UNIVERSIDADE DO ESTADO DO AMAZONAS – UEA
ESCOLA SUPERIOR DE TECNOLOGIA – EST
ENGENHARIA ELÉTRICA

MAURICIO SOUZA CALHEIRO

CLASSIFICAÇÃO DE ANOMALIAS EM SONS RESPIRATÓRIOS UTILIZANDO PROCESSAMENTO DIGITAL DE SINAIS DE ÁUDIO E REDES NEURAIS ARTIFICIAIS

Manaus, AM
2021

MAURICIO SOUZA CALHEIRO

CLASSIFICAÇÃO DE ANOMALIAS EM SONS RESPIRATÓRIOS UTILIZANDO PROCESSAMENTO DIGITAL DE SINAIS DE ÁUDIO E REDES NEURAIS ARTIFICIAIS

Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletricista.

Orientador: Prof. Dr. Edgard Luciano Oliveira da Silva

Manaus, AM

2021

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Reitor:
Cleinaldo de Almeida Costa
Vice-Reitor:
Cleto Cavalcante de Souza Leal
Diretora da Escola Superior de Tecnologia:
Ingrid Sammyne Gadelha Figueiredo
Coordenador do Curso de Engenharia Elétrica
Israel Gondres Torné

Banca Avaliadora composta por: *Data da defesa: 29/12/2021.*
Prof. Edgard Luciano Oliveira da Silva *(Orientador)*
Prof. Bruno da Gama Monteiro
Prof. Jozias Parente de Oliveira

CIP – Catalogação na Publicação

C152 Calheiro, Mauricio Souza

Classificação de anomalias em sons respiratórios utilizando processamento digital de sinais de áudio e redes neurais artificiais / Mauricio Souza Calheiro; [orientado por] Edgard Luciano Oliveira da Silva. – Manaus: 2021.
60 p.: il.

Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica). Universidade do Estado do Amazonas, 2021

1. Sons pulmonares. 2. Sinais de áudio. 3. Rede neural convolucional. I. Silva, Edgard Luciano Oliveira da.

MAURICIO SOUZA CALHEIRO

CLASSIFICAÇÃO DE ANOMALIAS EM SONS RESPIRATÓRIOS UTILIZANDO PROCESSAMENTO DIGITAL DE SINAIS DE ÁUDIO E REDES NEURAIS ARTIFICIAIS

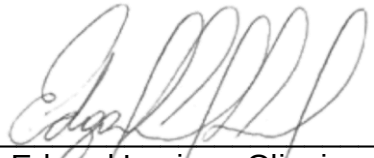
Pesquisa desenvolvida durante a disciplina de Trabalho de Conclusão de Curso II e apresentada à banca avaliadora do Curso de Engenharia Elétrica da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Engenheiro Eletricista.

Nota obtida: 9,6 (nove vírgula seis)

Aprovada em 29/12/2021

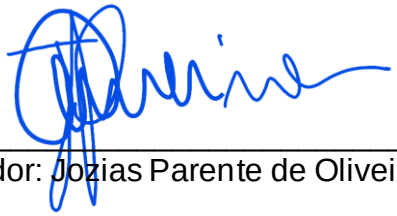
Área de concentração: Processamento digital de sinais

BANCA EXAMINADORA



Orientador: Edgard Luciano Oliveira da Silva, Dr.

Avaliador: Bruno da Gama Monteiro, Mr.



Avaliador: Jozias Parente de Oliveira, Dr.

Manaus, 2021

DEDICATÓRIA

Ao tio Paulo.

AGRADECIMENTOS

Agradeço a meus professores, que fizeram seu melhor para transmitir conhecimento e expandiram meus horizontes.

A meus amigos com quem dividi memoráveis dias na universidade, pelo suporte nos seminários e projetos que desenvolvemos juntos.

A minha família pelo amor, apoio e incentivo a minha educação acadêmica e profissional.

A minha amada esposa por toda a paciência e incentivo em tudo que busco. Por se permitir realizar sonhos comigo.

A Deus, pelo livre-arbítrio e pela capacidade de adquirir conhecimento.

RESUMO

Nesta pesquisa foi elaborado um modelo de aprendizado de máquina para classificar sons pulmonares, com a finalidade de identificar a presença de sons anormais contínuos (sibilos) e descontínuos (estertores). Para tal, utilizou-se técnicas de processamento digital de sinais para processar e extrair características dos sinais de áudio (MFCCs e espectrogramas em escala Mel) e uma rede neural convolucional para realizar a classificação em dois cenários. Os dados utilizados foram obtidos de um *dataset* distribuído gratuitamente pela internet. No primeiro cenário, preocupou-se em classificar os sons presentes nos ciclos respiratórios, e o melhor resultado obtido foi a detecção de estertores, com acurácia de 82%. O segundo cenário consiste na classificação das comorbidades dos pacientes, onde obteve-se 94% de acurácia e F1-score (média macro). Os resultados mostram que é possível realizar a classificação com precisão apenas para algumas classes, o que indica que é preciso refinamento do método ou dos dados utilizados.

Palavras-chave: sons pulmonares, sinais de áudio, rede neural convolucional

ABSTRACT

In this research, a machine learning model was developed to classify lung sounds, with the purpose of identifying the presence of continuous (wheezes) and discontinuous (crackles) abnormal sounds. Digital signal processing techniques were used to process audio signals and extract its features (MFCCs and Mel scale spectrograms) and a convolutional neural network was used to perform the classification in two scenarios. The data used were obtained from a dataset distributed free of charge on the internet. The first scenario is concerned with classifying the sounds present in the respiratory cycles, and the best result obtained was the detection of crackles, with 82% accuracy. The second scenario consists in classifying the patients' comorbidities, where 94% of accuracy and F1-score (macro average) were obtained. The results show that it is possible to perform the classification with precision only for some classes, which indicates that it is necessary to refine the method or data used.

Keywords: lung sounds, audio signals, convolutional neural network

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS

ADC	Conversor Analógico-Digital (*Analog-to-Digital Converter*)
ANN	Rede Neural Artificial (*Artificial Neural Network*)
CNN	Rede Neural Convolucional (*Convolutional Neural Network*)
DAC	Conversor Digital-Analógico (*Digital-to-Analog Converter*)
DCT	Transforma Discreta de Cosseno (*Discrete Cosine Transform*)
DFT	Transformada Discreta de Fourier (*Discrete Fourier Transform*)
DPOC	Doença Pulmonar Obstrutiva Crônica
FFT	Transformada Rápida de Fourier (*Fast Fourier Transform*)
FT	Transformada de Fourier (*Fourier Transform*)
GPU	Unidade de Processamento Gráfico (*Graphics Processing Unit*)
KNN	k-vizinhos mais próximos (*k-Nearest Neighbors*)
LTU	Unidade Linear com *Threshold* (*Linear Threshold Unit*)
MFCCs	Componentes Mel-Cepstrais (*Mel-Frequency Cepstral Coefficients*)
ML	Aprendizado de Máquina (*Machine Learning*)
MPL	*Perceptron* Multicamada (*Multilayer Perceptron*)
RNN	Rede Neural Recorrente (*Recurrent Neural Network*)
SVM	Máquinas de Vetores de Suporte (S*upport Vector Machine*)

SUMÁRIO

INTRODUÇÃO

Os pulmões são alguns dos órgãos mais vitais no corpo humano, pois são componentes chave do sistema respiratório. Sendo assim, a saúde pulmonar é de interesse de qualquer indivíduo. Doenças respiratórias, como as infecções de vias aéreas inferiores e a doença pulmonar obstrutiva crônica (DPOC), estão listadas entre as maiores causas de morte e deficiências no mundo. Sozinhas, as dez primeiras enfermidades listadas causaram mais que a metade dos óbitos nas últimas décadas no Brasil (SECRETARIA DE VIGILÂNCIA EM SAÚDE, 2017) e no mundo (WORLD HEALTH ORGANIZATION, 2020).

Partindo da hipótese que é possível processar sinais de áudio provenientes do pulmão e extrair suas características para classificar as anomalias presentes nos sons utilizando redes neurais artificiais, busca-se desenvolver um sistema para classificação de sons pulmonares. Dessa forma, esta pesquisa contempla a extração de características de sinais sonoros utilizando técnicas de processamento digital de sinais e elaboração de um modelo de aprendizado de máquina utilizando redes neurais convolucionais.

Esta pesquisa se justifica pela possibilidade de tornar o exame de ausculta pulmonar menos subjetivo por meio da classificação automática dos sons pulmonares, levando a diagnósticos mais precisos. Estudos anteriores já mostraram que técnicas de processamento digital de sinais e as redes neurais podem ser combinados para realizar essa tarefa de classificação (PALANIAPPAN, R.; SUNDARAJ, K.; AHAMED, N. U, 2013) e (SHARMA, G.; UMAPATHY, K.; KRISHNAN, S, 2019).

A composição deste estudo é dada em quatro capítulos. No referencial teórico são explorados os diversos temas necessários para a realização da pesquisa. Ele contém dados que embasam o que foi dito no primeiro parágrafo e uma breve descrição dos sons pleuropulmonares, seguido de sinais de áudio e suas características. Ao final, as redes neurais artificiais são exploradas, com foco nas convolucionais. Em seguida vem a metodologia, onde são explanadas as etapas, bem como os recursos necessários para o desenvolvimento. A implementação mostra como os dados foram processados e a elaboração do modelo, assim como todas as etapas necessárias para treiná-lo e fazer estimativas. Por fim, o capítulo de resultados e discussões apresenta a análise dos dados obtidos a partir das previsões realizadas.

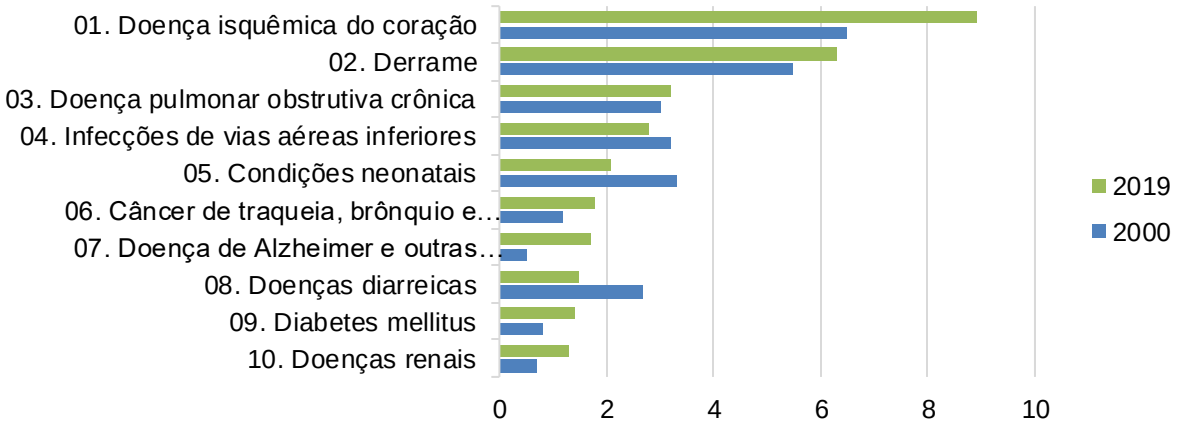
1 REFERENCIAL TEÓRICO

Nesta seção serão apresentados os conceitos, levantados através de pesquisa exploratória, que são necessários para o desenvolvimento da pesquisa. Os temas abordados neste estudo compreendem a ausculta de sons pulmonares, o processamento digital de sinais sonoros e redes neurais artificiais.

1.1 PANORAMA DAS DOENÇAS RESPIRATÓRIAS

Nas últimas décadas (2000 – 2019), doenças respiratórias estiveram entre as maiores causas de mortes no mundo. Somente em 2019, as 10 maiores causas listadas foram responsáveis por mais da metade dos óbitos no planeta (WORLD HEALTH ORGANIZATION, 2020). A mostra que a doença pulmonar obstrutiva crônica (DPOC) e as infecções de vias aéreas inferiores estão presentes do início ao final da década.

Figura 1: Principais causas de morte no mundo (em milhões)



Fonte: adaptado de (WORLD HEALTH ORGANIZATION, 2020)

Apesar de disponibilizados somente até 2017, os dados do Departamento de Análise em Saúde e Vigilância das Doenças Não Transmissíveis (DASNT) da Secretaria de Vigilância em Saúde (SVS), representados no Quadro 1, mostram que a situação no Brasil é similar ao cenário global.

Quadro 1: Principais causas de morte no Brasil entre 2000 e 2017.

2000	2010	2017
01 D isquêmica do coração	01 D isquêmica do coração	01 D isquêmica do coração
02 D cerebrovascular	02 D cerebrovascular	02 D cerebrovascular
03 D pulmonar obstrutiva crônica	**03 Infecção das vias aéreas inferiores**	**03 Infecção das vias aéreas inferiores**
04 Infecção das vias aéreas inferiores	04 Alzheimer e outras demências	04 Alzheimer e outras demências
05 Alzheimer e outras demências	**05 D pulmonar obstrutiva crônica**	**05 D pulmonar obstrutiva crônica**

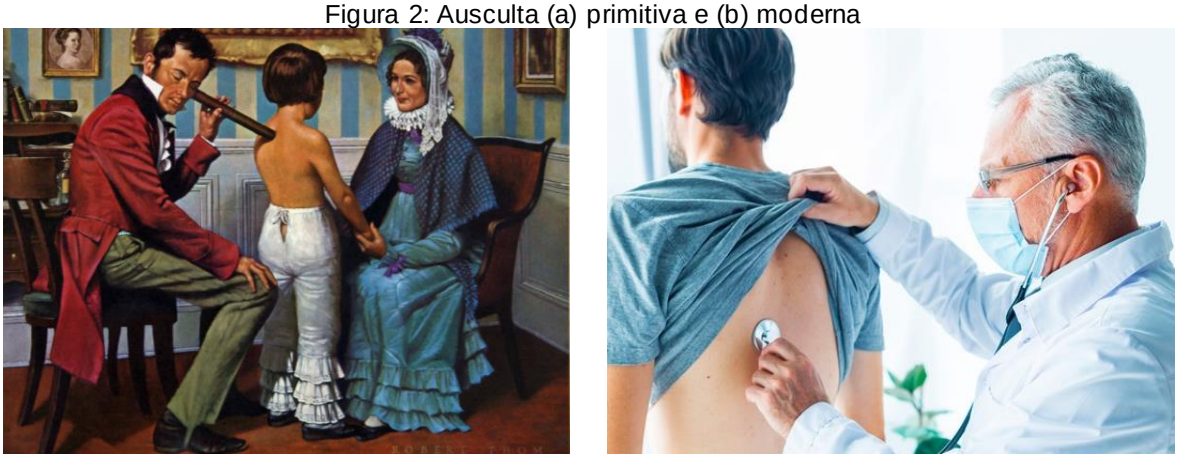
06 Transtornos do período neonatal	06 Violência interpessoal	06 Violência interpessoal
07 Violência interpessoal	07 Diabetes mellitus	07 Diabetes mellitus
08 Diabetes mellitus	08 Acidentes de trânsito	08 Acidentes de trânsito
09 Acidentes de trânsito	09 Transtornos do período neonatal	09 D renal crônica
10 Cirrose e outras ds hepáticas crônicas	10 Cirrose e outras ds hepáticas crônicas	10 Cirrose e outras ds hepáticas crônicas

Fonte: adaptado de (SECRETARIA, 2017).

1.2 AUSCULTA

A ausculta pulmonar (Figura 2) é um dos exames mais comuns ao qual um paciente é submetido durante uma consulta com o clínico geral ou médico de família, principalmente em casos de suspeita de doenças respiratórias. É um procedimento rápido, de fácil execução (não invasivo) e não requer tecnologia avançada – sendo comumente realizado com estetoscópios analógicos. No entanto, os resultados desse tipo de exame são de natureza subjetiva, fortemente dependentes da experiência e das habilidades perceptivas do examinador e, portanto, sujeitos a erros. De fato, a habilidade de ausculta pulmonar de um médico generalista ou médico de família não é satisfatoriamente superior à de um interno de medicina, com exceção dos pneumologistas, cujos resultados são estatisticamente superiores (HAFKE-DYS, *et al.*, 2019). Dadas essas limitações, existe o interesse em automatizar a análise de sons pulmonares.

