speech spectrograms using the fast fourier transform. **IEEE Spectrum**, v. 7, n. 8, p. 57–62, 1970. Citado na página 32.

O'SHEA, K.; NASH, R. An introduction to convolutional neural networks. **CoRR**, abs/1511.08458, 2015. Citado na página 47.

PAPASTRATIS, I. Speech recognition: a review of the different deep learning approaches. **https://theaisummer.com/**, 2021. Citado na página 24.

PARK, C.; AHMAD, R.; HAIN, T. Unsupervised data selection for speech recognition with contrastive loss ratios. In: **ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.: s.n.], 2022. p. 8587–8591. Citado na página 61.

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. In: **Interspeech 2019**. ISCA, 2019. (interspeech$_2$019).*Disponvelem* : <>. Citado na página 64.

PARKER, S. O livro do corpo humano. **São Paulo: Ciranda Cultural**, p. 10, 2007. Citado na página 33.

PEDDINTI, V.; POVEY, D.; KHUDANPUR, S. A time delay neural network architecture for efficient modeling of long temporal contexts. In: . [S.l.: s.n.], 2015. p. 3214–3218. Citado na página 63.

PONTI, M. A.; RIBEIRO, L. S. F.; NAZARE, T. S.; BUI, T.; COLLOMOSSE, J. Everything you wanted to know about deep learning for computer vision but were afraid to ask. In: IEEE. **2017 30th SIBGRAPI conference on graphics, patterns and images tutorials (SIBGRAPI-T)**. [S.l.], 2017. p. 17–41. Citado na página 46.

PONTI, M. A.; SANTOS, F. P. dos; RIBEIRO, L. S.; CAVALLARI, G. B. Training deep networks from zero to hero: avoiding pitfalls and going beyond. In: IEEE. **2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)**. [S.l.], 2021. p. 9–16. Citado na página 55.

QUINTANILHA, I. M.; NETTO, S. L.; BISCAINHO, L. 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, Sep. 2020. Disponível em: <https://jcis.sbrt.org.br/jcis/article/view/721>. Citado na página 24.

RADFORD, A.; KIM, J. W.; XU, T.; BROCKMAN, G.; MCLEAVEY, C.; SUTSKEVER, I. **Robust Speech Recognition via Large-Scale Weak Supervision**. 2022. Citado na página 75.

RAMCHOUN, H.; GHANOU, Y.; ETTAOUIL, M.; IDRISSI, M. A. J. Multilayer perceptron: Architecture optimization and training. International Journal of Interactive Multimedia and Artificial Intelligence . . . , 2016. Citado na página 40.

RASO, T.; MELLO, H. **C-ORAL-BRASIL: corpus de referência do português brasileiro falado informal. I**. [S.l.]: Editora UFMG, 2012. Citado na página 71.

RESENDE, D. C. O. d.; PONTI, M. A. Robust image features for classification and zero-shot tasks by merging visual and semantic attributes. **Neural Computing and Applications**, v. 34, n. 6, p. 4459–4471, 2022. Citado na página 48.

REYNOLDS, D. A. *et al.* Gaussian mixture models. **Encyclopedia of biometrics**, Berlin, Springer, v. 741, n. 659-663, 2009. Citado na página 65.

RISTA, A.; KADRIU, A. Automatic speech recognition: A comprehensive survey. **SEEU Review**, v. 15, p. 86–112, 12 2020. Citado na página 23.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, 1958. ISSN 0033-295X. Disponível em: <http://dx.doi.org/10.1037/h0042519>. Citado nas páginas 36 e 38.

RUDER, S. An overview of gradient descent optimization algorithms. **CoRR**, abs/1609.04747, 2016. Disponível em: <http://arxiv.org/abs/1609.04747>. Citado na página 40.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. In: RUMELHART, D. E.; MCCLELLAND, J. L. (Ed.). **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations**. Cambridge, MA: MIT Press, 1986. p. 318–362. Citado na página 41.

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M. *et al.* Imagenet large scale visual recognition challenge. **International journal of computer vision**, Springer, v. 115, p. 211–252, 2015. Citado na página 59.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3. ed. [S.l.]: Prentice Hall, 2010. Citado nas páginas 29 e 36.

SALEHINEJAD, H.; SANKAR, S.; BARFETT, J.; COLAK, E.; VALAEE, S. Recent advances in recurrent neural networks. **arXiv preprint arXiv:1801.01078**, 2017. Citado na página 51.

