

**UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO  
CENTRO TECNOLÓGICO  
DEPARTAMENTO DE ENGENHARIA ELÉTRICA  
PROJETO DE GRADUAÇÃO**

**LUCAS GRIGOLETO SCART**

**RECONHECIMENTO AUTOMÁTICO DE FALA EM  
PORTUGUÊS UTILIZANDO ARQUITETURAS DE REDES  
NEURAIS PROFUNDAS**

VITÓRIA  
2019

LUCAS GRIGOLETO SCART

**RECONHECIMENTO AUTOMÁTICO DE FALA EM PORTUGUÊS  
UTILIZANDO ARQUITETURAS DE REDES NEURAS PROFUNDAS**

Parte manuscrita do Projeto de Graduação do aluno Lucas Grigoletto Scart, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Orientador: Prof. Dr. Jorge Leonid Aching Samatelo

VITÓRIA  
2019

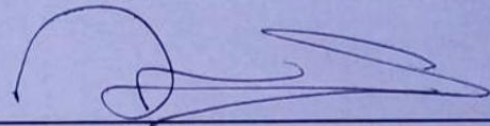
Lucas Grigoletto Scart

# RECONHECIMENTO AUTOMÁTICO DE FALA EM PORTUGUÊS UTILIZANDO ARQUITETURAS DE REDES NEURAS PROFUNDAS

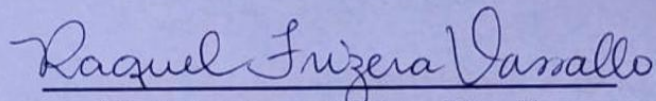
Parte manuscrita do Projeto de Graduação do aluno Lucas Grigoletto Scart, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Aprovado em 16 de dezembro de 2019.

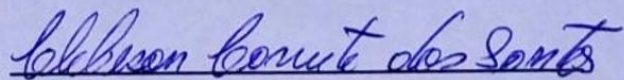
## COMISSÃO EXAMINADORA:



**Prof. Dr. Jorge Leonid Aching Samatelo**  
Universidade Federal do Espírito Santo  
Orientador



**Prof. Dra. Raquel Frizera Vassallo**  
Universidade Federal do Espírito Santo  
Examinadora



**Msc. Clebeson Canuto dos Santos**  
Universidade Federal do Espírito Santo  
Examinador

Vitória-ES

Dezembro/2019

Gostaria de agradecer a todas essas pessoas especiais que apareceram (e estão) na minha vida e que permitem que eu continue com os meus estudos. A todos vocês, minha eterna gratidão.

Aos meus pais, Agenor e Maluce, pelo apoio e dedicação durante toda a graduação.

A meu orientador Jorge por despertar meu interesse por esse tema fascinante e por toda ajuda, orientação e dedicação durante o desenvolvimento deste trabalho.

À banca examinadora pela aceitação do convite e pelo tempo investido para leitura e avaliação desse trabalho.

Agradeço à Universidade Federal do Espírito Santo pela minha formação.

## RESUMO

Dentre as formas de comunicação utilizadas entre seres humanos, a principal delas é a fala. Em busca de métodos que possibilitem a interação entre homens e máquinas de forma natural, modelos de aprendizado de máquina podem ser utilizados para transformar o áudio capturado da voz de uma pessoa em uma representação que os computadores consigam compreender. Apesar do grande avanço obtido com a utilização de modelos conhecidos como redes neurais profundas, a maior parte da pesquisa se concentra no estudo da língua inglesa, de modo que outros idiomas se encontram defasados. Com base nessa situação, o presente estudo propõe um sistema de reconhecimento automático de fala utilizando redes neurais profundas para o idioma Português Brasileiro. O modelo neural foi treinado de ponta a ponta para transformar a fala de entrada em uma sequência de caracteres que depois é decodificada em palavras. Utilizando apenas bases de dados disponíveis abertamente, foi obtido resultado próximo ao estado da arte.

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

## LISTA DE FIGURAS

Figura 1 – Diagrama de blocos de um sistema de reconhecimento de fala.....	18
Figura 2 – Extração de características a partir de sinal de áudio (a) Sinal original; (b) Espectrograma em escala Mel; (c) Coeficientes Mel-Cepstrais .....	19
Figura 3 – Célula básica que compõe rede GRU.....	22
Figura 4 – Fluxo de informação em uma célula GRU referente ao: (a) <i>Reset Gate</i> ; (b) <i>Update Gate</i> ; (c) atualização do estado da célula e (d) <i>Output Gate</i> .....	23
Figura 5 – Exemplo de decodificação utilizando o algoritmo CTC .....	25
Figura 6 – Cálculo da probabilidade de sequências específicas a partir de probabilidades distribuídas no tempo .....	26
Figura 7 – Arquitetura do modelo utilizado .....	31

## LISTA DE GRÁFICOS

Gráfico 1 – Variação dos hiper-parâmetros (a) taxa de aprendizado e (b) momento ao longo do treino.....	35
---	----

## **LISTA DE QUADROS**

Quadro 1 – Camadas presentes no modelo proposto .....	29
Quadro 2 – Exemplo de operações consideradas no cálculo da métrica.....	34
Quadro 3 – Exemplos de resultados obtidos .....	36



## **LISTA DE TABELAS**

Tabela 1 – Resultados para a abordagem proposta, na transcrição de áudios presentes na base de testes .....	36
Tabela 2 – Comparação com resultados obtidos por outros trabalhos.....	37

## LISTA DE ABREVIATURAS E SIGLAS

BRSD	<i>Brazilian Portuguese Speech Dataset</i>
CTC	<i>Connectionist Temporal Classification</i>
FFT	<i>Fast Fourier Transform</i>
GRU	<i>Gated Recurrent Unit</i>
MFCC	<i>Mel-Frequency Cepstral Coefficients</i>
UFES	Universidade Federal do Espírito Santo

## LISTA DE SÍMBOLOS

$\mathbf{X}$	Vetor acústico
$\mathbf{Y}$	Transcrição
$\mathbf{W}$	Sequência de palavras
$\mathbf{F}$	Sequência de fonemas
$P(\mathbf{X})$	Observação
$P(\mathbf{W})$	Modelo de linguagem
$P(\mathbf{X} \mathbf{W})$	Modelo acústico
$P(\mathbf{F} \mathbf{W})$	Modelo de pronúncia
$A_{\mathbf{X},\mathbf{Y}}$	Conjunto de alinhamentos entre $\mathbf{X}$ e $\mathbf{Y}$
$\alpha$	Caractere do vocabulário

# SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO.....</b>	<b>13</b>
1.1	<b>Apresentação e Objeto de Pesquisa .....</b>	<b>13</b>
1.2	<b>Justificativa.....</b>	<b>14</b>
1.3	<b>Objetivos.....</b>	<b>14</b>
1.3.1	Objetivo Geral .....	14
1.3.2	Objetivos Específicos .....	14
<b>2</b>	<b>EMBASAMENTO TEÓRICO .....</b>	<b>16</b>
2.1	<b>Introdução .....</b>	<b>16</b>
2.2	<b>Arquitetura de um sistema de reconhecimento de fala tradicional.....</b>	<b>16</b>
2.3	<b>Extração de características a partir de sinais de áudio.....</b>	<b>17</b>
2.3.1	Coeficientes Mel-Cepstrais .....	18
2.4	<b>Redes Neurais Profundas .....</b>	<b>20</b>
2.4.1	Redes Neurais Convolucionais.....	20
2.4.2	Redes Neurais Recorrentes .....	21
2.4.3	Gated Recurrent Unit.....	21
2.4.4	Processamento na célula GRU .....	22
2.5	<i>Connectionist Temporal Classification</i> .....	24
2.6	<b>Decodificação.....</b>	<b>26</b>
2.7	<b>Resumo .....</b>	<b>27</b>
<b>3</b>	<b>SOLUÇÃO PROPOSTA.....</b>	<b>28</b>
3.1	<b>Introdução .....</b>	<b>28</b>
3.2	<b>Pré-processamento dos dados .....</b>	<b>28</b>
3.3	<b>Modelo utilizado para o reconhecimento de fala .....</b>	<b>29</b>
3.4	<b>Resumo .....</b>	<b>30</b>
<b>4</b>	<b>EXPERIMENTOS E RESULTADOS.....</b>	<b>32</b>
4.1	<b>Introdução .....</b>	<b>32</b>
4.2	<b>Recursos Computacionais .....</b>	<b>32</b>
4.3	<b>Métricas .....</b>	<b>34</b>
4.4	<b>Experimento .....</b>	<b>35</b>
4.5	<b>Comparação de resultados.....</b>	<b>36</b>
4.6	<b>Resumo .....</b>	<b>37</b>
<b>5</b>	<b>CONCLUSÕES E PROJETOS FUTUROS.....</b>	<b>38</b>

<b>5.1 Conclusões .....</b>	<b>38</b>
<b>5.2 Temas a serem pesquisados.....</b>	<b>38</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS .....</b>	<b>39</b>

# 1 INTRODUÇÃO

## 1.1 Apresentação e Objeto de Pesquisa

Com o crescimento do ramo da inteligência artificial, mostra-se cada vez mais necessário o desenvolvimento de métodos que possibilitem a interação entre homens e máquinas de forma natural. Dentre as formas de comunicação utilizadas entre seres humanos, a principal delas é a fala, atraindo a atenção de pesquisadores que tentam transformar o áudio capturado da voz de uma pessoa em uma representação que os computadores consigam compreender.