Figura 2: Ausculta (a) primitiva e (b) moderna



(a) (b)

Fonte: (a) (THOM, 1960); (b) (SANAR RESIDÊNCIA MÉDICA)

O estetoscópio (Figura 3) foi inventado por René Laennec há mais de 200 anos e já sofreu diversas alterações em seu formato e em sua composição antes de atingir sua forma atual. Os novos materiais empregados na construção dos estetoscópios modernos melhoraram significativamente sua capacidade de conduzir som (PRIFTIS,

HADJILEONTIADIS e EVERARD, 2018). Ainda assim, seu uso ainda se dá da mesma forma que nas primeiras versões: é um instrumento passivo que serve como meio de propagação sonora entre o interior do paciente e os ouvidos do examinador.

Figura 3: (a) Estetoscópio de Laennec e (b) Littmann Cardiology IV

(a) (b)

Fonte: (a) (Laennec's stethoscope); (b) (3M Littmann Cardiology IV)

1.3 SONS PLEUROPULMONARES

Os sons pleuropulmonares são aqueles provenientes da região toráxica. Muitos deles são normais, enquanto alguns indicam comorbidades no sistema respiratório. Os tipos de sons pleuropulmonares são apresentados no Quadro 2.

Quadro 2: Sons pleuropulmonares

Sons normais	Sons anormais
▪ Som traqueal ▪ Respiração brônquica ▪ Respiração broncovesicular ▪ Murmúrio vesicular	▪ Descontínuos: estertores finos e grossos ▪ Contínuos: roncos, sibilos e estridor ▪ De origem pleural: atrito pleural

Fonte: adaptado de (PORTO e PORTO, 2016)

1.3.1 Sons normais

O som traqueal é audível na região de projeção da traqueia, no pescoço e na região esternal e origina-se na passagem do ar através da fenda glótica e na traqueia. Já a respiração brônquica é audível na zona de projeção dos brônquios de maior calibre, na face anterior do tórax e nas proximidades do esterno. Ambos são sons similares, e sua principal diferença é que o componente expiratório da respiração brônquica é menos intenso.

O murmúrio vesicular constitui-se pelos sons ouvidos na maior parte do tórax, que são produzidos quando o ar circulante ao chocar-se contra as saliências das bifurcações brônquicas ao passar por cavidades de tamanhos distintos, como bronquíolos e alvéolos. Seu componente inspiratório é mais intenso e duradouro e de tom mais alto que o expiratório. Apesar de poder ser detectado em quase todo o tórax,

não é ouvido de maneira uniforme, uma vez que sua intensidade pode variar dependendo da espessura da parede toráxica (PORTO e PORTO, 2016).

1.3.2 Sons anormais descontínuos

Os sons anormais descontínuos são os chamados estertores (Figura 4), que podem ser audíveis durante todo o ciclo respiratório como ruídos sobrepostos aos sons respiratórios normais. Classificam-se como finos ou grossos (Quadro 3).

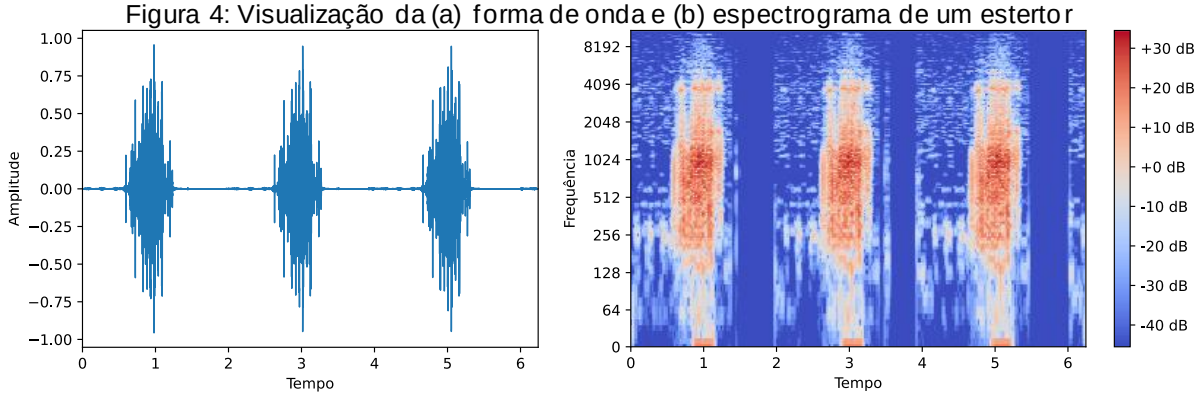
Quadro 3: Principais características dos estertores

Tipo	Fase do ciclo respiratório	Efeito da tosse	Efeito da posição do paciente	Áreas em que predominam
Estertor fino	Final da inspiração	Não se alteram	Modificam-se ou são abolidos	Bases pulmonares
Estertor grosso	Início da inspiração e toda a expiração	Alteram-se	Não se modificam	Todas as áreas do tórax

Fonte: adaptado de (PORTO e PORTO, 2016)

Os estertores finos são sons agudos, que possuem alta frequência e curta duração – menor que 100 ms (PALANIAPPAN, R.; SUNDARAJ, K.; AHAMED, N. U, 2013). Já os estertores grossos são sons de menor frequência e maior duração (com relação aos finos) que não se modificam com a tosse e podem ser ouvidos durante o início da inspiração e por toda a expiração.

Figura 4: Visualização da (a) forma de onda e (b) espectrograma de um estertor



(a) (b)

Fonte: o autor (2021)

Os estertores finos estão presentes na pneumonia, na congestão pulmonar da insuficiência ventricular esquerda e nas doenças intersticiais pulmonares, enquanto os estertores grossos são características das bronquites e bronquiectasias (PORTO e PORTO, 2016).

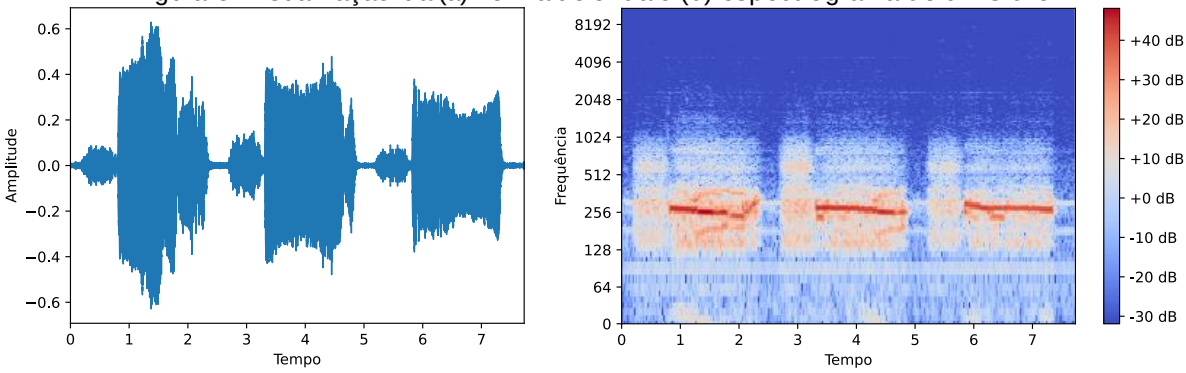
1.3.3 Sons anormais contínuos

Conforme verifica-se no Quadro 2, os sons anormais contínuos contêm os roncos, sibilos e estridores.

Os roncos são sons graves, de baixa frequência, que se originam nas vibrações das paredes brônquicas. Também ocorrem na inspiração, mas predominam na expiração e estão associados a asma brônquica, bronquites, bronquiectasias e em obstruções localizadas.

Os sibilos (Figura 5) se originam da mesma forma, mas possuem características sonoras distintas: são agudos e de alta frequência. Sua duração é tipicamente maior que 250 ms (PALANIAPPAN, R.; SUNDARAJ, K.; AHAMED, N. U, 2013). Além disso, são múltiplos podem ser disseminados por todo o tórax quando causados por enfermidades que acometem toda a árvore brônquica, tais como a asma e a bronquite (PORTO e PORTO, 2016).

Figura 5: Visualização da (a) forma de onda e (b) espectrograma de um sibilo



(a) (b)

Fonte: o autor (2021)

Por fim, o estridor é um ruído inspiratório causado pela obstrução da laringe ou da traqueia. Ele pode ser provocado por difteria, laringe aguda, câncer de laringe e estenose da traqueia. Sua intensidade varia conforme a respiração, sendo maior em na respiração forçada devido ao aumento do fluxo de ar (PORTO e PORTO, 2016).

1.4 SINAIS DE ÁUDIO

Um sinal é aquilo que utilizamos para representar uma quantidade que varia no tempo (GRUS, 2016). Uma vez que este trabalho se preocupa em estudar sinais de natureza sonora, é interessante entender o que é o som e como podemos representá-lo de forma digital.

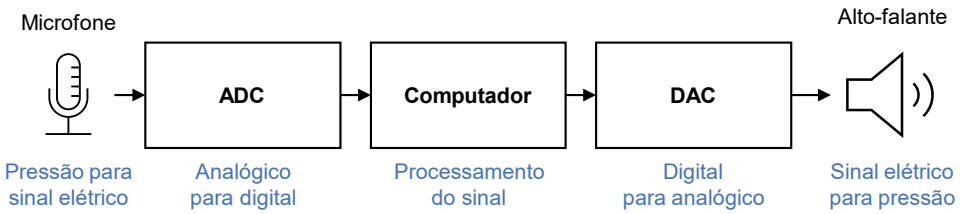
Som é uma das formas mais importantes de interagirmos com o mundo que nos cerca. É essencial para a comunicação da maioria dos seres humanos entre si, para expressar emoções através da música e até mesmo para nos orientarmos no ambiente, sendo tipicamente compreendido como uma sensação auditiva ou uma perturbação em um meio que causa tal sensação. Do ponto de vista físico, refere-se às ondas que se original em um ponto do espaço e viajam por um meio (sólido, líquido ou gasoso) para outro ponto do espaço, onde pode ser ouvido ou medido por algum instrumento (CHRISTENSEN, 2019).

A conversão de energia de uma forma para outra é realizada por dispositivos transdutores. A exemplo: um microfone é um transdutor cuja função é converter a variação de pressão em sinal elétrico. Para fazer o caminho inverso e reconstruir o sinal sonoro a partir do sinal elétrico, utiliza-se o alto-falante. Ambos estão representados na Figura 6, como as partes inicial e final de um exemplo sistema de processamento de áudio.

1.4.1 Sinal de áudio digital

Como tudo na natureza, o som é uma grandeza analógica, sendo possível medi-lo em qualquer instante. Deste modo, o microfone gera um sinal elétrico contínuo. Os computadores, por sua vez, são sistemas digitais e só possuem capacidade para armazenar quantidades finitas de valores. Isso implica na necessidade de uma interface entre o computador e o microfone. A Figura 6 exemplifica os componentes necessários para processar um sinal de áudio por meio de um computador.

Figura 6: Esquema de um sistema de processamento de áudio

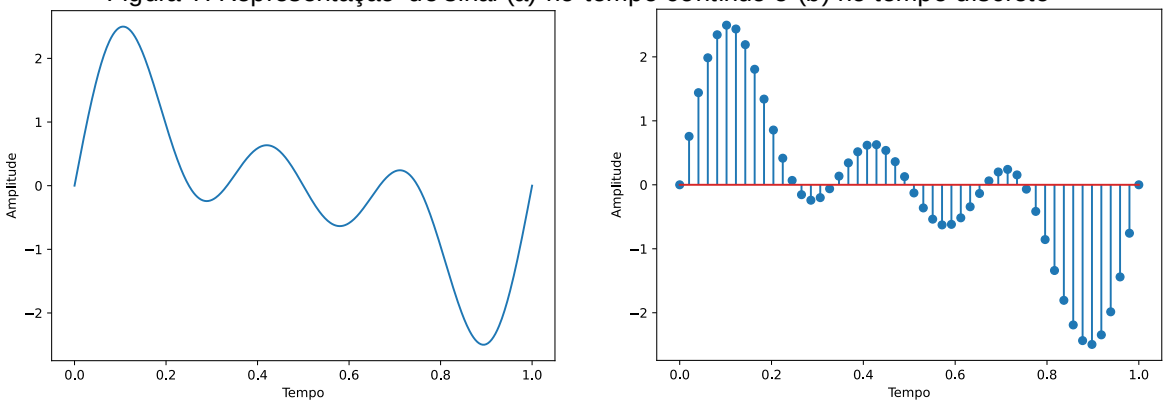


Fonte: adaptado de (CHRISTENSEN, 2019)

A ponte entre os mundos analógico e o digital é o conversor analógico-digital (ADC – *Analog-to-Digital Converter*), que é um dispositivo que realiza a amostragem e quantização do sinal analógico (Figura 7a) e o transforma em sinal digital. Devido à natureza de nossos sistemas auditivos, para ouvir o som após processá-lo em um computador é necessário converter o sinal digital em analógico. Esta operação de

reconstrução é realizada pelo conversor digital-analógico (DAC – *Digital-to-Analog Converter*).

Figura 7: Representação de sinal (a) no tempo contínuo e (b) no tempo discreto



(a) (b)

Fonte: o autor (2021)

O ADC entrega ao computador um sinal no tempo discreto (Figura 7b), que é aquele que pode ser representado por uma sequência numérica (DINIZ, DA SILVA e NETTO, 2014). Duas etapas importantes são necessárias para que isso aconteça: amostragem e quantização.

1.4.2 Amostragem

O processo de amostragem uniforme (ou periódica) de um sinal consiste em medir seu valor em intervalos regulares (OPPENHEIM e SCHAFER, 2012). Seja o sinal da Figura 7a dado por $x(t)$, definido para qualquer valor real de t, visto que se trata de um sinal no tempo contínuo. A amostragem deste sinal se dá através da medição em instâncias de tempo t_n, conforme a Equação (1). Cada instância de tempo t_n é definida por $t_n = T_s n$, onde T_s é o período de amostragem – tempo (em segundos) entre amostras consecutivas.

$$x_n = x(t_n), \qquad n = 0, 1, 2, \ldots \tag{1}$$

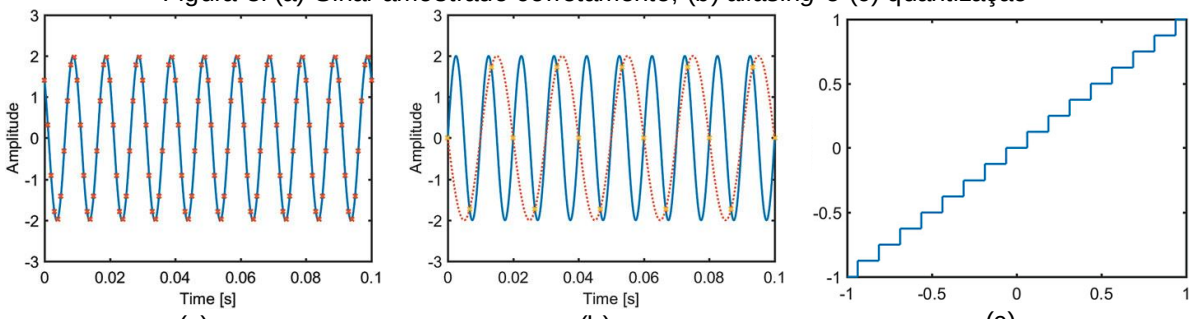
Para recuperar um sinal amostrado, ou seja, representá-lo novamente no tempo contínuo, é importante que este possua características similares ao sinal original (Figura 8a). Isso acontece quando o teorema da amostragem, Equação (2), é obedecido.

$$f_s > 2f_{max} \tag{2}$$

O teorema de amostragem diz que a taxa de amostragem f_s deve ser maior que o dobro da componente de maior frequência no sinal medido, f_{max}, que também

é conhecida como frequência de Nyquist em homenagem a um dos criadores do teorema. Quando a frequência de amostragem não obedece ao teorema, ocorre o fenômeno de *aliasing*, no qual o sinal amostrado não representa o original (Figura 8b).