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**. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2102.01757>. Citado na página 71.

SANTRA, S.; HSIEH, J.-W.; LIN, C.-F. Gradient descent effects on differential neural architecture search: A survey. **IEEE Access**, v. 9, p. 89602–89618, 2021. Citado na página 40.

SCHMIDHUBER, J.; HOCHREITER, S. *et al.* Long short-term memory. **Neural Comput**, v. 9, n. 8, p. 1735–1780, 1997. Citado na página 52.

SEARA, I. C.; NUNES, V. G.; LAZZAROTTO-VOLCÃO, C. Para conhecer fonética e fonologia do português brasileiro. **(No Title)**, 2017. Citado nas páginas 33 e 34.

SELLAT, Q.; BISOY, S. K.; PRIYADARSHINI, R. Chapter 10 - semantic segmentation for self-driving cars using deep learning: a survey. In: MISHRA, S.; TRIPATHY, H. K.; MALLICK, P. K.;

SANGAIAH, A. K.; CHAE, G.-S. (Ed.). **Cognitive Big Data Intelligence with a Metaheuristic Approach**. Academic Press, 2022, (Cognitive Data Science in Sustainable Computing). p. 211–238. ISBN 978-0-323-85117-6. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780323851176000029>. Citado na página 41.

SELVARAJU, R. R.; COGSWELL, M.; DAS, A.; VEDANTAM, R.; PARIKH, D.; BATRA, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In: **Proceedings of the IEEE international conference on computer vision**. [S.l.: s.n.], 2017. p. 618–626. Citado nas páginas 83 e 84.

SHARMA, S.; SHARMA, S.; ATHAIYA, A. Activation functions in neural networks. **towards data science**, v. 6, n. 12, p. 310–316, 2017. Citado na página 44.

SHI, X.; YU, F.; LU, Y.; LIANG, Y.; FENG, Q.; WANG, D.; QIAN, Y.; XIE, L. The accented english speech recognition challenge 2020: open datasets, tracks, baselines, results and methods. In: IEEE. **ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2021. p. 6918–6922. Citado na página 24.

SHI, Y.; WANG, Y.; WU, C.; YEH, C.-F.; CHAN, J.; ZHANG, F.; LE, D.; SELTZER, M. Emformer: Efficient memory transformer based acoustic model for low latency streaming speech recognition. In: IEEE. **ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2021. p. 6783–6787. Citado na página 64.

SILVA, T. C. Fonética e fonologia do português: roteiro de estudos e guia de exercícios. Contexto, 2003. Citado nas páginas 33 e 34.

SILVA, V. M. L. *et al.* Conversor a/d com amostragem não-uniforme e passo de quantização adaptativo. Universidade Federal da Paraíba, 2014. Citado na página 31.

SKANSI, S. **Introduction to Deep Learning: from logical calculus to artificial intelligence**. [S.l.]: Springer, 2018. Citado nas páginas 39 e 52.

SOLOVYEV, R. A.; VAKHRUSHEV, M.; RADIONOV, A.; ROMANOVA, I. I.; AMERIKANOV, A. A.; ALIEV, V.; SHVETS, A. A. Deep learning approaches for understanding simple speech commands. In: **2020 IEEE 40th International Conference on Electronics and Nanotechnology (ELNANO)**. [S.l.: s.n.], 2020. p. 688–693. Citado na página 32.

SUTSKEVER, I.; MARTENS, J.; HINTON, G. E. Generating text with recurrent neural networks. In: **ICML**. [S.l.: s.n.], 2011. Citado nas páginas 51 e 52.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. **Advances in neural information processing systems**, v. 27, 2014. Citado na página 56.

TANAKA, E.; CLIFTON, A.; CORREIA, J.; JAT, S.; JONES, R.; KARLGREN, J.; ZHU, W. **Cem Mil Podcasts: A Spoken Portuguese Document Corpus**. arXiv, 2022. Disponível em: <https://arxiv.org/abs/2209.11871>. Citado na página 70.

TEIXEIRA, C.; TRANCOSO, I.; SERRALHEIRO, A. Accent identification. In: IEEE. **Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP'96**. [S.l.], 1996. v. 3, p. 1784–1787. Citado na página 24.

TOSTES, W. A. Arquiteturas de redes neurais profundas para classificação de dialetos e sotaques. Serra, 2022. Citado nas páginas 67, 76 e 92.