Sinais obtidos a partir de fala podem fornecer diferentes tipos de informação, de acordo com a aplicação desejada. Segundo Nassif e outros (2019), a partir da fala pode-se extrair informações que permitem a identificação do locutor, da língua que está sendo falada, do sotaque de diferentes locutores dentro de uma mesma linguagem, o estado emocional e de saúde do locutor, além de sua idade e gênero.

Além das informações citadas anteriormente, pode-se ainda realizar o reconhecimento automático de fala, permitindo a uma máquina reconhecer o conteúdo das palavras e frases em uma linguagem pronunciada e transformá-las em um formato compreensível.

Nas últimas décadas, métodos baseados em aprendizado de máquina têm se mostrado eficazes em aplicações para processamento de voz, especialmente para a tarefa de reconhecimento de fala. Porém, pesquisas recentes têm como foco utilizar um subconjunto de modelos de aprendizado de máquina, conhecidos como redes neurais profundas, apresentando resultados superiores aos obtidos com a utilização de outros modelos clássicos de aprendizado.

Propôs-se o estudo e implementação de arquiteturas de redes neurais que, treinadas com bases de dados disponíveis abertamente, servem como linha de base para pesquisas futuras em linguagens defasadas, em especial o português.

## 1.2 Justificativa

Apesar do grande avanço obtido na área de processamento de voz utilizando redes neurais profundas, conforme mostrado por Nassif e outros (2019), ao serem analisados 174 artigos publicados entre 2006 e 2018 nas principais conferências da área, foi constatado que 85% dos artigos utilizavam apenas bases de dados referentes à língua inglesa, de modo que outros idiomas encontram-se defasados.

Existe também a necessidade do estudo de métodos mais eficientes para o treinamento de modelos em linguagens com poucos dados disponíveis. Em aplicações de visão computacional, é realizado o *fine tuning* ao se aproveitar partes de modelos de redes neurais convolucionais já treinados para tarefas associadas (MAHAJAN *et al.*, 2018). Já na área de processamento de linguagem natural, avanços recentes possibilitaram o treino de modelos de linguagem em grandes bancos de dados para a posterior utilização nas mais diversas tarefas (HOWARD; RUDER, 2018; DEVLIN *et al.*, 2018). *Wav2vec* (SCHNEIDER *et al.*, 2019) surge como uma alternativa para se tentar realizar o treinamento prévio de modelos para a utilização em aplicações de processamento de voz. Porém, novamente, existe o problema de que a principal linguagem utilizada foi o inglês.

## 1.3 Objetivos

### 1.3.1 Objetivo Geral

Este trabalho teve como objetivo desenvolver um sistema de reconhecimento automático de fala utilizando redes neurais profundas para o idioma Português Brasileiro (PT-BR) utilizando para isso somente bases de dados disponíveis abertamente. O modelo neural foi treinado para transformar a fala de entrada em uma sequência de caracteres que depois será decodificada em palavras.

### 1.3.2 Objetivos Específicos

Para se alcançar o objetivo geral, foi necessário atender a alguns objetivos específicos. Dentre eles:

- obter conjuntos de dados de domínio público que contenham áudio em português e a sua correta transcrição;
- estudar as principais transformações aplicadas ao áudio utilizadas em sistemas de reconhecimento de fala;
- estudar as arquiteturas de redes neurais profundas utilizadas para realizar o reconhecimento;
- treinar um modelo usando os conjuntos de dados selecionados;
- realizar a avaliação dos resultados obtidos sistematicamente, comparando com outros presentes na literatura.



## 2 EMBASAMENTO TEÓRICO

### 2.1 Introdução

Neste capítulo, será discutido e analisado todo o conhecimento teórico necessário para realizar o trabalho proposto. Será discutida, inicialmente, a arquitetura básica de um sistema de reconhecimento de fala tradicional. A seguir, será explicado o pré-processamento realizado para extrair os vetores acústicos a partir de um sinal de áudio. Em seguida, serão explicadas as arquiteturas de redes neurais profundas comumente utilizadas para a tarefa proposta. Por fim, serão explicados os métodos de decodificação das predições.

### 2.2 Arquitetura de um sistema de reconhecimento de fala tradicional

O objetivo de um sistema de reconhecimento de fala é transcrever uma fala em uma sequência de palavras  $\mathbf{W}^*$ . Para isso, um sinal de áudio captado a partir de um microfone é convertido em um vetor de características acústicas  $\mathbf{X}$  por meio de um processo de extração de características. Então, um decoder tenta encontrar a sequência de palavras  $\mathbf{W}$  que possui a maior probabilidade de ter gerado  $\mathbf{X}$ . Isso pode ser feito maximizando a probabilidade de cada palavra dado os vetores acústicos  $P(\mathbf{W}|\mathbf{X})$ . Porém, este é um problema de otimização que possui grande custo computacional. Com o intuito de obter uma solução realizável computacionalmente, pode-se utilizar a regra de *Bayes*, para descrever a probabilidade de um evento  $P(\mathbf{W}|\mathbf{X})$  baseado em um conhecimento à priori  $P(\mathbf{X}|\mathbf{W})$ , que pode estar relacionado ao evento, reescrevendo a probabilidade de acordo com a equação (2.1).

$$\begin{aligned}\mathbf{W}^* &= \operatorname{argmax}_w P(\mathbf{W}|\mathbf{X}) \\ &= \operatorname{argmax}_w \frac{P(\mathbf{X}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{X})}\end{aligned}\tag{2.1}$$

Onde:  $P(\mathbf{X}|\mathbf{W})$  é chamado de modelo acústico, representado como a probabilidade do vetor acústico dada a sequência de palavras;  $P(\mathbf{W})$  é o modelo de linguagem, representado como a probabilidade à priori da sequência de palavras;  $P(\mathbf{X})$  é a observação, representado como a probabilidade à priori do vetor acústico.

Modelar diretamente os vetores acústicos de milhares a milhões de palavras em sistemas de reconhecimento de fala com um amplo vocabulário não é uma opção realística devido ao problema de escassez de dados. Em vez disso, os modelos acústicos são treinados para reconhecer sequências dos fonemas ( $\mathbf{F}$ ), que são mapeados na sequência de palavras  $\mathbf{W}$ . Os fonemas são definidos como a menor unidade linguística perceptível da fala. Assim, todo o processo de reconhecimento de fala pode ser descrito como a busca da sequência ideal de palavras, conforme a equação (2.2).

$$\begin{aligned}
 \mathbf{W}^* &= \operatorname{argmax}_W P(\mathbf{W}|\mathbf{X}) \\
 &= \operatorname{argmax}_W \sum_{\mathbf{F}} P(\mathbf{W}, \mathbf{F}|\mathbf{X}) \\
 &= \operatorname{argmax}_W \sum_{\mathbf{F}} \frac{P(\mathbf{W}, \mathbf{F}, \mathbf{X})}{P(\mathbf{X})} \\
 &= \operatorname{argmax}_W \sum_{\mathbf{F}} \frac{P(\mathbf{X}|\mathbf{F})P(\mathbf{F}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{X})}
 \end{aligned} \tag{2.2}$$

Onde:  $\mathbf{F}$  é a sequência de fonemas;  $P(\mathbf{F}|\mathbf{W})$  é o modelo de pronúncia, representado como a probabilidade da sequência de fonemas dada a sequência de palavras.  $P(\mathbf{X}|\mathbf{F})$  é o modelo acústico, representado como a probabilidade do vetor acústico dada a sequência de fonemas.

A equação (2.2) geralmente é simplificada ao escolher a sequência de fonemas mais provável para permitir uma pesquisa eficiente, obtendo-se a equação (2.3).

$$\mathbf{W}^* = \operatorname{argmax}_W \frac{P(\mathbf{X}|\mathbf{F})P(\mathbf{F}|\mathbf{W})P(\mathbf{W})}{P(\mathbf{X})}. \tag{2.3}$$

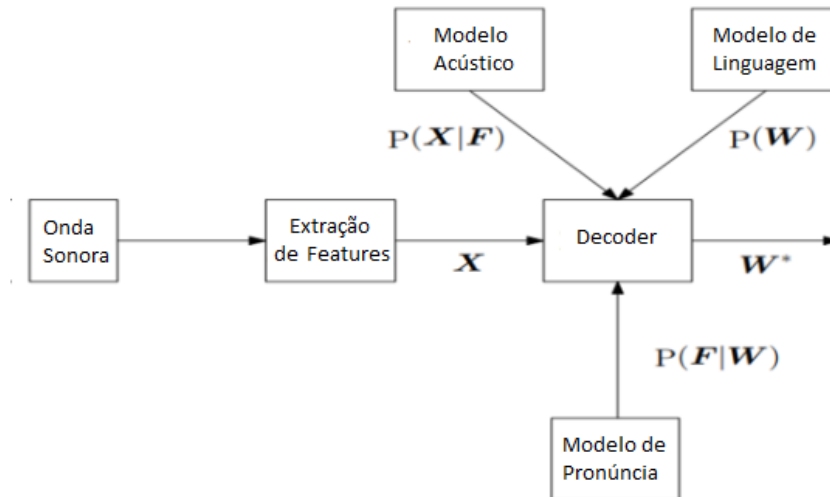
Um resumo visual de um sistema de reconhecimento de fala pode ser encontrado na Figura 1.