Figura 8: (a) Sinal amostrado corretamente, (b) *aliasing* e (c) quantização



(a) (b) (c)

Fonte: (CHRISTENSEN, 2019)

1.4.3 Quantização

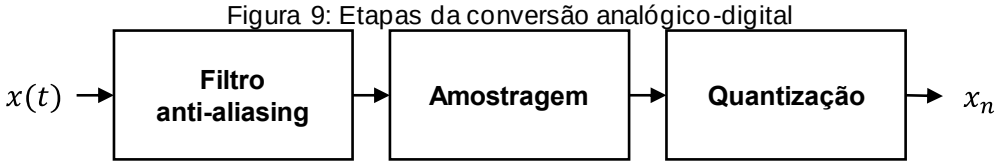
Computadores possuem um número finito de *bits* a partir dos quais podem representar valores. Sendo assim, não é possível representar todas as possíveis medições de cada amostra tomada, o que implica na necessidade mapear os infinitos valores medidos em um número finito de valores. Este processo é conhecido por quantização.

O primeiro passo é assumir que o sinal medido x_n, que tipicamente representa uma tensão, está compreendido em um intervalo, de $-\alpha$ a α. A quantidade de valores que um computador pode representar com β bits é 2^{β}. Dividindo o intervalo entre $-\alpha$ e α em 2^{β} partes iguais, obtém-se que o tamanho de cada degrau (Figura 8c) é:

$$\Delta = \frac{\alpha}{2^{\beta-1}} \tag{3}$$

Este processo é chamado de quantização uniforme, que é a mais comum para sinais de áudio (CHRISTENSEN, 2019). Ao final da quantização, o processo de digitalização do sinal de áudio está completo. A Figura 9 ilustra a ordem em que as operações acontecem.

Figura 9: Etapas da conversão analógico-digital



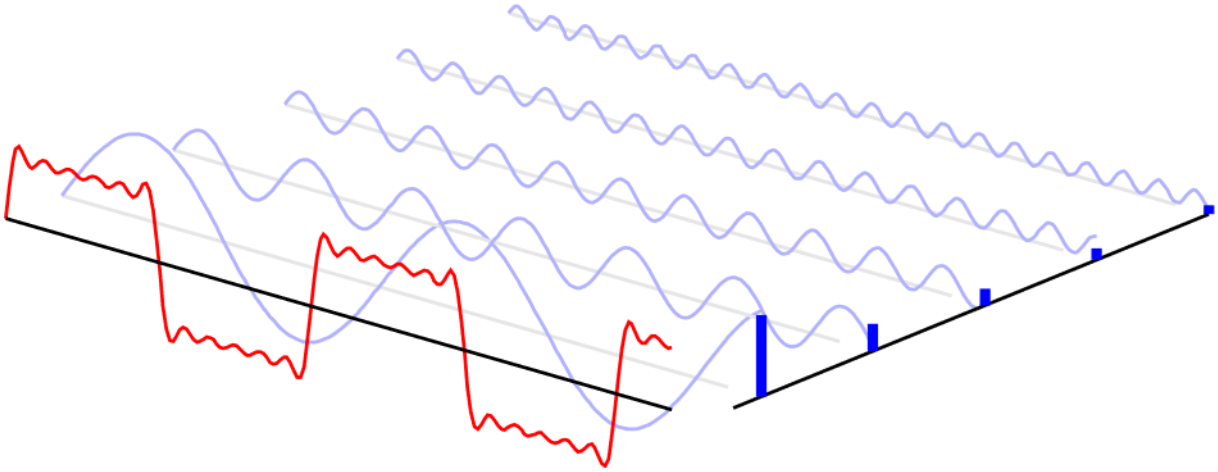
Fonte: adaptado de (CHRISTENSEN, 2019)

1.4.4 Domínio do tempo e domínio da frequência

O domínio do tempo (em vermelho na Figura 10) é aquele no qual um sinal varia em função do tempo, como aqueles explorados no item 1.3. Nele, é possível verificar as características da forma de onda, como sua amplitude e periodicidade. Além disso, pode-se dividir em tempo contínuo (onde o valor do sinal é conhecido para qualquer instante) ou tempo discreto (apenas valores amostrados são conhecidos).

Já o domínio da frequência (em azul na Figura 10) possibilita o conhecimento da frequência fundamental de um sinal, além de suas harmônicas e informações de fase, através da análise espectral.

Figura 10: Representação dos domínios do tempo e da frequência



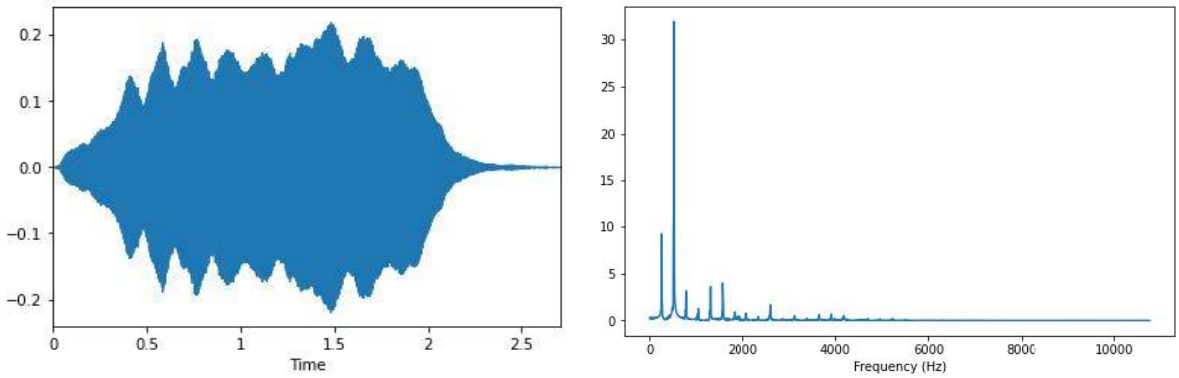
Fonte: (JAKE, 2021)

É possível obter a resposta em frequência de um sinal a partir do domínio do tempo e fazer o caminho inverso (reconstruir o sinal no domínio do tempo). Para isso, utiliza-se ferramentas matemáticas conhecidas como transformadas, das quais a transformada de Fourier é a mais utilizada para processamento de sinais de áudio (CHRISTENSEN, 2019).

1.4.4.1 Transformada de Fourier

A transformada de Fourier (FT – *Fourier Transform*) pode ser interpretada como uma decomposição do sinal em diferentes componentes de frequência (CHRISTENSEN, 2019). O resultado da operação da FT é chamado de espectro do sinal (Figura 11b) e mostra quais componentes sinusoidais estão presentes no mesmo. Através do espectro é possível determinar a frequência fundamental (de maior amplitude) e suas harmônicas.

Figura 11: (a) Forma de onda e (b) espectro de um sinal de áudio



(a) (b)

Fonte: o autor (2021)

1.4.4.2 Transformada Discreta de Fourier

Na prática, visto que os instrumentos de análise espectral são sistemas digitais, o sinal deve ser discretizado. Assim, para um sinal composto por N amostras espera-se que existam N amostras de frequência ω distribuídas uniformemente entre 0 e 2π (CHRISTENSEN, 2019). Assim, pode-se expressar a transformada discreta de Fourier (DFT – *Discrete Fourier Transform*) conforme a Equação (4) abaixo:

$$X(\omega) = \sum_{n=0}^{N-1} x_n e^{-j\omega_k n} \tag{4}$$

Onde $\omega_k = 2\pi \frac{k}{N}$, com k variando entre 0 e $N-1$. É possível reconstruir o sinal a partir do seu espectro utilizando a transformada inversa de Fourier, representada na Equação (5).

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X(\omega_k) e^{j\omega_k n} \tag{5}$$

1.4.4.3 Transformada Rápida de Fourier

O cálculo da DFT e de sua inversa é extremamente custoso e pouco eficiente, o que inviabiliza seu uso prático (DINIZ, DA SILVA e NETTO, 2014). Em face deste problema, foi criada a transformada rápida de Fourier (FFT – *Fast Fourier Transform*), que consiste em algoritmos capazes de implementar a DFT de forma eficiente. Estes algoritmos são os meios pelos quais os computadores calculam estas transformadas.

1.4.4.4 Espectrogramas

O espectrograma é uma representação bidimensional, uma imagem, do conteúdo dos sinais de áudio que visa transmitir informações sobre o espectro e como ele se comporta ao longo do tempo. Um de seus eixos representa o tempo e o outro a frequência. A quantidade de conteúdo do sinal, ou seja, a potência, é indicada por um código de cores (CHRISTENSEN, 2019). Exemplos de espectrogramas de sinais de áudio podem ser vistos na Figura 4b e na Figura 5b.

1.5 PERCEPÇÃO SONORA DOS SERES HUMANOS

A percepção sonora dos seres humanos não é linear, mas logarítmica. Isso se dá devido ao fato de que seu sistema auditivo é mais sensível a diferenças entre baixas frequências (DOSHI, 2021). Por exemplo, ao ser exposto a sons com as seguintes frequências:

- 100 Hz e 200 Hz
- 1.000 Hz e 1.100 Hz
- 10.000 Hz e 10.100 Hz

Por mais que a diferença entre a frequência de som em todos os pare seja 100 Hz, uma pessoa consegue distinguir facilmente os sons do primeiro par, enquanto tem dificuldades para distinguir os demais. A razão disso é que, no primeiro caso, um dos sinais tem o dobro da frequência do outro, enquanto no último caso a maior frequência é apenas 1% superior a menor.

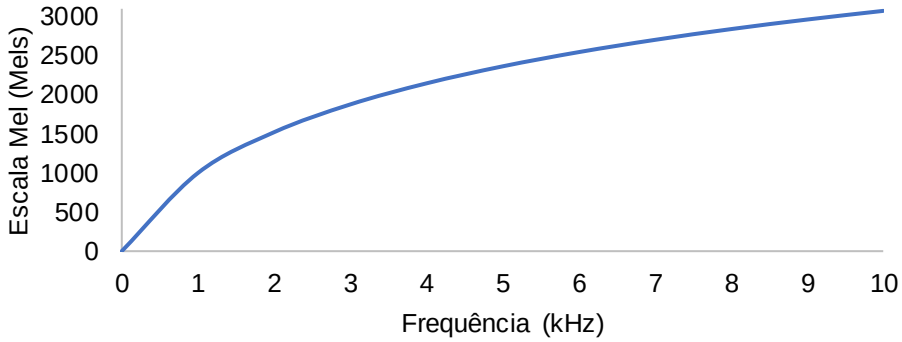
1.5.1 Escala mel

A escala mel consiste em uma escala logarítmica perceptual cujo intuito é manter os tons de frequência equidistantes a partir do limite de percepção humana em 1.000 Hz (STEVENS, S. S.; VOLKMANN, J.; NEWMAN, E. B, 1936). Para calcular quanto uma frequência f equivale em mels, utiliza-se a Equação (6) e é possível voltar ao valor em Hertz aplicando a exponenciação na mesma.

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{100}\right) \tag{6}$$

Pelo gráfico da Figura 12, é possível notar que até 1.000 Hz, a curva tem característica linear. Contudo, a partir deste ponto, a tendencia logarítmica se mostra mais evidente.

Figura 12: Gráfico da escala mel, gerado a partir da Equação (6)

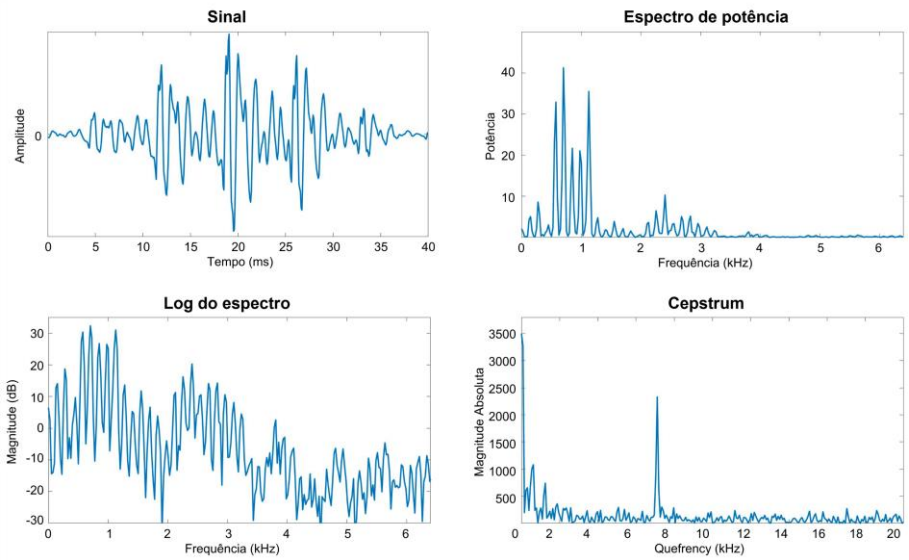


Fonte: o autor (2021)

1.6 O CEPSTRUM

O cepstrum é o resultado da aplicação da transformada inversa de Fourier no logaritmo do espectro de um sinal (Figura 13). Existem o cepstrum complexo, de potência, de fase e o real. Dentre eles, o de potência é o mais relevante para o processamento de sinais sonoros (SHARMA, G.; UMAPATHY, K.; KRISHNAN, S, 2019).

Figura 13: Etapas para obtenção do cepstrum de um sinal



Fonte: o autor (2021)

Visto que o cepstrum é obtido a partir da transformada inversa de Fourier um espectro, espera-se que seu eixo da abscissa seja expresso em unidades de tempo. Por conseguinte, a 'frequência' do cepstrum seria expressa nestas mesmas unidades.

Quadro 4: Nomenclatura dos parâmetros cepstrais

Português	Espectro	Frequência	Fase	Amplitude	Filtragem	Harmônico	Período
Inglês	*Spectrum*	*Frequency*	*Phase*	*Amplitude*	*Filtering*	*Harmonic*	*Period*
Cepstral	*Cepstrum*	*Quefrency*	*Saphe*	*Gamnitude*	*Liftering*	*Rahmonic*	*Repiod*

Fonte: adaptado de (CHILDERS, SKINNER e KEMERAIT, 1977).

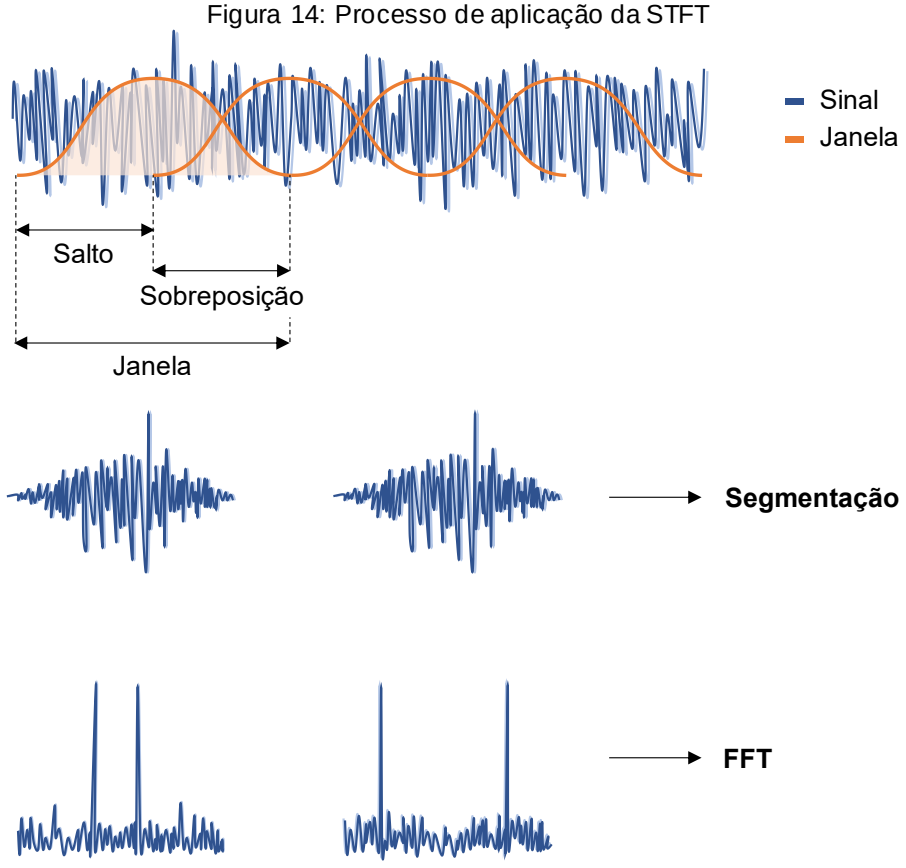
Para evitar confusão, os parâmetros cepstrais foram nomeados a partir dos parâmetros espectrais (Quadro 4), trocando a primeira sílaba pela segunda (CHILDERS, SKINNER e KEMERAIT, 1977).