TOSTES, W. A.; BOLDT, F. A.; KOMATI, K. S.; MUTZ, F. Classificaçao de sotaques brasileiros usando redes neurais profundas. In: **Simpósio Brasileiro de Automação Inteligente-SBAI**. [S.l.: s.n.], 2021. v. 1, n. 1. Citado nas páginas 24, 25, 66, 68, 76 e 77.

URBAN, S.; BASALLA, M.; SMAGT, P. van der. **Gaussian Process Neurons Learn Stochastic Activation Functions**. arXiv, 2017. Disponível em: <https://arxiv.org/abs/1711.11059>. Citado na página 42.

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. Citado nas páginas 24, 56, 57 e 58.

VENKATESAN, R.; LI, B. **Convolutional neural networks in visual computing: a concise guide**. [S.l.]: CRC Press, 2017. Citado na página 47.

VINCENT, P.; LAROCHELLE, H.; LAJOIE, I.; BENGIO, Y.; MANZAGOL, P.-A.; BOTTOU, L. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. **Journal of machine learning research**, v. 11, n. 12, 2010. Citado na página 50.

WANG, D.; YE, S.; HU, X.; LI, S.; XU, X. An end-to-end dialect identification system with transfer learning from a multilingual automatic speech recognition model. In: **Interspeech**. [S.l.: s.n.], 2021. p. 3266–3270. Citado nas páginas 63, 64 e 65.

WANG, M.; LU, S.; ZHU, D.; LIN, J.; WANG, Z. A high-speed and low-complexity architecture for softmax function in deep learning. In: **2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)**. [S.l.: s.n.], 2018. p. 223–226. Citado nas páginas 24 e 45.

WANG, X.; XU, J.; SHI, W.; LIU, J. Ogru: An optimized gated recurrent unit neural network. In: IOP PUBLISHING. **Journal of Physics: Conference Series**. [S.l.], 2019. v. 1325, n. 1, p. 012089. Citado nas páginas 54 e 55.

WEENINK, D. **Speech Signal Processing with Praat**. [s.n.], 2022. Disponível em: <http://www.fon.hum.uva.nl/david/sspbook/sspbook.pdf>. Citado na página 30.

WENG, L. Attention? attention! **lilianweng.github.io**, 2018. Disponível em: <https://lilianweng.github.io/posts/2018-06-24-attention/>. Citado na página 56.

WENINGER, F.; SUN, Y.; PARK, J.; WILLETT, D.; ZHAN, P. Deep learning based mandarin accent identification for accent robust asr. In: . [S.l.: s.n.], 2019. p. 510–514. Citado na página 24.

XU, J.; LI, Z.; DU, B.; ZHANG, M.; LIU, J. Reluplex made more practical: Leaky relu. In: **2020 IEEE Symposium on Computers and Communications (ISCC)**. [S.l.: s.n.], 2020. p. 1–7. Citado na página 44.

XU, X.; KANG, Y.; CAO, S.; LIN, B.; MA, L. Explore wav2vec 2.0 for mispronunciation detection. In: **Interspeech**. [S.l.: s.n.], 2021. p. 4428–4432. Citado na página 60.

YAN, S. **Understanding LSTM and Its Diagrams**. 2016. Disponível em: <https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714>. Citado na página 53.

YNOGUTI, C. Reconhecimento de fala contínua utilizando modelos ocultos de markov. **Faculdade de Engenharia Elétrica–UNICAMP**, 1999. Citado na página 65.

YU, D.; DENG, L. **Automatic Speech Recognition: A Deep Learning Approach**. London: Springer, 2015. (Signals and Communication Technology). ISSN 1860-4862. ISBN 978-1-4471-5778-6. Citado na página 23.

YU, Y.; SI, X.; HU, C.; ZHANG, J. A review of recurrent neural networks: LSTM cells and network architectures. **Neural Computation**, MIT Press - Journals, v. 31, n. 7, p. 1235–1270, jul 2019. Disponível em: <https://doi.org/10.1162%2Fneco_a_01199>. Citado na página 53.

ZHAI, X.; PUIGCERVER, J.; KOLESNIKOV, A.; RUYSSEN, P.; RIQUELME, C.; LUCIC, M.; DJOLONGA, J.; PINTO, A. S.; NEUMANN, M.; DOSOVITSKIY, A. *et al.* A large-scale study of representation learning with the visual task adaptation benchmark. arxiv 2019. **arXiv preprint arXiv:1910.04867**, 1910. Citado na página 59.

ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. Dive into deep learning. **arXiv preprint arXiv:2106.11342**, 2021. Citado nas páginas 42, 43, 45, 46, 47, 48 e 56.

ZHANG, J. Gradient descent based optimization algorithms for deep learning models training. **arXiv preprint arXiv:1903.03614**, 2019. Citado na página 40.

ZULUAGA, J. P.; AHMED, S.; VISOCKAS, D.; SUBAKAN, C. Commonaccent: Exploring large acoustic pretrained models for accent classification based on common voice. In: . [S.l.: s.n.], 2023. p. 5291–5295. Citado nas páginas 64, 65 e 95.

APÊNDICE

A

# ARTIGO

# A.1 Classificação de Sotaques para o Português do Brasil

## Accent Classification is Challenging but Pre-training Helps: a case study with novel Brazilian Portuguese datasets

**Ariadne Nascimento Matos**
ICMC – Universidade de São Paulo, Brazil
ariadnenmtos@usp.br

**Gustavo Evangelista Araújo**
ICMC – Universidade de São Paulo, Brazil

**Arnaldo Candido Junior**
IBILCE - Universidade Estadual Paulista

**Moacir Antonelli Ponti**
ICMC – Universidade de São Paulo, Brazil

### Abstract

Accents arise due to variations in pronunciation, intonation, and other speech characteristics caused by geographical, cultural, or linguistic differences. Investigating accent classification methods is a way towards accent-aware speech-processing. This paper evaluates accent classification for spontaneous speech using CNN-LSTM networks and the Wav2vec2 model. We study the importance of dataset size, pre-trained models, and external validation. For that we used 90 hours of data, encompassing 9 accents and involving 204 speakers of Brazilian Portuguese, obtained from manually annotated subsets from Spotify Podcasts [1] and CORAA ASR. Our best results range from 82% (closed-dataset) to 75% (cross-dataset) f1-scores for binary classification. Unless there is speaker leakage from training to testing, accent classification models trained from scratch fail for spontaneous speech data. Therefore, methods should be evaluated using both out-of-speaker and cross-dataset scenarios. We contributed with an experimental protocol for this task with a novel dataset. Finally, our results highlight the value of larger accent-annotated datasets, and the use of larger pretrained-models.

### 1 Introduction

Speech is a fundamental form of human communication, allowing expressing ideas and information. Automatic methods for processing and understanding speech are a relevant subject of study. Machine learning techniques are shown to be particularly useful in this scenario, becoming the state of the art in many speech processing tasks (Casanova et al., 2023, 2022). The two most remarkable tasks in this context are Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) systems. One of the challenges in ASR and TTS is how to deal with different accents of a given language, which can significantly impact the system's performance.

[1] https://github.com/aryamtos/spotify-subset

Accents arise due to variations in pronunciation, intonation, and other speech characteristics caused by geographical, cultural, or linguistic differences (Lippi-Green, 2012). There are two different types of accents: the first refers to foreignness, which occurs when a person speaks a language using rules and sounds of another language, and the second occurs within the native language itself (Teixeira et al., 1996). This paper aims at the automatic classification of the second type of accent.

Based on the dialectical division proposed by Nascentes (1953), Brazil is divided into two linguistic groups, related to Northern and Southern regions. The Northern region has specific phonological and morphological features, such as pretonic vowels, with a greater oral aperture facilitating airflow, in contrast to the closed vowel pronunciation typical of the Southern and Southeastern regions.

According to Ilari and Basso (2009), the regional characteristics of Brazilian Portuguese are distinguished by various pronunciation features. One notable feature is the absence of palatalization in the pronunciation of /t/ and /d/, a phenomenon widespread throughout Brazil except in São Paulo and the southern region. Additionally, the retroflex pronunciation of /r/ is a distinctive trait observed in the "caipira dialect" (Ilari and Basso, 2009). Previous accent classification methods also follow this definition (Batista et al., 2018; Batista, 2019). We focus on matching the accents within different Brazilian states, prioritizing the ones with the most data available. By that, we expect to offer a model that could fit in different dialectical divisions. When considering states within the North and South, we are offering a more fine-grained classification of Brazilian Portuguese accents.