### 2.3 Extração de características a partir de sinais de áudio

Apesar de redes neurais demandarem menos esforço na extração de características quando comparadas com outros métodos de aprendizado de máquina, alimentar o modelo diretamente com os sinais de áudios puros não funciona bem. Deve-se transformar o sinal de áudio para

uma representação que evidencie as componentes do áudio que são adequadas para o objetivo, enquanto descartem as informações que possam prejudicar o modelo, como ruídos presentes na gravação. Pode-se observar um exemplo na Figura 2.

Figura 1 – Diagrama de blocos de um sistema de reconhecimento de fala



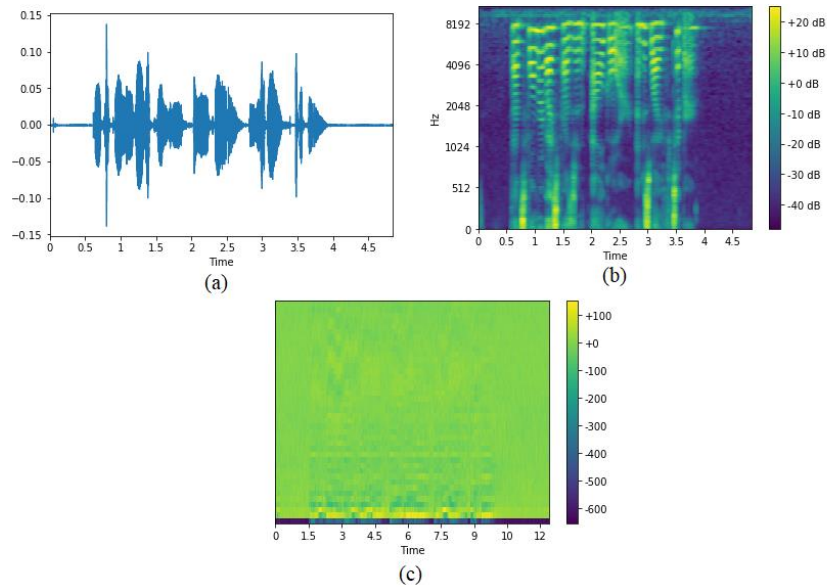
Fonte: Produção do próprio autor.

### 2.3.1 Coeficientes Mel-Cepstrais

De acordo com Nassif e outros (2019), Coeficientes Mel-Cepstrais (*MFCCs*) são o tipo de características mais populares extraídas a partir da fala, sendo utilizado por 69,5% dos artigos estudados. Introduzido por Davis e Mermelstein (1980), *MFCCs* são componentes principais de sistemas modernos de reconhecimento automático de fala.

O objetivo dos coeficientes cepstrais é extrair características a respeito do trato vocal da fonte de excitação, pois as características do trato vocal contêm informações sobre a dicção dos fonemas. Os coeficientes de ordem mais baixa estão mais fortemente relacionados ao trato vocal, que possui variação espectral mais suave. Já os coeficientes de ordem superior são correspondentes a pontos de maior excitação, provenientes de características de maior frequência, relacionadas ao tom de voz utilizado.

Figura 2 – Extração de características a partir de sinal de áudio (a) Sinal original; (b) Espectrograma em escala Mel; (c) Coeficientes Mel-Cepstrais



Fonte: Produção do próprio autor.

A forma mais popular de coeficientes cepstrais utiliza a escala de frequência MEL. Esta é baseada na forma como a audição humana distingue entre frequências, sendo obtida experimentalmente por Stevens, Volkman e Newman (1937). Para realizar a conversão entre uma frequência em Hertz ( $F_{hz}$ ) e a sua equivalente em Mel ( $F_{mel}$ ), deve-se utilizar a equação (2.4).

$$F_{mel} = 2295 \log_{10} \left( 1 + \frac{F_{hz}}{700} \right) \quad (2.4)$$

Os passos para a obtenção dos coeficientes Mel-Cepstrais são: (i) realizar a pré filtragem do sinal de entrada para reduzir os ruídos de alta frequência presentes; (ii) dividir o sinal em quadros periódicos com sobreposição, e sobre cada quadro é aplicada uma função janela de *Hamming*; (iii) transformar para o domínio da frequência por meio de uma transformada rápida de Fourier; (iv) realizar o mapeamento das frequências para a escala Mel, conforme visto na Figura 2(b); (v) calcular o logaritmo da magnitude do espectrograma obtido; (vi) aplicar a transformada cosseno discreto para obter os coeficientes Mel-Cepstrais.

## 2.4 Redes Neurais Profundas

Redes neurais profundas são uma classe de métodos de aprendizado de máquina que têm como foco a utilização de arquiteturas baseadas no uso de diversas camadas que realizam transformações não lineares, seguindo uma ordem hierárquica. Dentre os diversos tipos de redes neurais, os que possuem destaque são as redes neurais convolucionais e as recorrentes, que serão explicadas a seguir.

### 2.4.1 Redes Neurais Convolucionais

Esse tipo de rede é caracterizado pela utilização de camadas de convolução, seguidas por funções de ativação e camadas de *pooling*. Essas camadas possuem a propriedade de operar em forma local sobre suas entradas, propiciando assim a equivariância em relação a translações na entrada do modelo.

O elemento principal de uma rede convolucional é a camada de convolução, que recebe o seu nome devido à operação que realiza. A entrada da camada é composta por um tensor tridimensional que possui  $C$  canais, altura  $H$  e largura  $W$ , portanto dimensões  $(CxHxW)$ . A camada ainda possui um conjunto de pesos associados, que recebem a nomenclatura de filtros, cada um com dimensão  $(CxH_f x W_f)$ . O processo de convolução é caracterizado por realizar uma janela deslizante dos filtros sobre a entrada, realizando em cada posição um produto interno para se obter um escalar. Ao se organizarem todos os escalares gerados de acordo com a posição relativa entre o filtro e o tensor de entrada, obtemos um mapa de ativações para cada filtro presente na camada.

As dimensões do mapa de ativações gerado dependem das dimensões da entrada e saída, além de dois hiper-parâmetros: *stride* e *padding*. *Stride* controla a distância entre duas aplicações sucessivas dos filtros durante o processo da janela deslizante. Assim, o *stride* de um significa que os filtros se movem apenas uma posição entre dois produtos internos realizados. *Padding* é a adição de valores nas bordas da entrada, com o objetivo de permitir que o processo de janela deslizante realizado possa aproveitar completamente os dados presentes nas bordas, ou então para manter a mesma dimensão entre os tensores de entrada e saída. Escolhas comuns para o valor utilizado no *padding* incluem zero ou então o valor médio.

Quando aplicados ao reconhecimento automático de fala, modelos convolucionais tiram proveito de sua alta capacidade de extrair características de forma invariante em relação à posição na entrada, obtendo o estado da arte em bases de dados de referência (LI *et al.*, 2019). Trabalhos recentes também apontam que modelos convolucionais podem ser treinados a partir do áudio de entrada sem nenhum processamento específico, alcançando resultados competitivos (ZEGHIDOUR *et al.*, 2018). Apesar de eliminarem a necessidade da maior parte dos subsistemas apresentados que compõe modelos de reconhecimento de fala clássicos, redes convolucionais não possuem mecanismo para realizar a conexão entre predições de caracteres em diferentes instantes de tempo, de modo que é comum a utilização de modelos de linguagem em conjunto para melhorar as predições.

#### 2.4.2 Redes Neurais Recorrentes

Redes neurais recorrentes se diferenciam das redes convolucionais por serem criadas especificamente para operar sob uma sequência de vetores. Para guardar relações entre os passos ao longo do tempo, redes recorrentes utilizam uma representação interna por meio de um vetor de estado oculto, o qual é atualizado por meio de uma função aprendida pelo modelo.

Assim, redes recorrentes podem ser definidas por meio de duas funções, uma delas que realiza o mapeamento entre a entrada e o estado oculto anterior para o estado oculto atual, e a outra que transforma o estado oculto atual na saída do modelo.

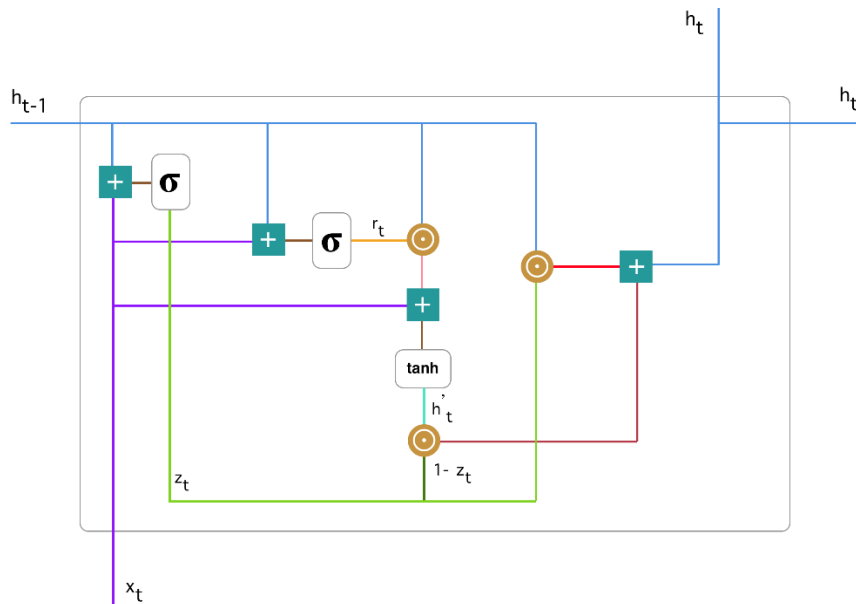
Aplicadas ao reconhecimento de fala, redes recorrentes têm sido utilizadas tanto de forma isolada (CHAN *et al.*, 2015) quanto em conjunto com camadas convolucionais (AMODEI *et al.*, 2016).