1.6.1 Coeficientes Mel-Cepstrais (MFCCs)

A técnica de extração de atributos MFCC consiste no janelamento do sinal, na aplicação da transformada discreta de Fourier (DFT – *Discrete Fourier Transform*), na obtenção da magnitude logarítmica e a aplicação da escala de frequência Mel, finalizando com a aplicação da transforma discreta de cosseno (DCT – *Discrete Cosine Transform*) (RAO e VUPPALA, 2014). Uma discussão mais detalhada destas etapas com base no neste mesmo autor é realizada a seguir:

Para a obtenção de características acústicas estáveis é preciso analisar o áudio em períodos intervalos de tempo suficientemente pequenos. Para isso, o áudio é divido em quadros (*frames*), nos quais é aplicada uma janela (tipicamente de Hanning ou Hamming). O janelamento (Figura 14) é realizado para suavizar as bordas de cada *frame* e evitar artefatos no espectro do sinal, que é obtido através da DFT.

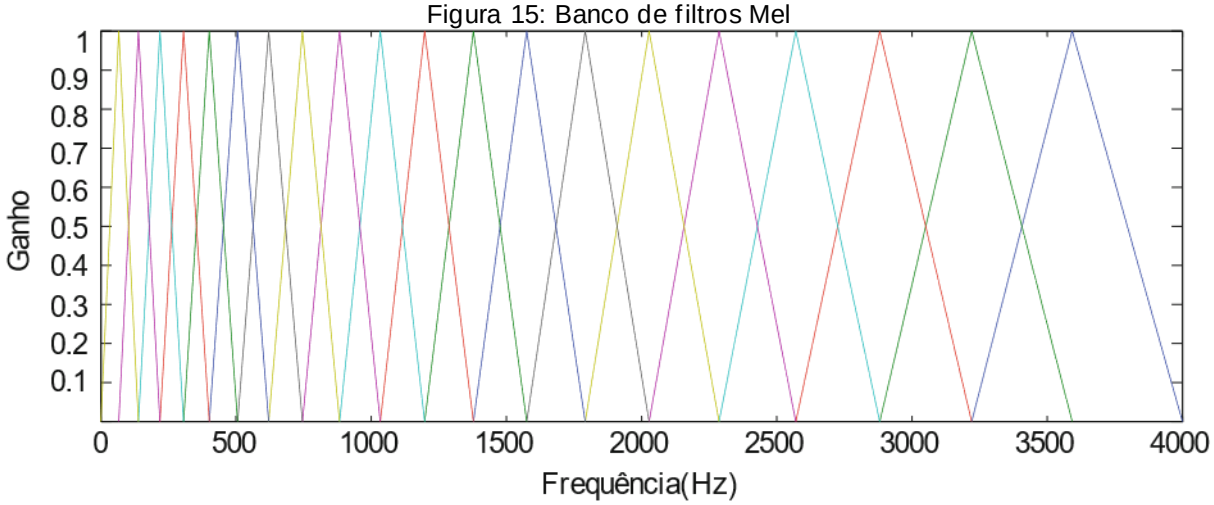
Figura 14: Processo de aplicação da STFT



Fonte:adaptado de (JEON, *et al.*, 2020)

Em seguida, o espectro passa por um banco de filtros Mel (Figura 15), que é composto por um conjunto de filtros passa-faixa a partir do qual se obtém o espectro em escala Mel. Para o cálculo de MFCCs, os bancos de filtros são geralmente implementados no domínio da frequência. As frequências centrais dos filtros são normalmente espaçadas uniformemente no eixo horizontal. No entanto, para imitar a percepção dos ouvidos humanos, o eixo é deformado de acordo com a função não linear dada na Equação (6).

Figura 15: Banco de filtros Mel



Fonte: adaptado de (RAO e VUPPALA, 2014)

A próxima etapa consiste em aplicar a transformada discreta de cosseno (DCT – *Discrete Cosine Transform*). Antes de calcular a DCT, o espectro mel é geralmente representado em uma escala logarítmica. Isso resulta em um sinal no domínio cepstral com um pico de *quefrency* que corresponde ao tom do sinal e um número de formantes representando picos de *quefrency* baixos. Uma vez que a maioria das informações do sinal é representada pelos primeiros coeficientes MFCC, o sistema pode se tornar robusto, extraindo apenas os coeficientes, ignorando ou truncando componentes DCT de ordem superior. Por fim, os MFCCs são calculados conforme a Equação (7):

$$c(n) = \sum_{m=0}^{M-1} \log_{10}\big(s(m)\big) \cdot \cos\left(\frac{\pi n(m-0.5)}{M}\right); \quad n = 0, 1, 2, \dots, C-1 \tag{7}$$

Onde $c(n)$ são os coeficientes e C é o número de MFCCs, que tipicamente varia entre 8 e 13 (RAO e VUPPALA, 2014).

1.7 APRENDIZADO DE MÁQUINA

Aprendizado de máquina (ML – *Machine Learning*) refere-se à criação e ao uso de modelos que são aprendidos a partir de dados. Tipicamente, utiliza-se dados existentes para desenvolver modelos que possam prever saídas para novos dados (GRUS, 2016). As categorias de sistemas de ML são o aprendizado supervisionado, o não supervisionado, o semi-supervisionado e por reforço (GÉRON, 2019), cujas características são discutidas a seguir.

No aprendizado supervisionado, os dados utilizados no treinamento do modelo são rotulados, ou seja, sua solução é conhecida. Por exemplo: um arquivo de áudio contendo um ciclo respiratório onde há presença de sibilos acompanhado de um arquivo de anotação que indique a presença destes eventos. Este tipo de aprendizado é utilizado em tarefas de classificação e regressão.

O aprendizado não supervisionado, por sua vez, utiliza dados de treinamento não rotulados. Estes são utilizados em algoritmos de visualização, redução de dimensionalidade e detecção de anomalias.

Intuitivamente, aprendizado semi-supervisionado é aquele no qual os dados de treinamento são parcialmente rotulados. A exemplo, os algoritmos de reconhecimento os serviços de hospedagem de fotos, que são capazes de identificar que uma mesma pessoa aparece em imagens distintas (agrupamento).

O aprendizado por reforço é aquele no qual o sistema de aprendizado (agente) observa o ambiente e executa ações para obter recompensas ou penalidades. De acordo com o que recebe, ele aprende a melhor estratégia (política) para executar as ações. Um exemplo desta categoria são os algoritmos utilizados por robôs para aprender a andar.

A classificação automatizada de sons pulmonares tem potencial para detectar anormalidades no sistema respiratório ainda em estágios iniciais e pode nortear os profissionais de saúde a indicar o tratamento mais adequado a seus pacientes (ROCHA, *et al.*, 2017).

Os métodos computacionais de análise de sons pulmonares já são estudados há bastante tempo. Algoritmos tradicionais de aprendizado de máquina (ML – *Machine Learning*), como árvores de decisão, k-vizinhos mais próximos (KNN – *k-Nearest Neighbors*), classificador bayesiano e máquinas de vetores de suporte (SVM –

Support Vector Machine) foram bastante explorados para classificação e mostraram resultados otimistas (NAVES, 2015) e (PALANIAPPAN, R.; SUNDARAJ, K.; AHAMED, N. U, 2013). Estudos mais recentes exploram as redes neurais artificiais para realizar a tarefa de classificação dos sons adventícios. Classificadores como redes neurais convolucionais (CNN – *Convolutional Neural Network*) e redes neurais recorrentes (RNN – *Recurrent Neural Network*) têm sido precisos quando alimentados por espectrogramas e componentes mel-cepstrais (MFCCs – *Mel-Frequency Cepstral Coefficients*) (NGUYEN e PERNKOPF, 2020) e (SENGUPTA, SAHIDULLAH e SAHA, 2016).

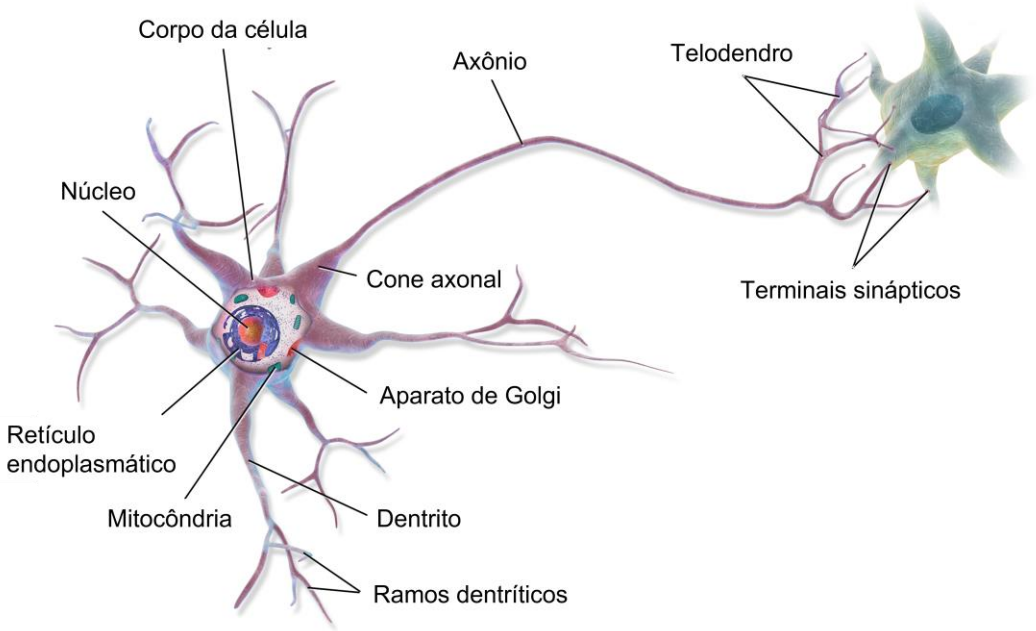
1.7.1 Redes Neurais Artificiais

Uma rede neural artificial (ANN – *Artificial Neural Network*) é um modelo preditivo motivado pela forma como o cérebro funciona (GRUS, 2016).

1.7.1.1 Neurônios biológicos e artificiais

Antes de explorá-las mais a fundo, é interessante entender a estrutura e o comportamento dos neurônios biológicos (Figura 16), que são as células encontradas no córtex cerebral animal que recebem impulsos elétricos de outros neurônios através das sinapses. Quando um neurônio recebe um número suficiente de sinais de outros em tempo curto (alguns milissegundos), ele dispara seus próprios sinais (GÉRON, 2019).

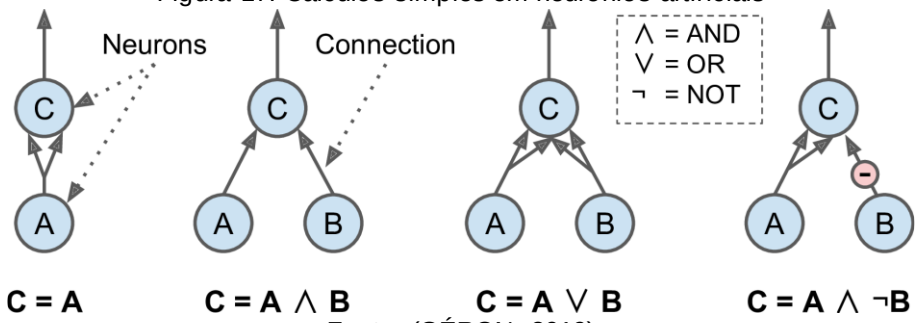
Figura 16: Neurônio biológico



Fonte: adaptado de (GÉRON, 2019).

Individualmente, os neurônios biológicos parecem se comportar de forma simples, mas estão existem bilhões que formam uma grande rede na qual, normalmente, um neurônio está conectado a milhares de outros. O neurônio artificial (Figura 17), por sua vez, é constituído de uma ou mais entradas binárias e uma saída binária. Ele ativa sua saída de acordo com o estado das entradas.

Figura 17: Cálculos simples em neurônios artificiais

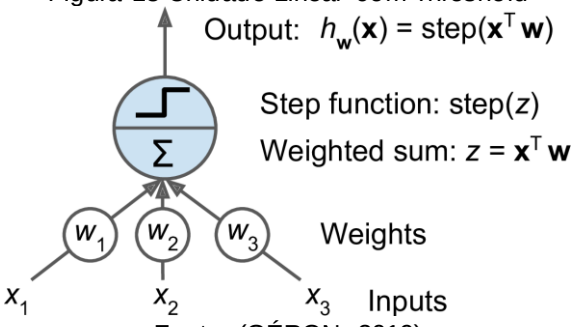


Fonte: (GÉRON, 2019)

1.7.1.2 O Perceptron

O *perceptron* é uma das arquiteturas ANN mais simples (GÉRON, 2019). Ele se baseia na unidade linear com *threshold* (LTU – *Linear Threshold Unit*), um neurônio artificial cujas entradas e saídas são números (e não valores binários, como visto anteriormente) e cada conexão está associada a um peso.

Figura 18 Unidade Linear com Threshold



Fonte: (GÉRON, 2019)

A LTU (Figura 18) calcula a soma ponderada das entradas e então aplica uma função degrau – função de Heaviside, Equaçao (8) – a esta soma.

$$heaviside(z) = \begin{cases} 0 \; se \; z < 0 \\ 1 \; se \; z \geq 0 \end{cases} \tag{8}$$

Um *perceptron* é composto por apenas uma camada de LTUs. Contudo, existem as os *perceptrons* multicamadas (MLP – *Multilayer Perceptron*), que são formados por uma camada de entrada, uma ou mais camadas LTUs ocultas e uma

camada final de LTU na saída (GÉRON, 2019). No caso de duas ou mais camadas ocultas, a ANN é chamada de rede neural profunda (DNN – *Deep Neural Network*).

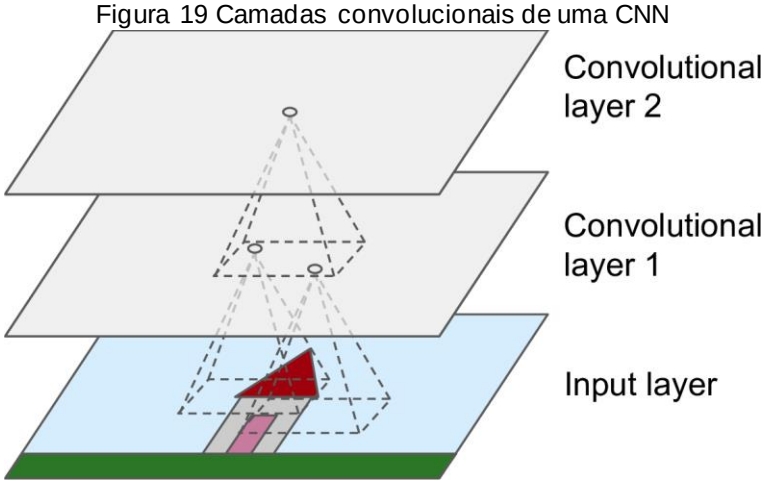
1.7.2 Redes Neurais Convolucionais

As redes neurais convolucionais (CNN – *Convolutional Neural Network*) surgiram do estudo do córtex visual do cérebro e são usadas principalmente no reconhecimento de imagens. Mas também são encontradas em tarefas como reconhecimento de voz e processamento de linguagem natural (GÉRON, 2019).

1.7.2.1 Camada convolucional

A camada convolucional é o bloco de construção mais importante de um CNN. Conforme a Figura 19, verifica-se que os neurônios da primeira camada não estão conectados a todos os pixels da imagem, apenas àqueles em seus campos de visão. Da mesma forma, cada neurônio da segunda camada está conectado somente a pixels compreendidos em uma certa área da camada anterior, e assim sucessivamente. Isso permite que as primeiras camadas reúnam características de baixo nível, enquanto as camadas ocultas lidam com as de nível mais alto.