The variations caused by accents can result in differences in acoustic features, such as the spectral content and timing of speech signals (Hansen et al., 2020). Amplitude modulations of the envelope with different timescales are also associated with

APÊNDICE

# B

# RESULTADOS CLASSIFICAÇÃO DE VARIAÇÕES LINGUÍSTICAS

## B.1 Classificação Multiclasse - Closed-set

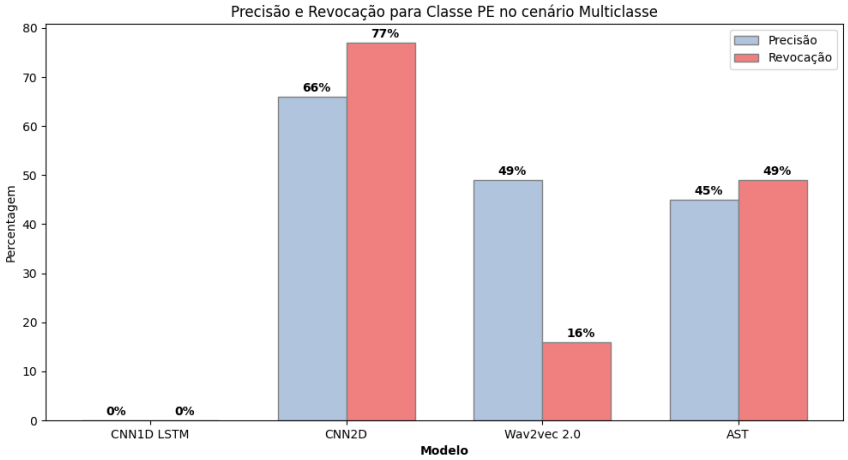



Figura 40 – Cenário Closed-set utilizando o Spotify-B para classificação multiclasse - Classe PE

## B.2 Classificação Multiclasse - Cross-dataset

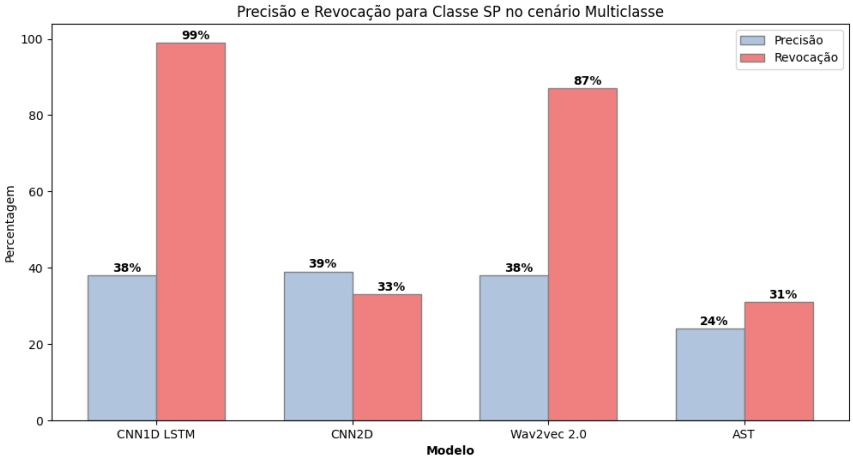



Figura 41 – Cenário Closed-set utilizando o Spotify-B para classificação multiclasse - Classe SP

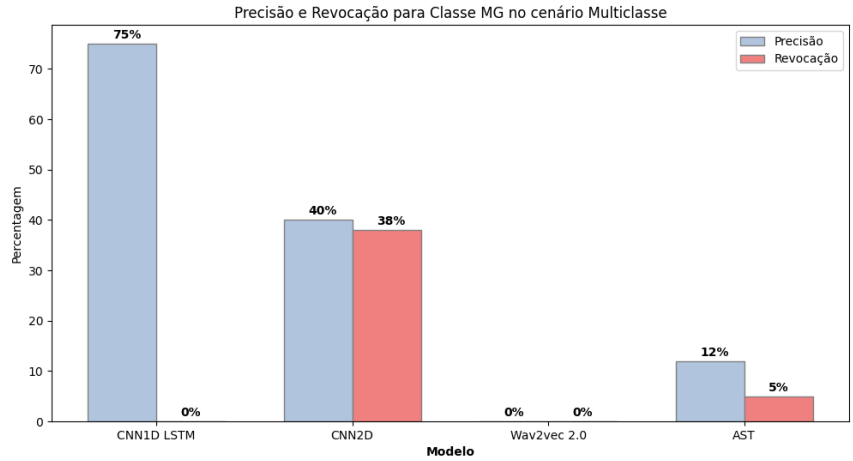



Figura 42 – Cenário Closed-set utilizando o Spotify-B para classificação multiclasse - Classe MG