#### 2.4.3 Gated Recurrent Unit

Redes *Gated Recurrent Unit* (GRU) são um tipo de rede recorrente com modificações na estrutura de suas camadas. Em uma rede neural GRU, busca-se propagar a entrada no tempo  $t$  assim como o estado da rede no passo anterior através do uso de estruturas denominadas portões.

Como apresentado na Figura 3, o diferencial reside na estrutura interna da camada de um GRU. A mesma é composta por células e portões que trabalham como comportas de dados. Tais estruturas de controle trabalham de forma cooperativa com o objetivo de controlar o fluxo de informação, operando como uma memória, definindo o que a rede deve guardar e o que deve esquecer ao longo do tempo. Especificamente, Uma célula GRU contém dois tipos de portões de controle: (i) portão para decidir qual informação não é relevante, o *Reset Gate* (tradução livre, Portão de esquecimento); (ii) portão para definir quais entradas serão utilizadas para atualizar as células de memória, o *Update Gate* (tradução livre, Portão de atualização).

Figura 3 – Célula básica que compõe rede GRU



Fonte: Kostadinov (2017).

#### 2.4.4 Processamento na célula GRU

As equações que definem a estrutura interna de uma célula GRU são mostradas a seguir:

$$\mathbf{r}_t = \sigma(\mathbf{V}^{(r)}\mathbf{x}_t + \mathbf{U}^{(r)}\mathbf{h}_{t-1}) \quad (2.5)$$

$$\mathbf{z}_t = \sigma(\mathbf{V}^{(z)}\mathbf{x}_t + \mathbf{U}^{(z)}\mathbf{h}_{t-1}) \quad (2.6)$$

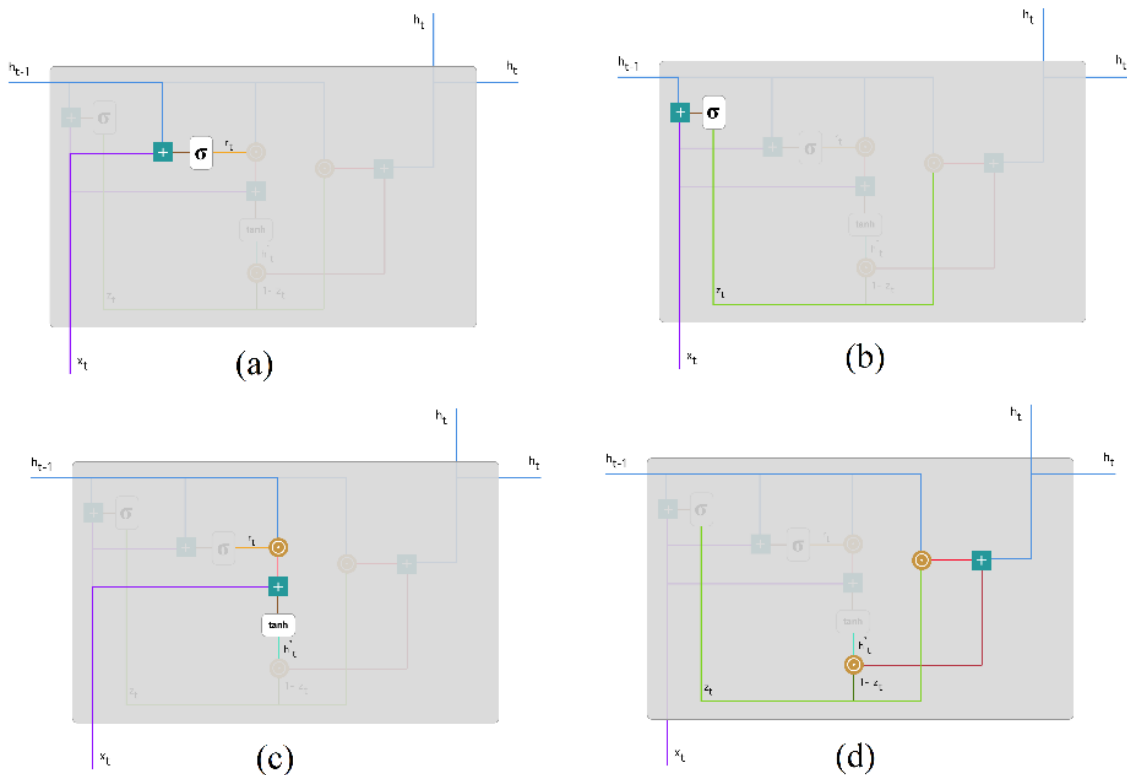
$$\mathbf{h}'_t = \tanh(\mathbf{V}\mathbf{x}_t + \mathbf{r}_t \odot \mathbf{U}\mathbf{h}_{t-1}) \quad (2.7)$$

$$\mathbf{h}_t = \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \mathbf{h}'_t \quad (2.8)$$

Onde:  $\odot$  é o operador do produto *Hadamard*;  $\mathbf{h}_t$ ,  $\mathbf{x}_t$  e  $\mathbf{h}'_t$  são a saída, a entrada e o estado da célula GRU no instante  $t$ , respectivamente;  $\mathbf{z}_t$ ,  $\mathbf{r}_t$ , são as saídas do *Update Gate* e do *Reset Gate*, respectivamente.  $\mathbf{U}$  e  $\mathbf{V}$  são matrizes de pesos.

A seguir uma breve explicação das equações (2.5) - (2.8): a Figura 4(a) mostra o efeito da equação (2.5), que representa o *Reset Gate*. Aqui,  $\mathbf{U}^{(r)}$  permite controlar quanto da informação do estado no tempo anterior influencia o estado atual; A Figura 4(b) mostra o efeito da equação (2.6), que representa o *Update Gate*. Note que a equação é a mesma utilizada para o *Reset Gate*, o que muda aqui são as matrizes e a forma como será utilizado o resultado; A Figura 4(c) mostra o efeito da equação (2.7), que representam a atualização do estado da célula. De forma intuitiva pode-se dizer que: (i) o termo  $\mathbf{r}_t \odot \mathbf{U} \mathbf{h}_{t-1}$  representa a informação que se deseja manter. (ii) o termo  $\mathbf{V} \mathbf{x}_t$  representa a informação que se deseja atualizar; por fim, a Figura 4(d) mostra o efeito da equação (2.8), que representa o *Output Gate*. O valor de  $\mathbf{z}_t$  atua como um filtro que pondera quanto da informação do estado atual será usado para determinar a saída da célula  $\mathbf{h}_t$ .

Figura 4 – Fluxo de informação em uma célula GRU referente ao: (a) *Reset Gate*; (b) *Update Gate*; (c) atualização do estado da célula e (d) *Output Gate*





## 2.5 *Connectionist Temporal Classification*

Bases de dados para aplicações de reconhecimento de fala são compostas por uma sequência de segmentos de áudio que são transformados em vetores acústicos  $\mathbf{X} = [x_1, x_2, \dots, x_T]$  e a sua transcrição correspondente  $\mathbf{Y} = [y_1, y_2, \dots, y_U]$ . Porém, existem alguns problemas ao utilizar os dados em conjunto com técnicas de aprendizado de máquina. Primeiramente, não é conhecido o alinhamento entre os caracteres de saída e a parte correspondente no áudio de entrada, visto que obter tal informação adiciona um grau de complexidade durante o processo de marcação que o torna proibitivo para bases de dados de tamanho considerável. Além disso, ambos  $\mathbf{X}$  e  $\mathbf{Y}$  podem variar em tamanho, assim como a proporção entre o tamanho deles.