Figura 19 Camadas convolucionais de uma CNN



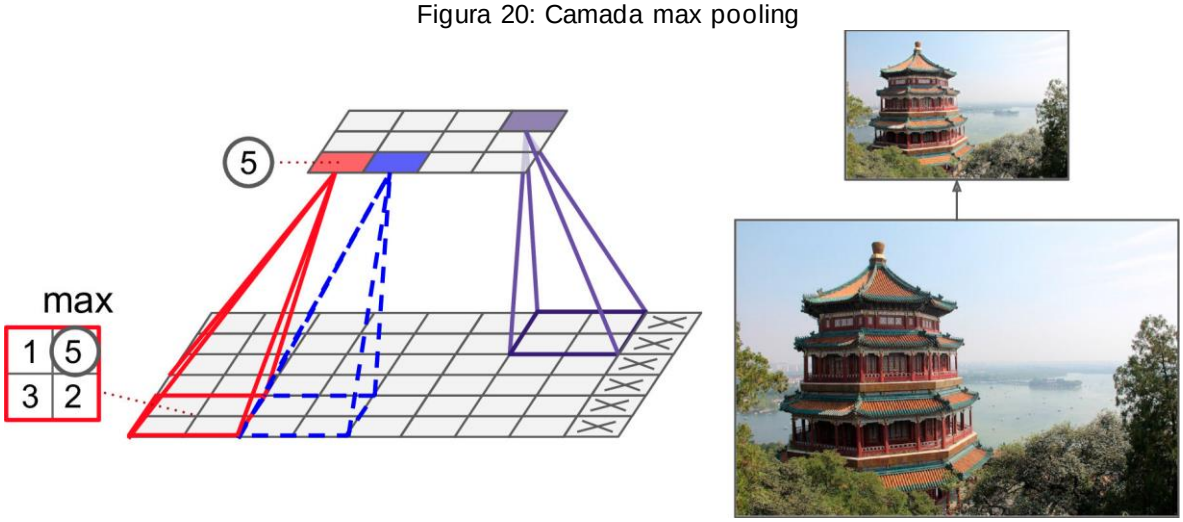
Fonte: (GÉRON, 2019)

Uma camada convolucional aplica simultaneamente vários filtros a suas entradas com a finalizada de detectar características. Isso é útil devido à natureza das imagens – entradas deste tipo de rede neural – que possuem uma camada para cada canal de cor. Tipicamente, uma imagem RGB possui três canais de cor (vermelho, verde e azul e imagens em escala de cinza possuem apenas um canal.

1.7.2.2 Camada de *Pooling*

Esta é uma camada mais simples, cujo objetivo é subamostrar (reduzir) o tamanho da imagem (Figura 20) a fim de reduzir a carga computacional. Assim como nas camadas convolucionais, os neurônios da camada de *pooling* são conectados somente à algumas saídas da camada anterior.

Figura 20: Camada max pooling

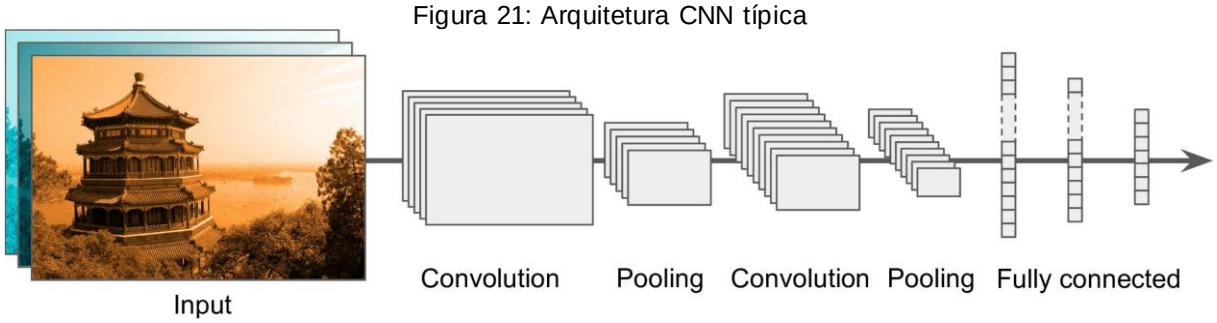


Fonte: (GÉRON, 2019)

1.7.2.3 Arquiteturas CNN

As arquiteturas CNN são compostas por uma pilha de camadas convolucionais, seguidas de uma camada de *pooling* (quantas vezes forem necessárias). As dimensões da imagem são reduzidas ao passo que a rede avança e no final é adicionada uma rede neural composta de algumas camadas totalmente conectadas, seguida de uma camada de saída, que gera a previsão.

Figura 21: Arquitetura CNN típica



Fonte: (GÉRON, 2019)

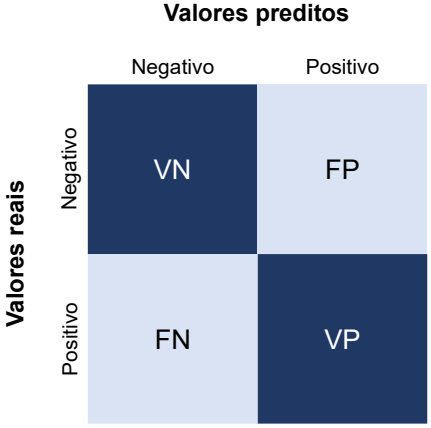
1.7.3 Métricas de avaliação

É importante determinar o desempenho de um modelo treinado para ter uma estimativa de sua taxa de erros, evitando surpresas na etapa de generalização – estimativa sobre novos dados, que não foram utilizados para treino (GÉRON, 2019).

1.7.3.1 Matriz de confusão

Uma matriz de confusão (Figura 22) é uma matriz quadrada utilizada para comparar as previsões de uma classificação com os valores verdadeiros. A diagonal principal contém os acertos do modelo e as os demais valores erros.

Figura 22: Matriz de confusão binária



Fonte: adaptado de (SCUDILIO, 2020)

Os dados que podemos extrair de uma matriz de confusão são: verdadeiros negativos, verdadeiros positivos, falsos negativos e falsos positivos.

- Verdadeiro negativo (VN): previsão negativa que realmente é negativa
- Verdadeiro positivo (VP): previsão correta de um valor que é positivo
- Falso negativo (FN): identificação de um positivo como negativo
- Falso positivo (FP): classificação positiva de algo que é negativo

1.7.3.2 Acurácia

A acurácia é a métrica mais simples de todas, sendo mais indicada quando conjunto de dados é bem balanceado. É utilizada para representar as previsões corretas de um modelo, e é calculada conforme a Equação (9).

$$\mathrm{Acc} = \frac{\mathrm{VP} + \mathrm{VN}}{\mathrm{VP} + \mathrm{VN} + \mathrm{FP} + \mathrm{FN}} \tag{9}$$

1.7.3.3 Valor preditivo negativo e positivo

Valor preditivo negativo (VPN) é a métrica que traz a informação da quantidade de observações classificadas como negativas que realmente são negativas, como mostra a Equação (10). Ou seja: os negativos que foram classificados corretamente.

$$\mathrm{VPN} = \frac{\mathrm{VN}}{\mathrm{VN} + \mathrm{FN}} \tag{10}$$

Por sua vez, o valor preditivo positivo (VPP), também comumente chamado de precisão, Equação (11), refere-se aos positivos que foram classificados corretamente:

$$\mathrm{Precision} = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FP}} \tag{11}$$

1.7.3.4 Sensibilidade e especificidade

A sensibilidade, ou *recall*, é a proporção dos verdadeiros positivos entre todos os que são de fato verdadeiros. É calculada conforme a Equação (12).

$$\mathrm{Recall} = \frac{\mathrm{VP}}{\mathrm{VP} + \mathrm{FN}} \tag{12}$$

De forma complementar, a especificidade (Equação ()) é a proporção dos verdadeiros negativos entre todos os que são negativos.

$$\mathrm{Especificidade} = \frac{\mathrm{VN}}{\mathrm{VN} + \mathrm{FP}} \tag{13}$$

1.7.3.5 F1-Score

F1-Score é a média harmônica entre o *recall* e a precisão (*precision*). Utilizada quando as classes estão desbalanceadas devido a capacidade de expressar os resultados de forma mais realista.

$$\mathrm{F1} = 2 * \frac{\mathrm{precision} * \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{14}$$

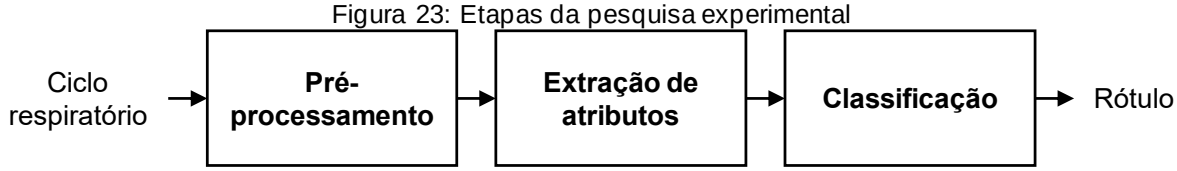
2 METODOLOGIA

Este trabalho consiste em uma pesquisa científica aplicada, na qual foi realizada pesquisa exploratória, a fim de levantar material relevante para o embasamento teórico a respeito de processamento de sinais de áudio e sobre as técnicas adequadas para classificação de sons respiratórios pulmonares. Além disso, houve uma pesquisa explicativa experimental na qual foram feitos estudos de laboratório a fim de determinar quais métodos apresentam a melhor avaliação dos dados.

A primeira etapa deste projeto consistiu em fazer o levantamento bibliográfico e das ferramentas necessárias para o processamento de sinais de áudio. Determinou-se que a linguagem Python possui as ferramentas necessárias para realizar tal processamento, além da classificação dos sons respiratórios.

Em seguida, verificou-se a necessidade de obter dados para a realização da etapa experimental. Em um cenário ideal, a coleta e a rotulação dos sons pulmonares seriam feitas pelo autor. Contudo, devido à complexidade desta tarefa e a impossibilidade de realizá-la em tempo hábil para a realização do projeto, optou-se por utilizar um *dataset* (conjunto de dados) já existente, conforme descrito mais à frente no item 2.3.1. A Figura 23 ilustra as etapas do ciclo experimental da pesquisa.

Figura 23: Etapas da pesquisa experimental



Fonte: o autor (2021)

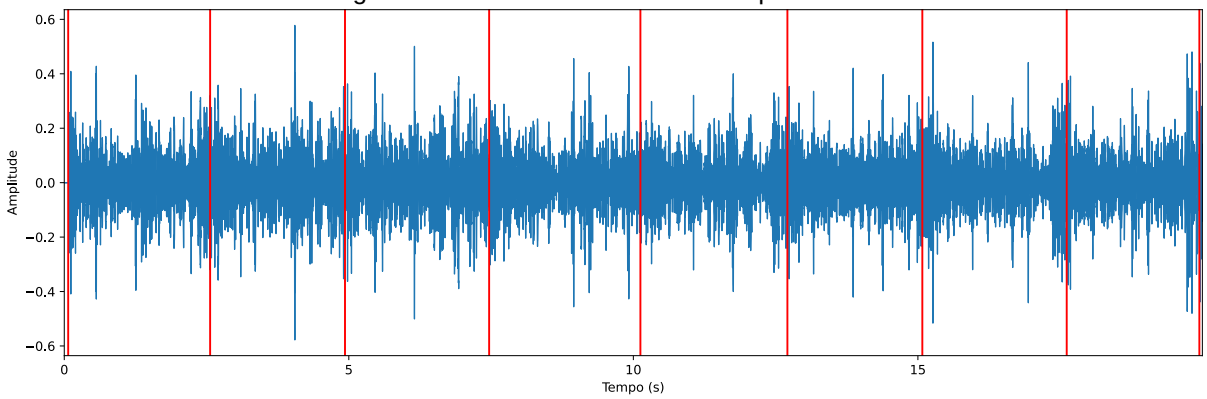
2.1 PRÉ-PROCESSAMENTO DE DADOS

Os dados disponibilizados contêm gravações de diversos pacientes acometidos por doenças respiratórias pulmonares. Contudo, os arquivos de anotação se referem a trechos (ciclos respiratórios) contidos em cada arquivo. Sendo assim, existe a necessidade de separar os arquivos originais em novos arquivos de áudio para cada ciclo respiratório, de modo que se classifique cada um dos ciclos quanto a presença de sons adventícios.

O arquivo de anotação contém as seguintes informações: tempos de início e fim dos ciclos respiratórios e se há presença de estertor, sibilo, ambos ou nenhum

(sons normais). A Figura 24 foi gerada a partir de uma gravação e das informações de seu respectivo arquivo de anotação e mostra a forma de onda (azul) e os ciclos respiratórios (delimitados pelas linhas vermelhas).

Figura 24: Forma de onda ciclos respiratórios



Fonte: o autor (2021)

Devido ao processo de aquisição dos sons respiratórios se dar por mais de um equipamento (ROCHA, *et al.*, 2017), os sinais possuem frequências de amostragem distintas. Logo, se faz necessário uma etapa de reamostragem para padronizar este parâmetro. As amplitudes das formas de onda também podem estar em escalas diferentes, o que implica na necessidade de normalização. Optou-se em realizar estas etapas ainda nas gravações originais, por serem uma quantidade de arquivos menores do que o total de arquivos de ciclos respiratórios, de modo que os ciclos respiratórios exportados já mantenham tais propriedades.

Os ciclos respiratórios possuem durações distintas, mas as redes neurais convolucionais utilizadas na etapa de classificação requerem dados com as mesmas dimensões (NGUYEN e PERNKOPF, 2020). Sendo assim, é preciso determinar uma duração padrão para cada arquivo de áudio que será gerado e preencher com zeros (silêncio) aqueles cuja duração for menor que a determinada ou truncar aqueles que tiverem duração superior. Este processo de preenchimento é conhecido como *zero padding* e é utilizado no processamento digital de imagens, durante a aplicação de filtros.

2.2 EXTRAÇÃO DE ATRIBUTOS E CLASSIFICAÇÃO

Após a etapa de pré-processamento do sinal, planeja-se extrair atributos dos ciclos respiratórios para poder alimentar a rede neural. Inicialmente, os atributos que se deseja extrair são os MFCCs e imagens de espectrogramas, que já se mostraram eficientes em classificar sinais sonoros em estudos anteriores (AYKANAT, *et al.*, 2017)

e (CHAUDHARI, *et al.*, 2020). Uma alternativa para trabalhar com sinais sonoros de duração distinta é RNNs (GÉRON, 2019), que poderá ser explorada caso haja tempo hábil durante o desenvolvimento da pesquisa.

2.3 RECURSOS

Este trabalho foi desenvolvido em um computador *desktop* rodando o sistema operacional Windows 10, cujas características estão detalhadas no Quadro 5. Todo o *software* será escrito em linguagem Python utilizando ferramentas e módulos gratuitos.

Quadro 5: Recursos utilizados no projeto

Recurso	Descrição
Computador *desktop*	▪ Sistema operacional: Windows 10 de 64 bits; ▪ Processador: Intel core i7-9700 de 8 núcleos @3GHz; ▪ Memória RAM: 16GB DDR4 @2667MHz; ▪ GPU Intel UHD Graphics 630 (integrada) ▪ Armazenamento: 512GB NVMe + 960GB SATA SSD
Python	Além do Python, são necessários alguns módulos e ferramentas adicionais, descritos posteriormente.
Base de dados	*Dataset* elaborado em 2017, contendo gravações de sons pulmonares classificados como: ▪ Normais; ▪ Presença de estertor; ▪ Presença de sibilo; ▪ Presença de estertor e sibilo Além disso, estão rotuladas as 7 comorbidades que acometem os pacientes, estando listadas mais a frente.

Fonte: o autor (2021)

O custo dos recursos foi zero, visto que o *hardware* necessário já estará disponível antes da realização do projeto e tantos os dados quanto o software utilizado são distribuídos gratuitamente.