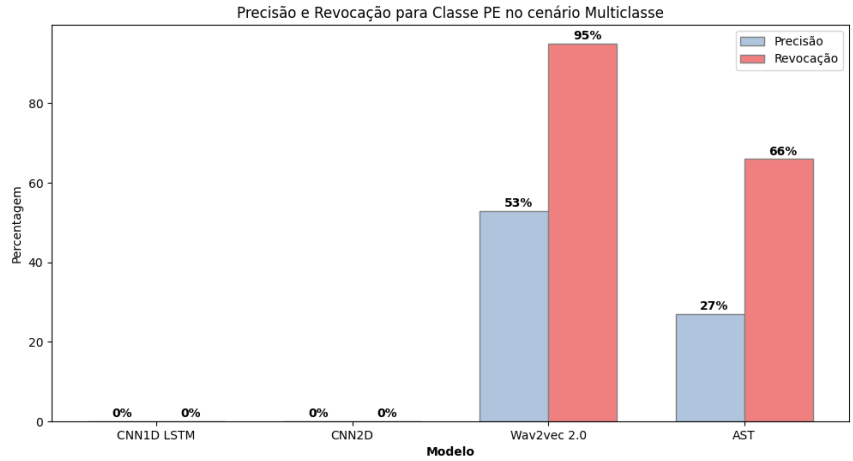



Figura 43 – Cenário cross-dataset testando com o CORAA-ASR - Classe PE

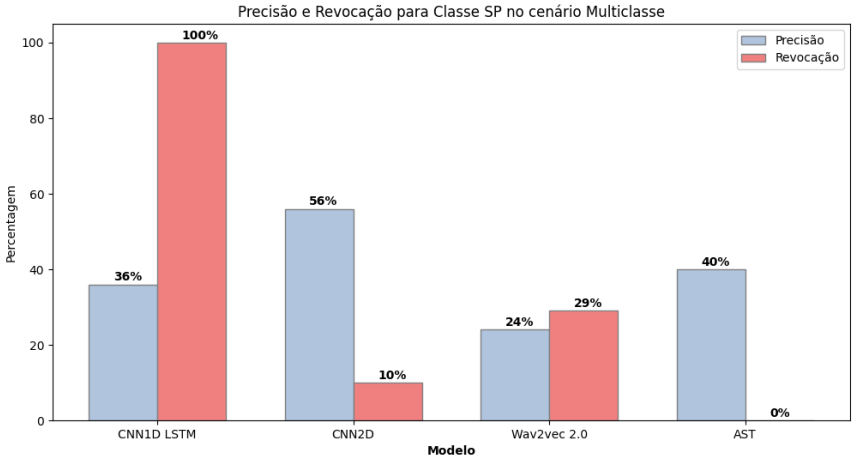



Figura 44 – Cenário cross-dataset testando com o CORAA-ASR - Classe SP

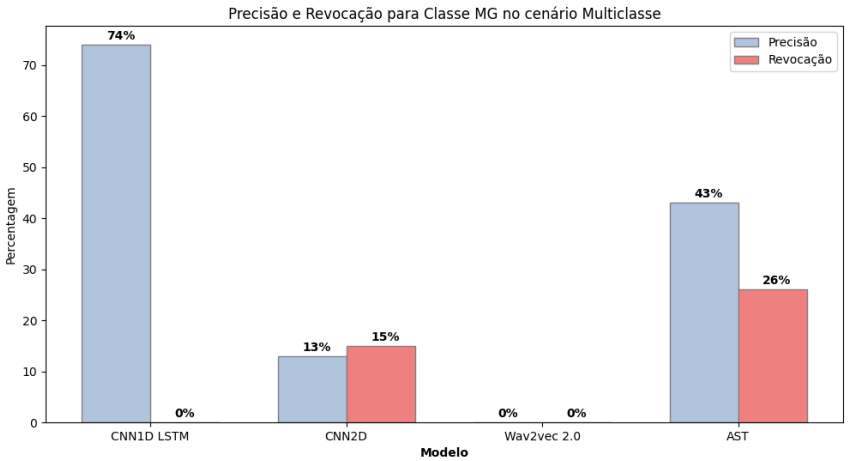



Figura 45 – Cenário cross-dataset testando com o CORAA-ASR - Classe MG


ICMC
USP
SÃO CARLOS