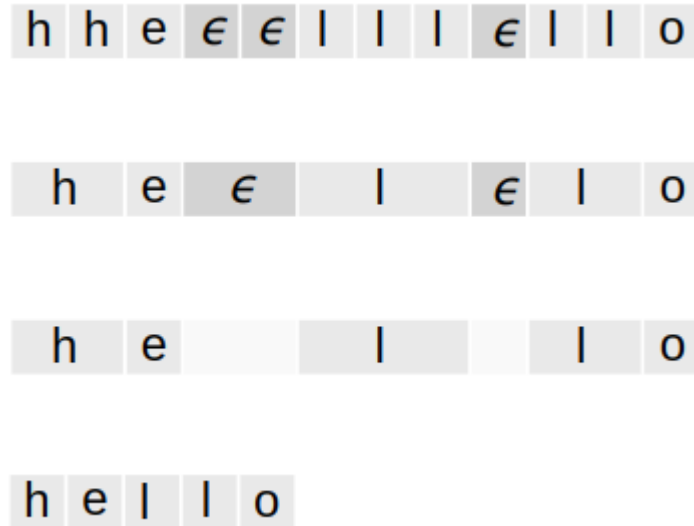
O algoritmo *Connectionist Temporal Classification* (CTC) (GRAVES *et al.*, 2006) é uma forma de lidar com a falta do alinhamento entre a entrada e a saída. Para um dado de entrada  $x$ , ele gera como saída a distribuição entre todas as possibilidades de  $y$ . Pode-se então utilizar essa distribuição para inferir qual é a saída mais provável ou para encontrar a probabilidade de que certa saída específica ocorra. Durante o treinamento, é desejável que o modelo maximize a probabilidade que ele atribui à resposta correta. Utiliza-se o CTC para calcular a probabilidade condicional  $P(\mathbf{Y}|\mathbf{X})$  de forma a ser diferenciável, para poder ser utilizada em conjunto com os algoritmos de otimização baseados em gradiente descendente. Já durante a inferência, deve-se percorrer todas as possibilidades de saída e encontrar aquela que maximiza a probabilidade  $P(\mathbf{Y}|\mathbf{X})$ . Utilizando o CTC é possível encontrar uma solução aproximada em um tempo computacionalmente aceitável.

O algoritmo CTC não depende do alinhamento entre entrada e saída para encontrar a probabilidade de que ocorra uma sequência específica qualquer  $\mathbf{Y}$  dada uma entrada  $\mathbf{X}$ . Porém, na tentativa de encontrar essa probabilidade, o algoritmo realiza a soma das probabilidades de todos os possíveis alinhamentos entre  $\mathbf{X}$  e  $\mathbf{Y}$ . Como são parte fundamental do CTC, o entendimento de como funcionam esses alinhamentos será explicado com mais detalhes a seguir.

Primeiramente, é adicionado ao vocabulário um símbolo  $\epsilon$ , correspondente a uma predição em branco. Dado um vetor de predições, que possui o mesmo tamanho da entrada, são removidos

os caracteres repetidos e, logo após, são removidos os *tokens* especiais  $\epsilon$  para se obter a saída correspondente, como presente na Figura 5.

Figura 5 – Exemplo de decodificação utilizando o algoritmo CTC



Fonte: Hannun (2017).

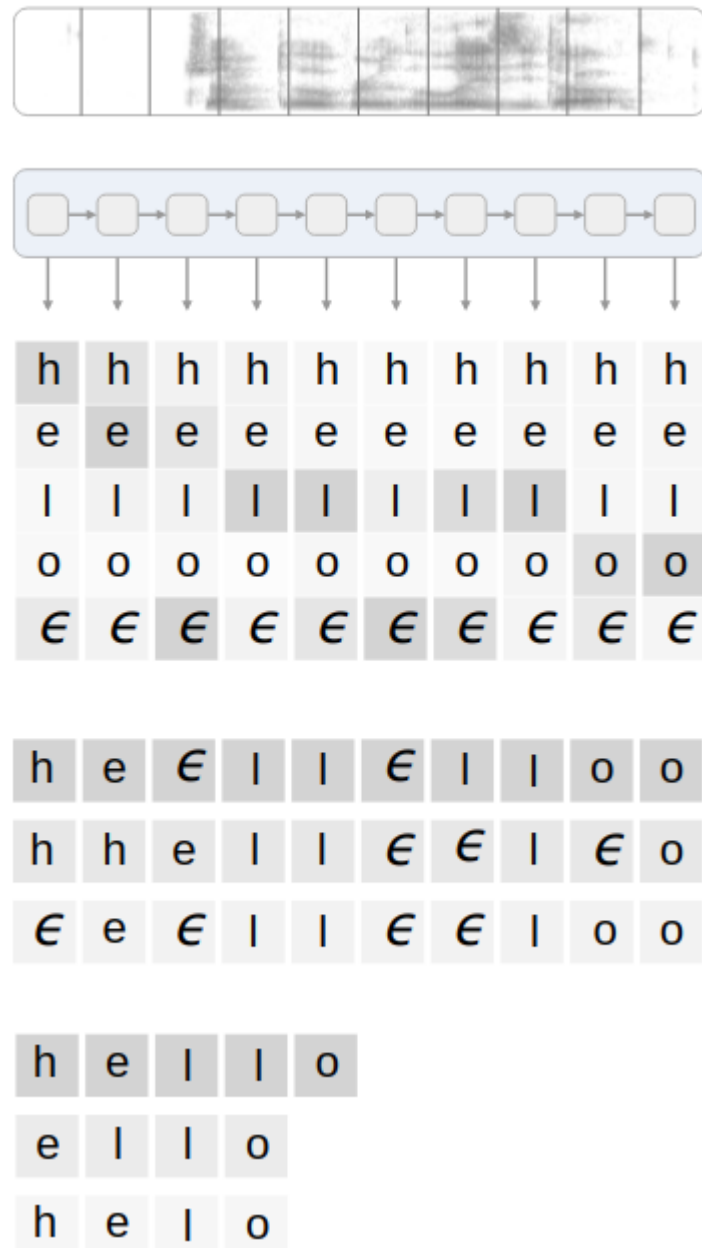
Realizar esse mapeamento entre diferentes alinhamentos para a mesma sequência de rótulos é o que viabiliza ao CTC utilizar dados sem alinhamento prévio, pois ele possibilita que o modelo realize previsões sem saber onde elas ocorrem.

Pode-se observar o processo para o cálculo da probabilidade de sequências específicas a partir de probabilidades distribuídas no tempo na Figura 6. Partindo de uma sequência de entrada, como um espectrograma de áudio, será alimentado um modelo que irá gerar, para cada instante de tempo, uma distribuição  $p_t(\alpha|x)$  que modela a probabilidade da ocorrência de cada um dos caracteres  $\alpha$  pertencentes ao vocabulário. A partir da distribuição de saída gerada para cada instante, é calculada a probabilidade para sequências diferentes, e é realizada a operação de marginalização sobre os diferentes alinhamentos para se obter a distribuição com relação à saída desejada. Matematicamente, essa operação é definida de acordo com a equação (2.9).

$$P(\mathbf{Y}|\mathbf{X}) = \sum_{A \in A_{\mathbf{X},\mathbf{Y}}} \prod_{t=1}^T p_t(\alpha_t|\mathbf{X}) \quad (2.9)$$

Onde  $A_{\mathbf{X},\mathbf{Y}}$  representa o conjunto de alinhamentos entre  $\mathbf{X}$  e  $\mathbf{Y}$ .

Figura 6 – Cálculo da probabilidade de seqüências específicas a partir de probabilidades distribuídas no tempo



Fonte: Hannun (2017).

## 2.6 Decodificação

Após ter treinado o modelo, deseja-se obter a saída correspondente à certa entrada, ou seja, é necessário encontrar a seqüência de saída que maximiza a sua probabilidade condicional em relação à entrada conforme definido na equação (2.10).

$$\mathbf{W}^* = \underset{W}{\operatorname{argmax}} P(\mathbf{W}|\mathbf{X}) \quad (2.10)$$

Uma heurística que pode ser utilizada para encontrar o alinhamento que maximiza a probabilidade de  $\mathbf{W}^*$  é escolher a saída que possui a maior probabilidade em cada instante de tempo e utilizar o algoritmo de decodificação do CTC para encontrar a sequência de saída, utilizando assim um *Greedy Decoder*. Porém, um problema dessa heurística é que ela não leva em conta que uma única saída pode conter diferentes alinhamentos.

Uma extensão que pode ser utilizada para melhorar a predição é a utilização de um modelo de linguagem. Tal modelo é treinado a partir de um *corpus* textual para predizer a probabilidade de que certa palavra específica esteja presente em uma frase tendo em vista as outras palavras presentes naquela frase. Assim, tem como objetivo corrigir problemas de gramática presentes nas predições. Um modelo de linguagem pode ser incluído como fator durante a inferência por meio da modificação presente na equação (2.11).

$$\mathbf{W}^* = \underset{W}{\operatorname{argmax}} P(\mathbf{W}|\mathbf{X}) \cdot P(\mathbf{W})^\gamma \quad (2.11)$$

Aqui,  $P(\mathbf{W})$  é a probabilidade que o modelo de linguagem atribui a certa sequência de saída, e  $\gamma$  é um hiper-parâmetro encontrado experimentalmente.

## 2.7 Resumo

Neste capítulo foram descritos os principais componentes de um sistema de reconhecimento de fala, o processamento que deve ser realizado nos áudios para que sejam extraídas informações no domínio da frequência, além das arquiteturas de redes neurais profundas que compõe os modelos utilizados. No próximo capítulo, será descrito o modelo proposto para a solução do problema de reconhecimento de fala, tomando como base as arquiteturas apresentadas neste capítulo.

## 3 SOLUÇÃO PROPOSTA

### 3.1 Introdução

Neste capítulo é apresentada a solução implementada para a tarefa de reconhecimento automático de fala. Tal proposta está baseada em duas etapas, especificamente:

- **Etapa de pré-processamento dos dados.** Nessa etapa cada áudio e sua transcrição correspondente serão tratados e convertidos para poderem ser utilizados em conjunto com a rede neural.
- **Etapa de reconhecimento.** Aqui, será descrito o modelo utilizado, com a descrição das camadas que compõe sua arquitetura.