2.3.1 BASE DE DADOS DE SONS RESPIRATÓRIOS

Os dados processados durante este projeto estão disponíveis para *download* no seguinte endereço: https://bhichallenge.med.auth.gr. O arquivo é composto por:

- 920 arquivos de áudio no formato .wav;
- 920 arquivos de anotação no formato .txt;
- 1 arquivo de texto listando o diagnóstico de cada paciente;
- 1 arquivo de texto explicando a nomenclatura dos arquivos;
- 1 arquivo de texto contendo informações demográficas dos pacientes

O nome de cada arquivo de áudio contém informações relevantes, separadas por um sublinado (_). As informações são: número do paciente, índice da gravação,

local da gravação (no tórax), modo de aquisição e equipamento utilizado. Por exemplo: o arquivo ‘101_1b1_Al_sc_Meditron.wav’ refere-se ao paciente de número 101, tem índice de gravação 1b1, foi gravado na região anterior esquerda do tórax, a gravação é mono (canal único) e o equipamento utilizado foi o estetoscópio eletrônico WelchAllyn Meditron Master Elite. A seguir, lista-se as regiões torácicas das quais os sons foram obtidos:

- Traqueia (Tc);
- Anterior esquerda (Al);
- Anterior direita (Ar);
- Posterior esquerda (Pl);
- Posterior direita (Pr);
- Lateral esquerda (Ll);
- Lateral direita (Lr)

Estes pontos estão representados na Figura 25, junto de um exemplo do processo de anotação, que se deu pelo *software* Audacity.

Figura 25: Coleta de dados: (a) regiões do tórax e (b) processo de anotação

(a) (b)

Fonte: (ROCHA, *et al.*, 2017)

Além de gravações limpas dos sons respiratórios, os autores da base de dados informam que há arquivos contendo ruídos. Deste modo, os dados analisados contemplam diversas situações do mundo real. Além disso, conforme o exemplo anterior, há arquivos gravados em apenas um canal (mono) e outros em múltiplos canais (gravados simultaneamente por mais de um estetoscópio) (ROCHA, *et al.*, 2017). Os equipamentos utilizados na captação foram:

- Microfone AKG C417L (AKGC417L);
- Estetoscópio 3M Littmann Classic II SE(LittC2SE);
- Estetoscópio eletrônico 3M Litmmann 3200 (Litt3200);
- Estetoscópio eletrônico WelchAllyn Meditron Master Elite (Meditron)

Esta base de dados inclui arquivos de áudio com gravações que variam de entre 10 e 90 segundos de 126 pacientes de diferentes faixas etárias (crianças,

adultos e idosos), resultando em mais de 5 horas de gravações contendo o total de 6898 ciclos respiratórios. Destes, 1864 ciclos contêm estertor, 886 contém sibilos, 506 contém ambos e o restante são ciclos com sons normais (ROCHA, *et al.*, 2017).

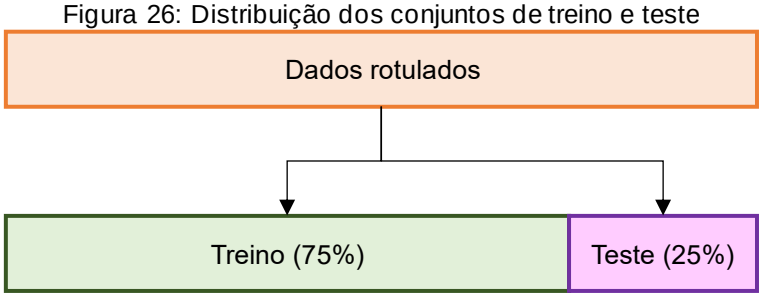
2.4 CENÁRIOS EXPLORADOS

O modelo proposto foi testado para classificar os ciclos respiratórios em dois cenários: prevendo os eventos (sibilo, estertor, ambos ou nenhum), tendo 4 classes possíveis; e prevendo as comorbidades, que são 7 mais nenhuma (saudável), resultando em 8 classes:

- *COPD*: doença pulmonar obstrutiva crônica (DPOC)
- *URTI*: infecção do trato respiratório superior
- *Bronchiectasis*: bronquiectasia
- *Pneumonia*: pneumonia
- *Bronchiolitis*: bronquiolite
- *LRTI*: infecção do trato respiratório inferior
- *Asthma*: asma
- *Healthy*: saudável

Os dois cenários utilizam a mesma rede neural, diferenciando-se na quantidade de dados por rótulo, uma vez que os arquivos são redistribuídos de acordo com o que se deseja classificar. Os dados originais serão distribuídos aleatoriamente conforme mostra a Figura 26 (75% para treino e 25% para testes).

Figura 26: Distribuição dos conjuntos de treino e teste



Fonte: o autor (2021).

3 IMPLEMENTAÇÃO

Nesta seção serão descritos todos os procedimentos da parte experimental, na qual uma rede neural convolucional foi elaborada para classificar os sinais sonoros. Para cada etapa descrita será mostrado o fragmento do código em Python utilizado.

3.1 MÓDULOS UTILIZADOS

Os experimentos a seguir foram desenvolvidos na versão 3.9.5 do Python. A plataforma escolhida para execução foi Jupyter Notebook (versão 6.4.0).

```
import os # arquivos
import librosa, librosa.display # processamento de audio
import numpy as np # matematica
import pandas as pd # dataframes
import tensorflow as tf # machile learning
import matplotlib.pyplot as plt # graficos
import IPython.display as ipd # player de audio
import soundfile as sf # gerar arquivos de audio
from tqdm import tqdm # ansiedade
from tensorflow import keras # redes neurais
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score # metricas de desempenho
from mlxtend.plotting import plot_confusion_matrix # exibir matriz de confusão
```

O fragmento de código acima mostra todas as ferramentas utilizadas com um comentário indicando sua função (comentários vem após a cerquilha – #). O Quadro 6, a seguir, indica as versões dos principais módulos, bem como uma breve descrição.

Quadro 6: Módulos utilizados e suas versões.

Módulo	Versão	Descrição
Librosa	0.8.1	Módulo para análise e processamento de sinais de áudio. Utilizado para ler arquivos de som e extrair caraterísticas.
Numpy	1.19.5	Biblioteca que suporta o processamento *arrays* multidimensionais e matrizes, juntamente com uma grande coleção de funções matemáticas de alto nível para operar sobre estas matrizes.
Pandas	1.2.4	Biblioteca para manipulação e análise de dados. Oferece estruturas e operações para manipular tabelas numéricas e séries temporais. Utilidada para organizar os dados fornecidos em *dataframes*.
TensorFlow	2.5.0	Biblioteca de código aberto para aprendizado de máquina aplicável a uma ampla variedade de tarefas. É um sistema para criação e treinamento de redes neurais.
Matplotlib	3.4.2	Biblioteca para criação de gráficos e visualizações de dados em geral.
TQDM	4.62.3	Módulo que habilita uma barra de progresso em *loops*. Utilizado para acompanhar o desenvolvimento dos modelos.
Soundfile	0.10.3	Biblioteca ler e escrever arquivos de som. Utilizada para gerar novos arquivos com os ciclos respiratórios a partir das gravações fornecidas no *dataset*.
Keras	2.5.0	Keras é uma API de aprendizado profundo escrita em Python, executada na plataforma de aprendizado de máquina TensorFlow. Utilzada aqui para criar o modelo da rede neural.
Scikit-learn	0.24.2	Biblioteca que fornece muitos algoritmos de aprendizagem não supervisionados e supervisionados. Utilizada para medir o desempenho do modelo.
Mlxtend	0.19.0	Biblioteca que contém ferramentas que auxiliam em tarefas relacionadas a ciência de dados. Utilizada para exibir a matriz de confusão de forma mais detalhada.

Fonte: o autor (2021)

3.2 PREPARAÇÃO DOS DADOS

Dentre os arquivos fornecidos, um deles mostra o diagnóstico dos 126 pacientes. O arquivo é do formato CSV, e pode ser lido utilizando Pandas da seguinte forma:

```
# dataset dos pacientes

pacientes_header = ['id_paciente', 'comorbidade']
pacientes_df = pd.read_csv('patient_diagnosis.csv',names=pacientes_header)
pacientes_df
```

O resultado é guardado em um *dataframe*, representado na Tabela 1, a seguir. Boa parte dos dados está oculta para fins práticos de visualização:

Tabela 1: *Dataframe* dos pacientes

	id_paciente	comorbidade
0	101	URTI
1	102	Healthy
2	103	Asthma
...	...	...
123	224	Healthy
124	225	Healthy
125	226	Pneumonia

Fonte: o autor (2021)

Cada uma das 920 gravações acompanha um arquivo de texto com as anotações dos tempos em que os eventos ocorrem. Foi feito um procedimento similar ao anterior para inserir estas informações no *dataframe*:

```
# 112_1p1_Ll_sc_Litt3200.txt
path = 'audio_and_txt_files/'
txt = path + '112_1p1_Ll_sc_Litt3200.txt'
txt_header = ['inicio_ciclo', 'fim_ciclo', 'estertor', 'sibilo']
txt_df = pd.read_csv(txt, sep='\t', names = txt_header)
```

O fragmento de código acima mostra como um arquivo de texto, relativo à gravação 112_1p1_Ll_sc_Litt3200, é lido e transformado no *dataframe* da Tabela 2.

Tabela 2: *Dataframe* de ciclos e eventos

	inicio_ciclo	fim_ciclo	estertor	sibilo
0	0.0000	4.3188	0	0
1	4.3188	7.6336	0	0
2	7.6336	11.0150	0	0
...	...	...	...	...
7	22.9670	25.6470	0	1
8	25.6470	28.2140	0	1
9	28.2140	29.3600	0	0

Fonte: o autor (2021)

Através de uma estrutura de repetição, foi possível coletar as informações de todos os 920 arquivos de texto. Estes dados foram gravados em um *dataframe* que continha os tempos inicial e final, além dos eventos que aconteciam em todos os 6898

ciclos respiratórios. Contudo, é possível notar na Tabela 2 que a presença de estertor e sibilo é indicada em colunas separadas. Assim, utilizou-se o seguinte fragmento de código para criar a coluna ‘eventos’, que contém:

- ‘Nenhum’ quando as colunas ‘estertor’ e ‘sibilo’ forem iguais a 0
- ‘Estertor’, quando ‘estertor’ for igual a 1 e ‘sibilo’ for igual a 0
- ‘Sibilo, quando ‘estertor’ for igual a 0 e ‘sibilo’ for igual a 1
- ‘Estertor e sibilo’, quando ‘estertor’ for igual a 1 e ‘sibilo’ for igual a 0

```
estertor = arquivos_df.estertor.tolist()
sibilo = arquivos_df.sibilo.tolist()

eventos = []

for i in range(len(estertor)):
    if sibilo[i] == 0 and estertor[i] == 0:
        eventos.append('Nenhum')
    elif sibilo[i] == 0 and estertor[i] == 1:
        eventos.append('Estertor')
    elif sibilo[i] == 1 and estertor[i] == 0:
        eventos.append('Sibilo')
    else:
        eventos.append('Estertor e sibilo')

arquivos_df['eventos'] = eventos
arquivos_reindex = ['inicio_ciclo', 'fim_ciclo', 'eventos', 'id_paciente', 'nome_arquivo']
arquivos_df = arquivos_df.reindex(columns=arquivos_reindex)
```

Feito isso, os dois *dataframes* foram unidos para formar um terceiro, que combina consolida todas as informações.

```
# unir dataframes: pacientes + arquivos

pacientes_df.id_paciente = pacientes_df.id_paciente.astype('int32')
arquivos_df.id_paciente = arquivos_df.id_paciente.astype('int32')

dados_df = pd.merge(pacientes_df, arquivos_df, on='id_paciente')
```

Tabela 3: *Dataframe* de mesclados

	id_paciente	Comorbidade	inicio_ciclo	fim_ciclo	eventos	nome_arquivo
0	101	URTI	0.036	0.579	Nenhum	101_1b1_Al_sc_Meditron
1	101	URTI	0.579	2.450	Nenhum	101_1b1_Al_sc_Meditron
2	101	URTI	2.450	3.893	Nenhum	101_1b1_Al_sc_Meditron
...	...	...	...	...	...	...
6895	226	Pneumonia	15.536	17.493	Nenhum	226_1b1_Pl_sc_LittC2SE
6896	226	Pneumonia	17.493	19.436	Estertor	226_1b1_Pl_sc_LittC2SE
6897	226	Pneumonia	19.436	19.979	Nenhum	226_1b1_Pl_sc_LittC2SE

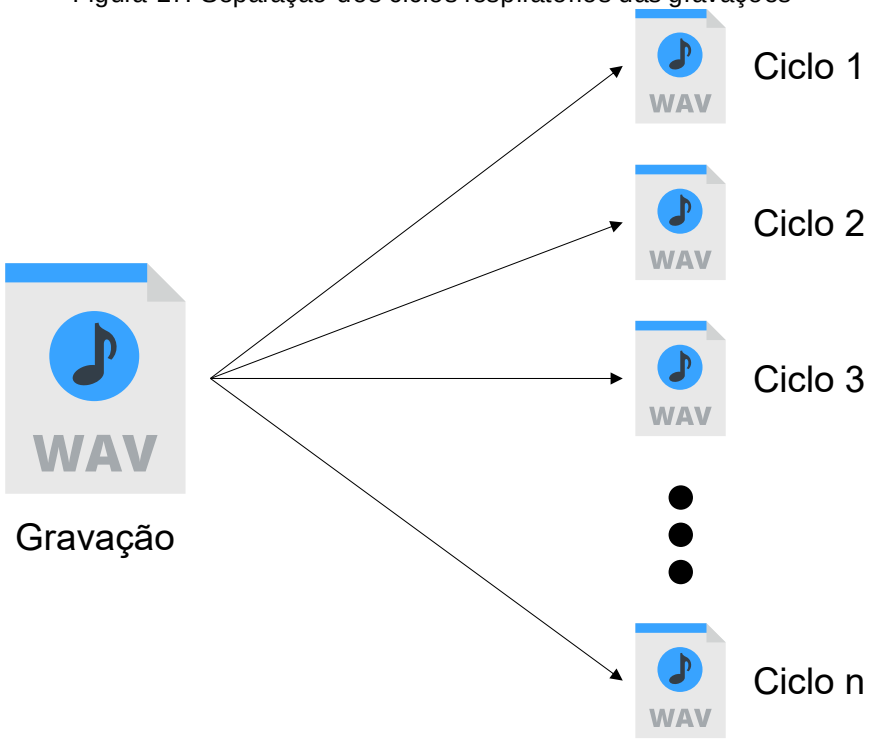
Fonte: o autor (2021)

Conforme observa-se na Tabela 3, na coluna ‘nome_arquivo’, as três primeiras linhas possuem o mesmo valor (101_1b1_Al_sc_Meditron). Isso também acontece nas últimas três colunas) e em diversos outros casos, pelo fato de que originalmente havia somente 920 arquivos. Então, é necessário associar estes valores aos novos arquivos que serão gerados uma vez que os ciclos respiratórios forem separados em novos arquivos de áudio.

3.2.1 Separação dos ciclos respiratórios

Não é possível utilizar os arquivos de áudio fornecidos diretamente como entradas da rede neural, devido ao fato de os eventos estarem rotulados em intervalos dentro desses arquivos. Sendo assim, foi necessário gerar novos arquivos a partir das gravações, conforme ilustrado na Figura 27.

Figura 27: Separação dos ciclos respiratórios das gravações



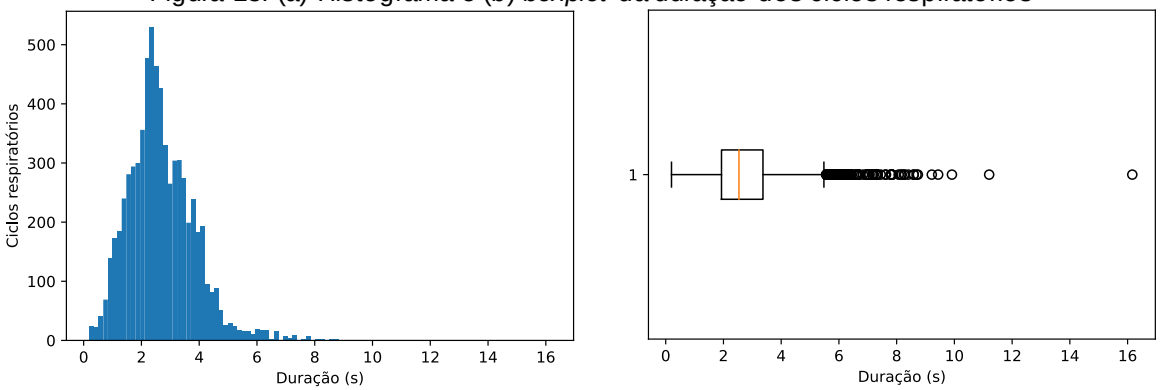
Fonte: o autor (2021)

Os ciclos respiratórios possuem durações distintas. Isso pode ser verificado fazendo a diferença entre os tempos finais e iniciais de cada ciclo. Entretanto, uma rede neural convolucional requer que todas as suas entradas possuam a mesma dimensão. Assim, foi necessário determinar uma duração padrão para os novos arquivos.