Cada etapa será explicada em detalhe nas próximas seções.

### 3.2 Pré-processamento dos dados

Os conjuntos de dados utilizados neste trabalho são compostos por uma sequência de segmentos de áudio e a sua transcrição correspondente. Em busca da maior eficiência computacional, os dados devem ser agrupados em um *batch* para que o processamento ocorra de forma paralela. Porém, devido a limitações computacionais, somente é possível realizar o agrupamento em *batch* de elementos que possuem o mesmo tamanho. Para que seja possível utilizar os dados em conjunto com modelos de aprendizado de máquina, algumas etapas de processamento são necessárias:

- **Criação de vocabulário:** A transcrição dos áudios corresponde a um arquivo de texto para cada áudio. Todavia, como modelos de aprendizado de máquina operam somente com representações numéricas de dados, é necessária a conversão das transcrições da representação textual para numérica. Para isso é criado um vocabulário, onde é armazenado o mapeamento dos caracteres para números inteiros distintos. O vocabulário utilizado neste trabalho possui 44 entradas, sendo as duas primeiras correspondentes ao símbolo especial  $\epsilon$ , utilizado pelo CTC, e ao caractere espaço. Outros 27 mapeamentos correspondem aos caracteres do alfabeto (de a até z), em sua

versão minúscula. Por último, temos 15 entradas para as vogais com acentos e o caractere ç.

- **Leitura dos arquivos de texto:** Após ser realizada a leitura de um arquivo, todo o texto é convertido para caracteres minúsculos. Em seguida, ocorre a quebra do texto nos caracteres individuais que o compõe. O próximo passo é a utilização do vocabulário definido anteriormente, para converter cada caractere em sua representação numérica correspondente. Por fim, é realizado o *padding* para que todas as transcrições possuam o mesmo tamanho, e são armazenados os tamanhos originais de cada transcrição.
- **Leitura dos arquivos de áudio:** Para utilizar os áudios, inicialmente eles deverão ser convertidos de modo que todos tenham a mesma taxa de amostragem. Então, cada áudio é transformado separadamente para a representação no domínio da frequência. Finalmente, o *padding* é realizado diretamente sobre os espectrogramas para que seja possível armazená-los em um *batch*, e os tamanhos originais dos espectrogramas são armazenados

### 3.3 Modelo utilizado para o reconhecimento de fala

Como base para este trabalho, foi utilizado o modelo *DeepSpeech2* (AMODEI *et al.*, 2016). *DeepSpeech2* é uma arquitetura neural *end-to-end* que foi treinada para reconhecer tanto Inglês quando mandarim, sem precisar gerar características específicas para cada tipo de linguagem.

Conforme mostrado na Figura 7, a arquitetura consiste em camadas convolucionais na entrada, seguidas de várias camadas recorrentes e, por fim, uma camada completamente conectada que alimenta o algoritmo *Connectionist Temporal Classification*.

Quadro 1 – Camadas presentes no modelo proposto

Camada	Tamanho do <i>Kernel</i>	Camada oculta	<i>Stride</i>	Filtros	<i>Padding</i>
Conv1	41 x 11		(2, 2)	32	(0, 10)
Conv2	21 x 11		(2, 1)	32	(0, 0)
RNN1-5		800			
Linear	800 x 44				

Fonte: Produção do próprio autor.

Pode-se observar no Quadro 1 as camadas presentes no modelo. As duas primeiras são convolucionais 2D, o que significa que elas conseguem extrair características tanto temporais quanto a respeito da frequência ao mesmo tempo. Cada uma delas possui 32 filtros, e devido ao *stride* maior do que um utilizado, elas possuem a função de reduzir a longa sequência de entrada.

Após cada camada convolucional são inseridas uma camada de normalização por *batch* (IOFFE; SZEGEDY, 2015) e uma função de ativação tangente hiperbólica.

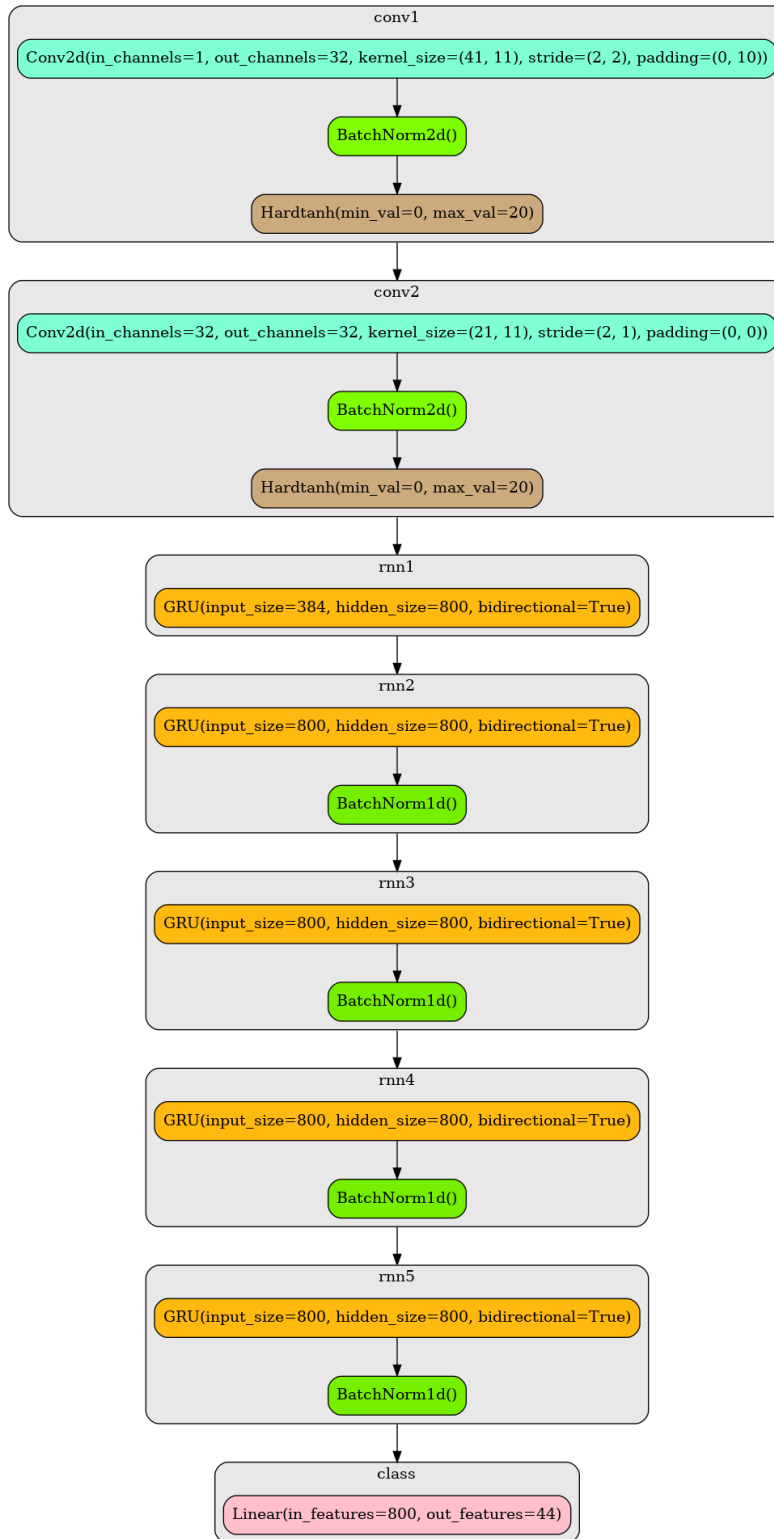
A seguir, são utilizadas cinco camadas recorrentes do tipo GRU, cada uma delas com 800 neurônios em sua camada oculta. Entre as camadas recorrentes é utilizada novamente a normalização por *batch*.

Por fim, é utilizada uma camada linear que leva da dimensionalidade 800 gerada pela camada recorrente para os 44 diferentes *tokens* presentes no vocabulário, para cada instante de tempo. Para isso, é realizada uma manipulação no modo como a camada linear é aplicada, para que a sua operação gere um resultado em cada instante.

### **3.4 Resumo**

Após descrever os passos necessários para o pré-processamento dos dados e a arquitetura do modelo utilizado para o reconhecimento de fala é possível compreender a importância de cada uma das etapas. No próximo capítulo serão descritos os recursos utilizados para a implementação do modelo aqui proposto, além dos experimentos realizados com bases de dados e a sua comparação com os resultados de trabalhos similares.

Figura 7 – Arquitetura do modelo utilizado



Fonte: Produção do próprio autor.



## 4 EXPERIMENTOS E RESULTADOS

### 4.1 Introdução

Neste capítulo serão apresentados os resultados obtidos. Inicialmente, serão listados os recursos computacionais utilizados na solução do problema, tanto os de software quanto hardware, além da base de dados. A seguir, uma explicação das métricas utilizadas será realizada. Após isso, serão descritos os experimentos implementados e os resultados em função das métricas de desempenho definidas na seção anterior. Por fim, é realizada uma comparação com outros resultados presentes na literatura.