3.2.1.1 Duração dos ciclos respiratórios

Por meio dos gráficos da Figura 28, determinou-se que 6 segundos é uma duração razoável, pois engloba a maior parte dos dados. Desta forma, tem-se muitos ciclos respiratórios que possuem menos de 6 segundos de informação e alguns que serão encurtados para se encaixar no critério.

Figura 28: (a) Histograma e (b) *boxplot* da duração dos ciclos respiratórios



(a) (b)

Fonte: o autor (2021)

O fragmento de código abaixo mostra a implementação da geração de novos arquivos dos ciclos respiratórios.

```
# função para fazer a extração dos ciclos respiratórios

def ciclo_respiratorio(gravacao, inicio, fim, sr=22050):
    duracao = len(gravacao)
    inicio = min(int(inicio*sr), duracao)
    fim = min(int(fim*sr), duracao)
    ciclo = gravacao[inicio, fim]
    return ciclo

##########################################################################################
# gerar arquivos de audio de ciclos respiratorios

os.makedirs('ciclos_respiratorios')

i = 0
path = 'audio_and_txt_files/'

for index, row in tqdm(data.iterrows()):
    duracao = 6
    inicio = row['inicio_ciclo']
    fim = row['fim_ciclo']
    arquivo = row['nome_arquivo']

    # se a duração > max_dur, alterar
    if fim - inicio > duracao:
        fim = inicio + duracao

    # verificar se há mais de um ciclo na gravação
    if index > 0:
        if data.iloc[index-1]['nome_arquivo'] == arquivo:
            i = i + 1
        else:
            i = 0
    wav = path + arquivo + '.wav'

    arquivo = arquivo + ' ' + str(i) + '.wav'

    salvar = 'ciclos_respiratorios/' + arquivo

    sinal, sr = librosa.load(wav) #sampleRate padrão é 22050
    ciclo = ciclo_respiratorio(sinal, inicio, fim, sr)

    # padding: preencher com 0 se a duração < max dur e centralizar os dados
    ciclo_pad = librosa.util.pad_center(ciclo, 6*sr)

    # gerar arquivo
    sf.write(file=salvar, data=ciclo_pad, samplerate=sr)
```

Por fim, foi necessário voltar a Tabela 3 e modificar os dados da coluna ‘nome_arquivo’ com os nomes dos arquivos gerados. Agora, cada linha apontar para o arquivo de áudio correspondente ao ciclo respiratório que representa. Além disso foram removidas as colunas ‘estertor’ e ‘sibilo’, pois a coluna ‘eventos’ substitui bem estas duas. Assim, o *dataframe* final simplificado pode ser verificado na Tabela 4.

Tabela 4: *Dataframe* final

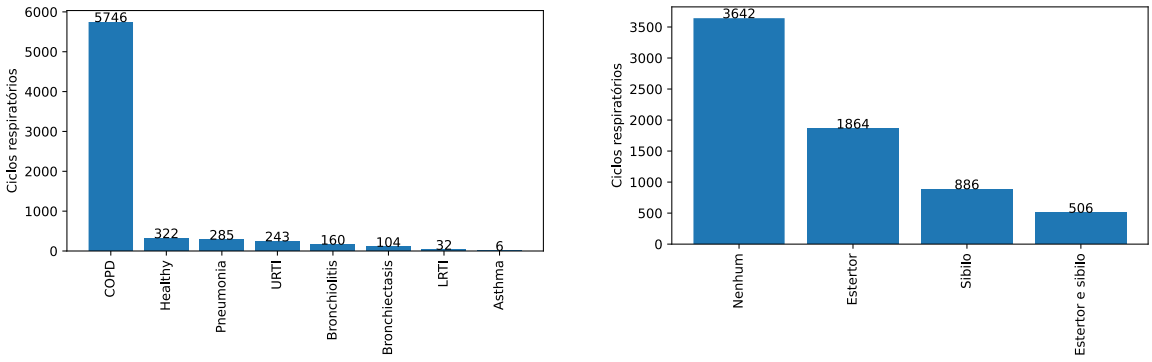
	id_paciente	comorbidade	eventos	nome_arquivo
0	101	URTI	Nenhum	101_1b1_Al_sc_Meditron_0.wav
1	101	URTI	Nenhum	101_1b1_Al_sc_Meditron_1.wav
2	101	URTI	Nenhum	101_1b1_Al_sc_Meditron_10.wav
...	...	...	...	...
6895	226	Pneumonia	Nenhum	226_1b1_Pl_sc_LittC2SE_7.wav
6896	226	Pneumonia	Estertor	226_1b1_Pl_sc_LittC2SE_8.wav
6897	226	Pneumonia	Nenhum	226_1b1_Pl_sc_LittC2SE_9.wav

Fonte: o autor (2021)

3.2.2 Dados de treino e teste

Antes de treinar o modelo é importante verificar o balanceamento das classes. Na Figura 29a se pode observar que a maioria dos ciclos respiratórios está relacionada a DPOC e que o desbalanceamento é muito alto, com asma sendo a comorbidade menos favorecida. Ao analisar o balanceamento relativo à coluna de eventos (Figura 29b), nota-se um cenário similar, porém as classes possuem mais representatividade.

Figura 29: Quantidade de ciclos respiratórios (a) por comorbidade e (b) evento



(a) (b)

Fonte: o autor (2021)

Sendo assim, para os dois cenários, é necessário estratificar os dados para garantir que cada classe tenha proporcionalidade relevante, evitando que o modelo fique enviesado para as classes dominantes.

Os dados foram distribuídos da seguinte maneira: 25% para testes (validação) e 75% para treino, conforme o fragmento de código a seguir:

```
from sklearn.model_selection import train_test_split

Xtrain,     Xval,     ytrain,     yval     =     train_test_split(dados_df,     dados_df.eventos,
stratify=dados_df.eventos, random_state=42, test_size=0.25)
```

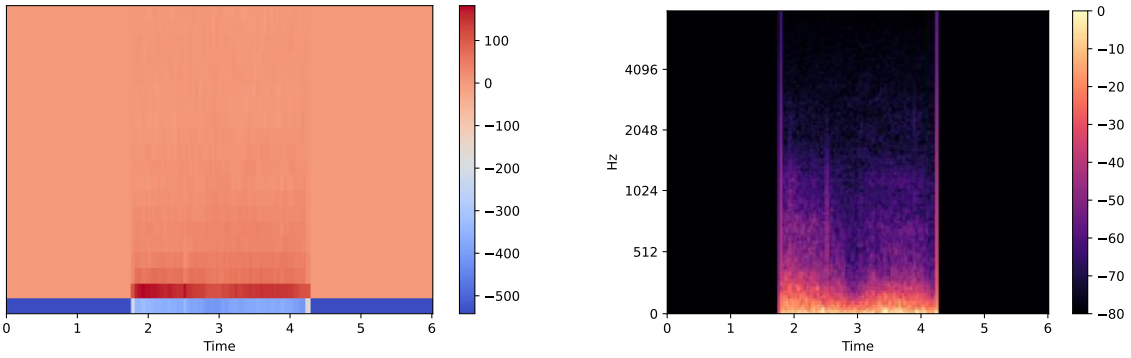
3.3 EXTRAÇÃO DE CARACTERÍSTICAS

Os coeficientes mel-cepstrais (MFCCs) e os espectrogramas em escala Mel serão as características passadas como entradas para a CNN. O fragmento de código a seguir mostra uma função para adquirir estes dados por meio da Librosa.

```
# extrair caracteristicas

def caracteristicas(path):
    sinal, sr = librosa.load(path)
    mfcc = librosa.feature.mfcc(y=sinal,sr=sr)
    mel spec=librosa.feature.melspectrogram(y=sinal,sr=sr)
    return mfcc, mel_spec

##########################################################################################

path = 'ciclos_respiratorios/'
mfcc, mel spec = [],[]

for idx,row in tqdm(val.iterrows()): # mudar  'val' para 'train' para obter dados de treino
    audio = path + row['nome arquivo']
    coefs, specs = caracteristicas(audio)
    mfcc.append(coefs)
    mel spec.append(specs)

mfcc_val = np.array(mfcc)
mel_spec_val = np.array(mel_spec)
```

A exemplo, a Figura 30 mostra a visualização das características extraídas pelo fragmento de código acima. O exemplo em questão é o arquivo do ciclo respiratório ‘130_2b2_Ll_mc_AKGC417L_3.wav’, no qual há presença de estertor.

Figura 30: Visualização de características (a) MFCC e (b) espectrograma



(a) (b)

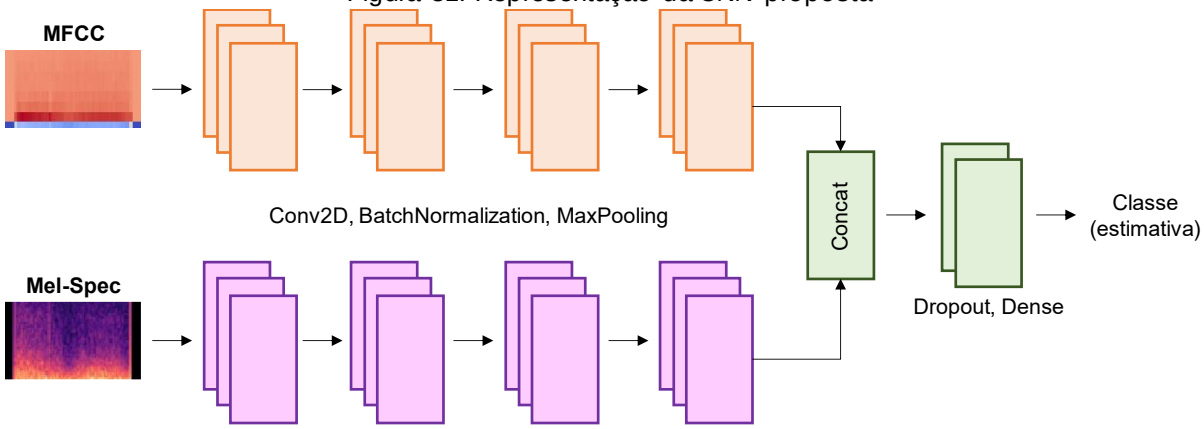
Fonte: o autor (2021)

É interessante notar que nas duas imagens é possível visualizar que a duração deste ciclo é pouco mais de 2 segundos. E, conforme explanado no item 3.2.1, toda informação foi centralizada e as extremidades foram preenchidas com zero, representando silêncio.

3.4 CRIAÇÃO DO MODELO

Como a CNN será alimentada tanto por MFCCs quanto espectrogramas, o modelo precisa ter múltiplas entradas (Figura 31).

Figura 31: Representação da CNN proposta



Fonte: o autor (2021).

O fragmento de código a seguir mostra como foi feito o modelo sequencial para a entrada MFCCs, que consiste em camadas de convolução e *pooling*, além de *bach normalization* (normalização em lote) para tornar a rede mais rápida e mais estável. A ativação das camadas convolucionais é relu e o padding é *same* (preenchimento com zeros).

```
# modelo para MFCCs

mfcc_input=keras.layers.Input(shape=(20,259,1),name="mfccInput")
x=keras.layers.Conv2D(32,5,strides=(1,3),padding='same')(mfcc_input)
x=keras.layers.BatchNormalization()(x)
x=keras.layers.Activation(keras.activations.relu)(x)
x=keras.layers.MaxPooling2D(pool_size=2,padding='valid')(x)

x=keras.layers.Conv2D(64,3,strides=(1,2),padding='same')(x)
x=keras.layers.BatchNormalization()(x)
x=keras.layers.Activation(keras.activations.relu)(x)
x=keras.layers.MaxPooling2D(pool_size=2,padding='valid')(x)

x=keras.layers.Conv2D(96,2,padding='same')(x)
x=keras.layers.BatchNormalization()(x)
x=keras.layers.Activation(keras.activations.relu)(x)
x=keras.layers.MaxPooling2D(pool_size=2,padding='valid')(x)

x=keras.layers.Conv2D(128,2,padding='same')(x)
x=keras.layers.BatchNormalization()(x)
x=keras.layers.Activation(keras.activations.relu)(x)
mfcc_output=keras.layers.GlobalMaxPooling2D()(x)

mfcc_model=keras.Model(mfcc_input, mfcc_output, name="mfccModel")
```

O código para os espectrogramas é similar ao apresentado anteriormente, mudando apenas as variáveis e nomes e a dimensão da entrada. Por fim, os dois modelos são concatenados e são adicionadas algumas camadas densas totalmente

conectadas. Antes de cada camada densa, são adicionadas camadas *dropout* com o intuito de prevenir o sobreajuste.

```
input_mfcc=keras.layers.Input(shape=(20,259,1),name="mfcc")
mfcc=mfcc_model(input_mfcc)

input_mSpec=keras.layers.Input(shape=(128,259,1),name="mspec")
mSpec=mSpec_model(input_mSpec)

concat=keras.layers.concatenate([mfcc,mSpec])
hidden=keras.layers.Dropout(0.2)(concat)
hidden=keras.layers.Dense(50,activation='relu')(concat)
hidden=keras.layers.Dropout(0.3)(hidden)
hidden=keras.layers.Dense(25,activation='relu')(hidden)
hidden=keras.layers.Dropout(0.3)(hidden)
output=keras.layers.Dense(4,activation='softmax')(hidden) #alterar numero de classes

net=keras.Model([input_mfcc,input_mSpec], output, name="Net")
```

O fragmento de código acima mostra a saída do modelo utilizado para classificar os eventos. Como há 4 classes possíveis, esta camada tem somente 4 neurônios. No caso da classificação de comorbidades, este parâmetro foi alterado para 8. A saída do modelo é uma camada densa de ativação *softmax*, devido à natureza do problema de classificação de múltiplas classes.

3.4.1 Compilar e treinar modelo

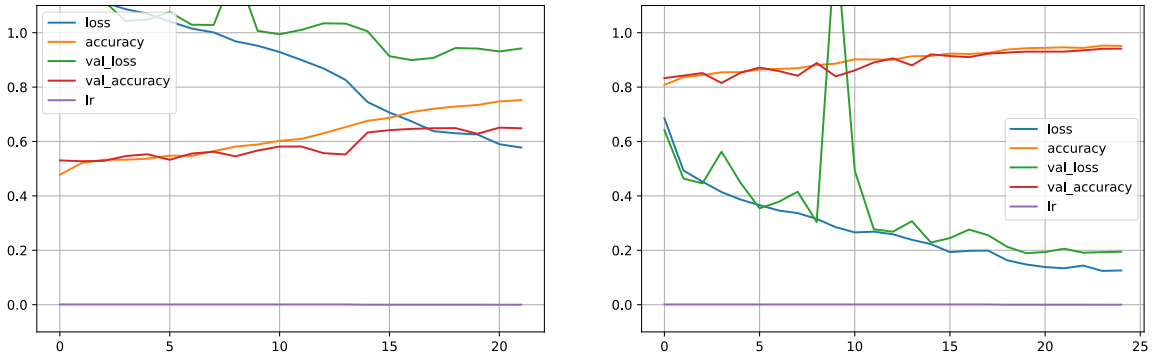
Uma vez que o modelo foi criado, é preciso compila-lo para especificar a função de perda (*loss*) e o otimizador a ser usado. A função de perda é 'sparse_categorical_crossentropy' porque cada amostra pertence a apenas uma classe. O fragmento de código a seguir a seguir mostra como o modelo foi compilado e treinado.

```
from keras import backend as K

net.compile(loss='sparse_categorical_crossentropy', optimizer='nadam', metrics=['accuracy'])
K.set_value(net.optimizer.learning_rate, 0.001)

import tensorflow_addons as tfa

my_callbacks = [
    tfa.callbacks.TQDMProgressBar(),
    tf.keras.callbacks.EarlyStopping(patience=5),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',      factor=0.1,      patience=3,
min_lr=0.00001,mode='min')]

history=net.fit(
    {"mfcc":mfcc_train,"mspec":mel_spec_train}, ytrain,
    validation data=({"mfcc":mfcc val,"mspec":mel spec val}, yval),
    epochs=100, verbose=0, callbacks=my_callbacks)
```

Aqui é importante se atentar ao uso de *callbacks* durante o treinamento, com destaque ao *Early Stopping*, cuja função é parar o treinamento caso o modelo não esteja mais aprendendo. Deste modo, o modelo nunca é treinado durante as 100 épocas, pois sua precisão não terá melhora. A Figura 32 mostra as curvas de

aprendizado para o modelo ao classificar eventos e as comorbidades nos ciclos respiratórios.