### 4.2 Recursos Computacionais

As etapas de treinamento e teste da arquitetura proposta foram realizadas utilizando *Pytorch*, software de código aberto desenvolvido pelo *Facebook*, destinado à programação numérica utilizando programação baseada em fluxo de dados em grafos (PASZKE *et al.*, 2017). Apesar de poder ser utilizado para outros propósitos, *Pytorch* está fortemente pensado para ser utilizado em problemas de *machine learning*, mais especificamente, para aplicações de *deep learning*. Assim, por ter um grande suporte de desenvolvimento, ser livre e possuir a capacidade de fácil utilização de GPU para acelerar o treinamento, está sendo usado como umas das principais ferramentas destinadas a este fim. Além disso, como sua programação é toda baseada na linguagem *Python*, torna o código mais fácil de ser implementado e analisado.

Além disso, a máquina utilizada nos experimentos possuía a seguinte configuração: (i) sistema operacional *Linux*, distribuição *Ubuntu* 18.04; (ii) processador *Intel Core* i7-7700, 3.60GHz com 4 núcleos físicos; (iii) memória RAM de 16 GB; (iv) unidade de armazenamento de 1TB (disco rígido); (v) placa de vídeo *Nvidia Geforce* GTX 1070, com 8 GB de memória dedicada.

Para a realização do treinamento dos classificadores e validação dos resultados, é necessário um conjunto de dados consolidado da literatura. Portanto, foram analisadas os conjuntos de dados mais utilizadas em trabalhos com objetivos semelhantes, chegando à escolha de se utilizar uma combinação entre quatro conjuntos disponíveis abertamente. Três conjuntos de dados são parte do *Brazilian Portuguese Speech Dataset* (BRSD) (QUINTANILHA, 2017), enquanto o

último faz parte do projeto Tatoeba (HO; SIMON, 2016). Enquanto parte dos dados presentes no BRSD tem como origem bases de dados pagas, todos os dados utilizados neste trabalho podem ser obtidos de forma gratuita.

A base de dados Sid contém 72 locutores (20 são mulheres), variando de 17 a 59 anos. Cada locutor tem informações sobre local de nascimento, idade, gênero, educação e ocupação. Todos os áudios foram gravados em 22,05 kHz em ambiente não controlado. Um total de 5.777 falas foram transcritas no nível de palavra sem alinhamento de tempo. As frases variam de dígitos falados, palavras simples, sequências complexas, ortografia do nome e local de nascimento.

A base de dados *Voxforge* (VOXFORGE, 2007) é o corpus mais heterogêneo. A ideia do *VoxForge* é distribuir o áudio da fala transcrita sob licença GPL, facilitando o desenvolvimento de modelos acústicos. A seção em português do Brasil contém pelo menos 111 locutores, porém nem todos têm informações sobre gênero ou idade. Os arquivos de áudio foram gravados em diferentes taxas de amostragem, variando de 16 kHz a 44,1 kHz, e muitos registros estão em baixa qualidade, apresentando baixa relação sinal / ruído (SNR). Um total de 4.130 falas foram transcritas no nível da palavra.

O projeto Tatoeba (HO; SIMON, 2016) teve início como um conjunto de dados online de frases para ser utilizado por estudantes de idiomas estrangeiros. Uma fração das frases disponíveis possui o áudio correspondente, que foi gravado e compartilhado por usuários. Assim, este conjunto é o que possui a maior variabilidade na qualidade dos áudios dentre os utilizados. A quantidade de locutores distintos, além da informação sobre o seu gênero ou idade são desconhecidos. Um total de 10.001 frases em português presentes no conjunto de dados possuem áudio associado, porém esse número pode aumentar a medida que novos usuários compartilham mais áudios.

Finalmente, o LapsBM1.4 (UFPA, 2019) é um conjunto de dados usado pelo grupo Fala Brasil da Universidade Federal do Pará e está orientado para avaliação de sistemas de reconhecimento automático de fala. Contém 35 locutores (10 mulheres), cada um com 20 enunciados únicos, totalizando 700 enunciados. O áudio foi gravado em 22,05 kHz sem controle do ambiente

Os dados dos quatro conjuntos de dados foram divididos em três conjuntos distintos: treino, validação e teste. Seguindo o trabalho de Quintanilha (2017), o conjunto LapsBM será utilizado para compor os conjuntos de validação e teste, tendo em vista que ela foi criada especificamente para a avaliação de sistemas. Foi utilizada a divisão recomendada, com os áudios de 20 locutores compondo o conjunto de validação, e 15 locutores para o conjunto de testes.

Todos os outros dados foram utilizados para treinamento, realizando um pré-processamento para corrigir erros nas transcrições e converter todos os áudios para a mesma taxa de amostragem de 16 kHz. Durante a conversão dos áudios para os coeficientes Mel-Cepstrais, foi utilizado um tamanho de janela correspondente a 20 ms, ou 320 amostras.

### 4.3 Métricas

Para a avaliação do sistema proposto neste trabalho, a métrica utilizada será a taxa de erro por caractere (CER). Ela pode ser computada de acordo com a equação (4.1).

$$CER = \frac{S + D + I}{N} \quad (4.1)$$

Onde: S é o número de substituições realizadas, D é o número de deleções, I é o número de inserções e N é o número de caracteres presentes na referência.

Como exemplo temos o cálculo da taxa de erro conforme a frase original e predição presentes na Quadro 2. São realizadas duas operações de inserção, três deleções e uma substituição para transformar a frase predita na original. Assim, como foram 6 operações, e existem 28 caracteres na frase original (espaços também são considerados no cálculo), a taxa de erro equivalente é de 21,42%.

Quadro 2 – Exemplo de operações consideradas no cálculo da métrica

	Frase	Número de Operações
Original	O céu é azul e o sol amarelo	
Predição	Oh céu é azl e oh sol amriloh	
Inserções	Oh céu é <u>azu</u> l e oh sol amar <u>ar</u> iloh	2
Deleções	<u>Oh</u> céu é azul e <u>oh</u> sol amaril <u>oh</u>	3
Substituições	O céu é azul e o sol amare <u>lo</u>	1

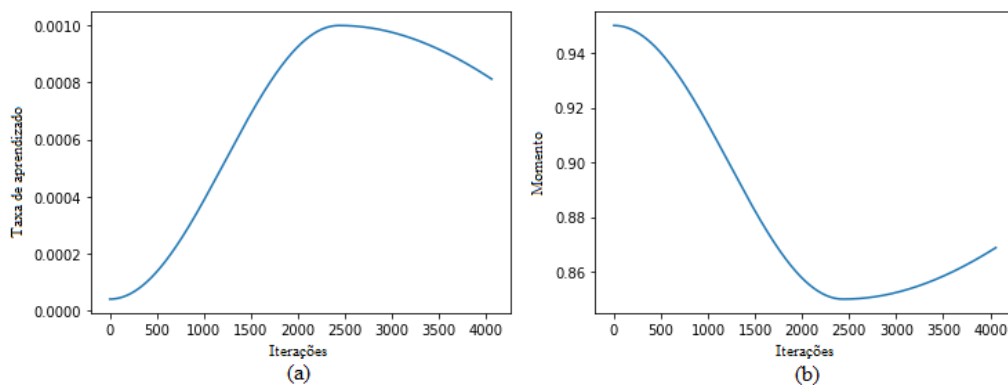
Fonte: Produção do próprio autor.

#### 4.4 Experimento

Foi realizado o treinamento do modelo utilizando o otimizador ADAM (KINGMA; BA, 2015), em conjunto com variações nos hiper-parâmetros taxa de aprendizado e momento conforme presentes em Smith (2018).

No Gráfico 1 é possível visualizar a variação dos hiper-parâmetros utilizada durante o treinamento. O motivo para utilizar tal variação é mitigar os efeitos da alta variância presente nos estágios iniciais de treino de otimizadores adaptativos, evitando assim que várias atualizações baseadas em uma parcela limitada dos dados observados levem o modelo para um mínimo local (LIU *et al.*, 2019).

Gráfico 1 – Variação dos hiper-parâmetros (a) taxa de aprendizado e (b) momento ao longo do treino



Fonte: Produção do próprio autor.

Durante o treino, foi observado o fenômeno de *overfit*, no qual o modelo progressivamente se torna melhor para prever o resultado de dados presentes na base de treino, enquanto a performance medida com outros dados sofre piora. Para reduzir os efeitos do *overfit*, foram utilizados métodos de *data augmentation*, a partir da aplicação de duas transformações ao áudio de forma aleatória. A primeira delas é uma variação na duração total, sendo gerado um novo áudio que possui entre 85% e 115% da duração do áudio original, alterando assim a sua velocidade de reprodução. A outra transformação é uma variação no volume, escolhida aleatoriamente em uma escala que vai de -6db até +8db. Outra forma de regularização do modelo utilizada foi o *early stopping*, onde o treino é finalizado antes que aconteça o *overfit* (CARUANA; LAWRENCE; GILES, 2001).

Os resultados dos experimentos estão expostos na Tabela 1. Durante o treino, foram observados áudios para os quais o modelo não gerava saída. Isso ocorre devido ao ambiente em que foram gravados não ser controlado, sendo que o nível de ruído presente impedia o funcionamento do modelo. Foi detectado esse problema em cerca de 8% dos dados presentes no conjunto de dados, sendo assim foi realizado um segundo treinamento no qual foram descartados esses áudios. Pode-se observar pelos resultados que a remoção dos *outliers* propiciou uma melhora nos resultados obtidos.

Tabela 1 – Resultados para a abordagem proposta, na transcrição de áudios presentes na base de testes

Dados utilizados	CER (%)
Todos	29,844
Removendo <i>outliers</i>	24,790

Fonte: Produção do próprio autor.

No Quadro 3 podem-se observar exemplos de predições realizadas pelo modelo para áudios 36 presentes na base de testes. Apesar de gerar saídas com erros gramaticais, o modelo consegue capturar a estrutura geral do texto. Acompanhada de cada transcrição está a taxa de erro por caractere para aquele exemplo.

Quadro 3 – Exemplos de resultados obtidos

Frase		CER (%)
Original	o treino de hoje será das treze às dezessete horas em Brasília	16,13
Predição	o tremho de hoje será date trze adezessete horas em brazila	
Original	segundo ele a polícia não iria ceder a exigências	8,00
Predição	segundo ele apolícia não iria cebde axgências	

Fonte: Produção do próprio autor.

#### 4.5 Comparação de resultados

Os resultados obtidos foram comparados com o trabalho de Quintanilha (2017), que utilizou a mesma base de dados para teste, porém com a adição de bases de dados pagas nos dados de treinamento. Estão disponíveis ainda, neste trabalho, resultados obtidos para sistemas comerciais, conforme gerado em 15/02/2016. Todos os resultados se encontram na Tabela 2.

Tabela 2 - Comparação com resultados obtidos por outros trabalhos

<b>Trabalho</b>	<b>CER (%)</b>
Proposto	29,844
Quintanilha	25,13
Google API	<b>10,25</b>
IBM Watson	11,38
Microsoft Bing Speech	14,77

Fonte: Produção do próprio autor.

Apesar do resultado obtido estar próximo ao estado da arte obtido em publicações acadêmicas, ele ainda está distante daquele presente em sistemas comerciais que utilizam grande quantidade de dados privados para o treinamento, além de possuírem infraestrutura para treinar modelos maiores e mais complexos.

#### **4.6 Resumo**

Neste capítulo foram apresentados os recursos necessários para a realização deste trabalho, tanto de software quanto computacionais, além do conjunto de dados utilizado. Também foram apresentados os experimentos e as métricas utilizadas para a validação da solução proposta. Por fim foram expostos os resultados dos experimentos e realizada uma comparação entre os resultados deste experimento e um trabalho similar.

No próximo capítulo se encontram as conclusões tiradas deste trabalho e também os temas a serem pesquisados no futuro.

## 5 CONCLUSÕES E PROJETOS FUTUROS

### 5.1 Conclusões

O objetivo principal deste trabalho foi propor uma solução para o reconhecimento automático de fala para a língua portuguesa baseada em quatro conjuntos de dados de domínio público. A técnica proposta está baseada na utilização de um modelo baseado na arquitetura *DeepSpeech2* que gera uma sequência de caracteres a partir de uma representação do áudio.

Foi obtido um resultado próximo ao estado da arte (25,13% de CER), porém problemas relacionados à áudios com baixa relação sinal-ruído presentes no conjunto de dados impediram o funcionamento do modelo em alguns casos.

O trabalho permitiu: (i) comprovar os benefícios de usar um modelo de rede neural para a tarefa de reconhecimento automático de fala; (ii) perceber a dificuldade de se lidar com ruídos não relacionados à fala de interesse presentes no áudio.

### 5.2 Temas a serem pesquisados

Como trabalhos futuros: (i) adicionar um modelo de linguagem à etapa de decodificação dos caracteres para diminuir os erros; (ii) comparar com outras formas de extração de características do áudio, incluindo modelos que operam diretamente sobre a representação temporal; (iii) estudar técnicas para tornar o modelo mais robusto à presença de ruído nos áudios de entrada.

## REFERÊNCIAS BIBLIOGRÁFICAS

- AMODEI, D. *et al.* Deep speech 2: End-to-end speech recognition in english and mandarin. *In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING*, 33, 2016, Nova York. **Proceedings of the 33rd International Conference on Machine Learning**. Nova York: ACM, 2016. p. 173–182.
- CARUANA, R.; LAWRENCE, S.; GILES, C. L. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *In: NEURAL INFORMATION PROCESSING SYSTEMS*, 14, 2001, Vancouver. **Advances In Neural Information Processing Systems**. Vancouver: Neural Information Processing Systems, 2001. p. 402–408.
- CHAN, W.; JAITLY, N.; LE, Q. V.; VINYALS, O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. *In: INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING (ICASSP)*, 41, 2016, Shanghai. **Anais [...]**. Shanghai: IEEE, 2016. p. 4960-4964.
- DAVIS, S.; MERMELSTEIN, P. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. **IEEE Transactions on Acoustics, Speech, and Signal Processing**, v. 28, n. 4, p. 357–366, ago. 1980.
- DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *In: 2019 ANNUAL CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, 2019, Minneapolis. **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)**. Minneapolis: Association for Computational Linguistics, 2019. p. 4171-4186.
- GRAVES, A.; FERNÁNDEZ, S.; GOMEZ, F.; SCHMIDHUBER, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. *In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING*, 23, 2006, Pittsburgh. **Proceedings of the 23rd International Conference on Machine Learning**. Nova York: ACM, 2006. p. 369–376.
- HANNUN, A. **Sequence Modeling with CTC**. 2017. Disponível em: <https://distill.pub/2017/ctc/> Acesso em: 10 set. 2019.
- HO, T.; SIMON, A. **Tatoeba: Collection of sentences and translations**. 2006. Disponível em: <https://tatoeba.org/> Acesso em: 10 set. 2019
- HOWARD, J.; RUDER, S. Universal Language Model Fine-tuning for Text Classification. *In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, 56, 2018, Melbourne. **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)**. Melbourne: ACL, 2018. p. 328-339.



IOFFE, S.; SZEGEDY, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING*, 32, 2015, Lille. **Proceedings of the 32rd International Conference on Machine Learning**. Nova York: ACM, 2015. p. 448–456

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS*, 3, 2015, San Diego. **3rd International Conference on Learning Representations, Conference Track Proceedings**. San Diego: ICLR, 2015. p. 108-123.

KOSTADINOV, S. **Understanding GRU Networks**. 2017. Disponível em: <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>. Acesso em: 10 set. 2019.

LI, J.; LAVRUKHIN, V.; GINSBURG, B.; LEARY, R.; KUCHARIEV, O.; COHEN, J. M.; NGUYEN, H.; GADDE, R. T. **Jasper: An end-to-end convolutional neural acoustic model**. 2019. Disponível em: <https://arxiv.org/pdf/1904.03288.pdf>. Acesso em: 10 set. 2019.

LIU, L.; JIANG, H.; HE, P.; CHEN, W.; LIU, X.; GAO, J.; HAN, J. **On the variance of the adaptive learning rate and beyond**. 2019. Disponível em: <https://arxiv.org/pdf/1908.03265.pdf>. Acesso em: 10 set. 2019.

MAHAJAN, D.; GIRSHICK, R.; RAMANATHAN, V.; HE, K.; PALURI, M.; LI, Y.; BHARAMBE, A.; MAATEN, L. Exploring the limits of weakly supervised pretraining. *In: EUROPEAN CONFERENCE ON COMPUTER VISION*, 15, 2018, Munique. **Proceedings of the European Conference on Computer Vision (ECCV)**. New York: Springer, 2018. p. 181–196.

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

PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DEVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; LERER, A. Automatic differentiation in PyTorch. *In: NEURAL INFORMATION PROCESSING SYSTEMS*, 31, 2017, Long Beach. **NeurIPS Autodiff Workshop**. Long Beach: Neural Information Processing Systems, 2017.

QUINTANILHA, I. M. **End-to-End Speech Recognition Applied to Brazilian Portuguese Using Deep Learning**. 2017. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Engenharia Elétrica/COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2017.

SCHNEIDER, S.; BAEVSKI, A.; COLLOBERT, R.; AULI, M. wav2vec: Unsupervised Pre-Training for Speech Recognition. *In: ANNUAL CONFERENCE OF THE INTERNATIONAL SPEECH COMMUNICATION ASSOCIATION*, 20, 2019, Graz. **Proceedings Interspeech 2019**. Graz: Interspeech, 2019. p. 3465–3469.

SMITH, L. N. **A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay**. 2018. Disponível em: <https://arxiv.org/pdf/1803.09820.pdf>. Acesso em: 10 set. 2019.

STEVENS, S. S.; VOLKMANN, J.; NEWMAN, E. B. A scale for the measurement of the psychological magnitude pitch. **The Journal of the Acoustical Society of America**, v. 8, n. 3, p. 185–190, 1937.

UFPA, F. **Recursos livres para ASR em Português Brasileiro**. 2019. Disponível em: <https://ufpafalabrasil.gitlab.io/res/asr/>. Acesso em: 10 set. 2019.

VOXFORGE. **Voxforge**. 2006. Disponível em: <http://www.voxforge.org/>. Acesso em: 10 set. 2019.

ZEGHIDOUR, N.; XU, Q.; LIPTCHINSKY, V.; USUNIER, N.; SYNNAEVE, G.; COLLOBERT, R. **Fully convolutional speech recognition**. 2018. Disponível em: <https://arxiv.org/pdf/1812.06864.pdf>. Acesso em: 10 set. 2019.