Figura 32: Curvas de aprendizado (a) para eventos e (b) comorbidades



(a) (b)

Fonte: o autor (2021)

3.4.2 Avaliar o modelo e fazer previsões

O seguinte fragmento de código mostra o procedimento realizar a avaliação do modelo e como estimar a classe de novos dados:

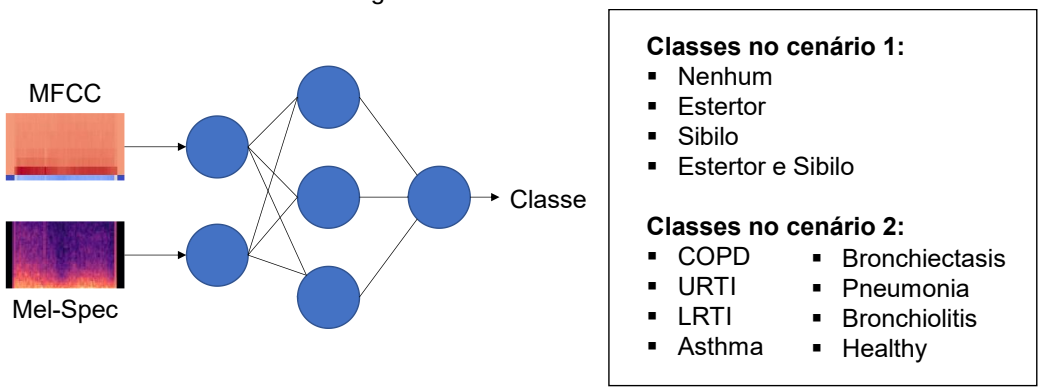
```
# avaliação
net.evaluate({"mfcc":mfcc_val,"mspec":mel_spec_val},yval)

# previsoes
ypred = net.predict({"mfcc":mfcc_val,"mspec":mel_spec_val})
ypred = np.argmax(ypred, axis=1
labels = dados_df['eventos'].unique().tolist()

# metricas
from sklearn.metrics import ConfusionMatrixDisplay, f1_score, classification_report
print(classification_report(yval, ypred, target_names=labels))

f1 = f1_score(yval, ypred, average='macro')
np.mean(f1)
```

O modelo será avaliado quanto a sua acurácia e F1 score. Deste ponto, cabe realizar a interpretação desses dados para tirar conclusões a respeito da performance.

4 RESULTADOS E DISCUSSÃO

Neste item serão apresentados os resultados explorados nos dois cenários descritos na metodologia. A Figura 33 ilustra o processo.

Figura 33: Cenários de teste

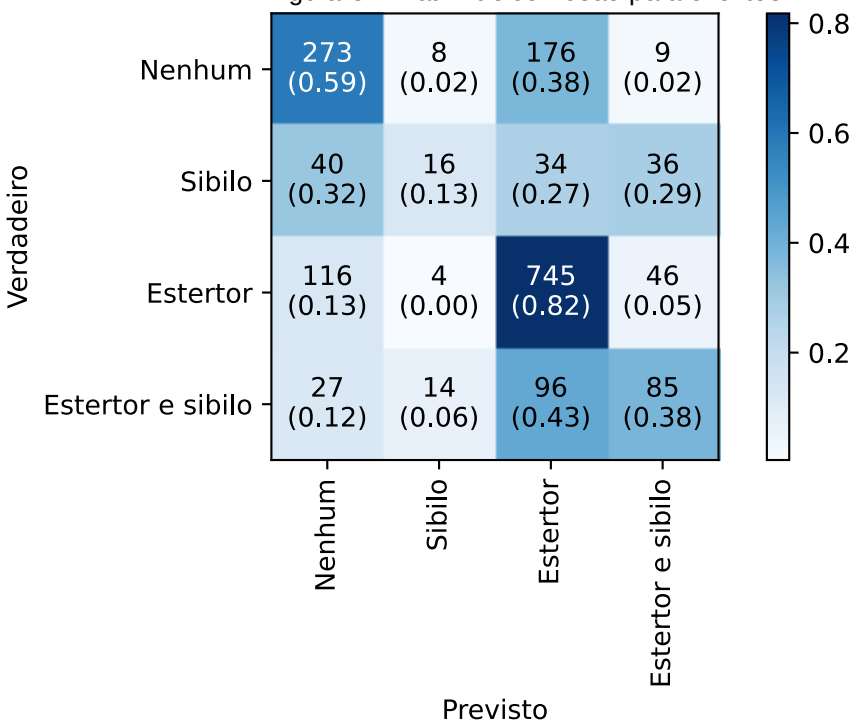


Fonte: o autor (2021)

4.1 CLASSIFICAÇÃO DE EVENTOS

Neste cenário, o modelo treinou por 22 épocas, quando ocorreu *early stopping*. Durante o treino a acurácia foi 75% e a perda ficou aproximadamente 58%. Na validação a acurácia foi 65%, enquanto a perda foi 94%. Esta grande divergência nas perdas das duas etapas indica que o modelo generaliza mal, ou seja, tende a fazer previsões ruins para novos dados.

Figura 34: Matriz de confusão para eventos



Fonte: o autor (2021)

Analisando a matriz de confusão (Figura 34), é possível ter uma ideia mais detalhada (classe a classe) da performance do modelo. Nota-se que o modelo é melhor em prever a presença de estertores (acurácia de 82%), mas ainda assim faz estimativas incorretas em todas as classes.

É sabido que os dados deste conjunto são muito desbalanceados. Assim, a acurácia é uma métrica ruim, pois pode levar a conclusões errôneas.

	precision	recall	f1-score	support
Nenhum	0.60	0.59	0.59	466
Sibilo	0.38	0.13	0.19	126
Estertor	0.71	0.82	0.76	911
Estertor e sibilo	0.48	0.38	0.43	222
accuracy			0.65	1725
macro avg	0.54	0.48	0.49	1725
weighted avg	0.63	0.65	0.63	1725

Analisando o F1-score, obtém-se uma performance geral de 49%, o que apesar de desanimador, é mais realista. Acima é mostrado um resumo do F1-score para cada uma das classes.

4.2 CLASSIFICAÇÃO DE COMORBIDADES

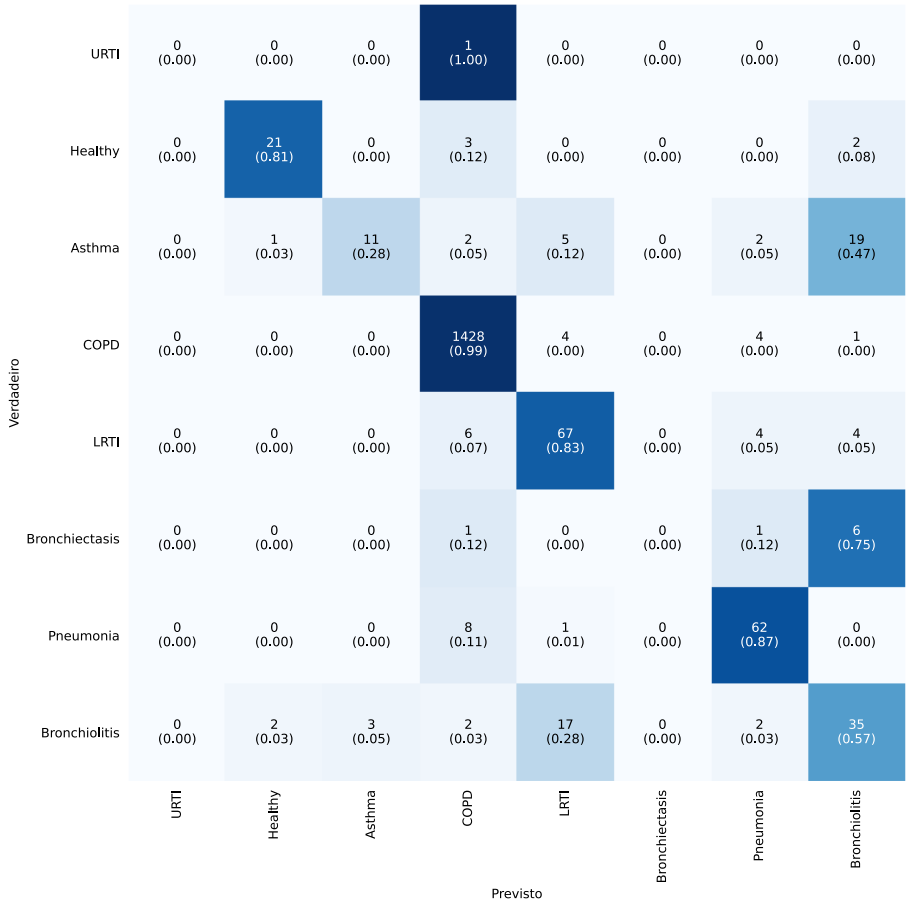
No segundo cenário, foram classificadas as comorbidades rotuladas no *dataset*. O resultado se resume na matriz de confusão da Figura 35. Desta vez, o *early stopping* aconteceu após 25 épocas, de maneira similar ao cenário anterior. No treino, a acurácia foi de 95% e na validação foi 94%. As perdas durante o treino e validação foram, respectivamente, 11% e 19%. Sendo assim, entende-se que desta forma o modelo aprendeu melhor do que antes, visto que diferença das perdas é menor e as acurácias são simulares.

Analisando a matriz de confusão, percebe-se que este modelo é muito bom em classificar DPOC. Contudo, é fácil de ver que a classificação é penalizada conforme a quantidade de amostras diminui. Para os rótulos URTI e *Bronchiectasis*, por exemplo, não houve acertos.

Tal como o primeiro caso, neste cenário ainda existe o problema do balanceamento de classes, de tal modo que é mais interessante analisar outra métrica para tirar conclusões. O F1-score geral é de 55%, similar aquele visto anteriormente, o que é de se esperar, visto que se trata do mesmo modelo. As diferenças são a

presença de mais neurônios na camada de saída, para comportar todas as classes possíveis e os conjuntos de treino e validação, uma vez que houve a redistribuição dos dados com a mudança da quantidade de classes.

Figura 35: Matriz de confusão para comorbidades



Fonte: o autor (2021)

A seguir, apresenta-se o resumo das métricas para o segundo cenário após a validação:

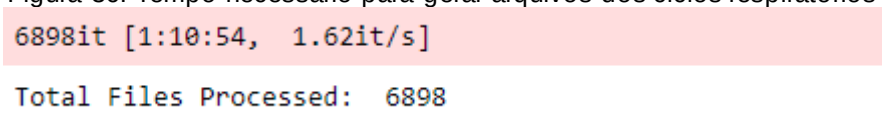
	precision	recall	f1-score	support
URTI	0.00	0.00	0.00	1
Healthy	0.88	0.81	0.84	26
Asthma	0.79	0.28	0.41	40
COPD	0.98	0.99	0.99	1437
LRTI	0.71	0.83	0.77	81
Bronchiectasis	0.00	0.00	0.00	8
Pneumonia	0.83	0.87	0.85	71
Bronchiolitis	0.52	0.57	0.55	61
accuracy			0.94	1725
macro avg	0.59	0.54	0.55	1725
weighted avg	0.94	0.94	0.94	1725

4.3 TEMPO DE PROCESSAMENTO

O tempo de processamento é uma variável extremamente dependente do *hardware* utilizado. Fatores como processador e velocidades de leitura e escrita do armazenamento são os que mais influenciam. Os valores expostos aqui foram obtidos com o *hardware* descrito no item 2.3.

A etapa mais longa é a de geração dos arquivos .wav para cada ciclo respiratório. Ela consiste em ler cada arquivo de áudio fornecido no *dataset* e escrever novos arquivos referentes a cada ciclo contido nas gravações. A Figura 36 mostra que foi preciso pouco mais de 1 hora, sendo que cada iteração dura menos de 2 segundos.

Figura 36: Tempo necessário para gerar arquivos dos ciclos respiratórios



Fonte: o autor (2021).

A extração de características é bem rápida: para o cenário de eventos foram necessários 2 minutos para processar os conjuntos de treino e teste. Já no cenário das comorbidades, o tempo foi 2 minutos e 30 segundos. É natural que esta etapa seja mais rápida uma vez que apenas operações com números estão sendo feitas, o que exige bem menos do computador. O tempo necessário no treino do modelo depende da quantidade de épocas executadas. Cada época levou aproximadamente 17 segundos para ser completada.

Recomenda-se utilizar o TQDM (módulo Python) para acompanhar o progresso de todas as etapas. Caso contrário, etapas mais longas podem causar a impressão de travamentos – especialmente o treinamento dos modelos, que utiliza bastante recursos de CPU.

Todo o conteúdo necessário para a reprodução (notebooks, arquivos de áudio e arquivos de anotação) pode ser encontrado no seguinte endereço: https://drive.google.com/drive/folders/1qpd1zrvVoDKij8cEvAhc322AczvQbKSL?usp=sharing. Em caso de problemas com o link fornecido, é possível contactar o autor em mscl.eng@uea.edu.br. É importante ressaltar que ao reproduzir esta pesquisa em um ambiente diferente do Windows, pequenas adaptações podem ser necessárias.

CONCLUSÃO

A tarefa de classificar sons de qualquer natureza é desafiadora e empolgante, pois requer conhecimento de temas como processamento digital de sinais, características dos sinais sonoros e de aprendizado de máquina. Ao longo do desenvolvimento foram feiras revisões bibliográficas sobre esses temas e experimentos práticos que permitiram aplicar a teoria explorada no referencial teórico. A metodologia proposta no projeto de pesquisa pode ser aplicada e os recursos e custos mantiveram-se inalterados.

Dos objetivos propostos, a extração de características de sinais de áudio foi atingida. Os demais, foram atingidos parcialmente, uma vez que se nota que o modelo elaborado classifica bem alguns casos – estertor no primeiro cenário e algumas doenças no segundo cenário – mas ainda erra bastante, sendo notável a necessidade de refinamento. A razão pela qual os resultados obtidos foram inferiores aos desejados pode ter muitos contribuintes: os dados desbalanceados, as características escolhidas ou até mesmo o modelo desenvolvido. É difícil determinar qual fator tem maior influência, devido a necessidade de fazer testes em cenários muito distintos, o que inviabiliza o tempo que pode ser gasto na execução da pesquisa.

É pertinente ressaltar que vale a pena investigar os resultados mais a fundo a fim de fazer uma análise mais realista da performance do modelo proposto. Embora o resultado geral (acurácia e F1-score) da classificação de eventos seja baixo, a classe de estertores mostrou boas métricas. Da maneira complementar, o resultado da classificação de comorbidades apresentou uma acurácia fantástica e as classes *Healthy*, *COPD*, *LRTI* e *Pneumonia* tiveram métricas boas. Contudo, verificou-se que a performance do modelo é baixa ao analisar o F1-score geral.

Os arquivos de áudio dos ciclos respiratórios, gerados a partir daqueles fornecidos no *dataset* são uma contribuição desta pesquisa, uma vez que já estão padronizados em uma mesma taxa de amostragem e normalizados. Ter estes arquivos prontos já auxilia a quem quiser reproduzir integralmente ou realizar uma pesquisa similar utilizando a mesma base, uma vez que economiza bastante tempo que pode ser investido em análises de resultados ou ajustes aos modelos. Estes arquivos podem ser baixados no mesmo endereço fornecido previamente.

Para trabalhos futuros, sugere-se a elaboração de um novo conjunto de dados, que seja mais balanceado e que possua mais amostras. Além disso, alternativas interessantes seriam analisar a performance de outras características além de MFCCs e espectrogramas e até mesmo explorar outras redes neurais. É possível que redes capazes de interpretar séries temporais com entradas de dimensões variáveis sejam melhores nesta tarefa, uma vez que não seria necessário utilizar duração fixa nos arquivos de entrada. Sugere-se também que as técnicas exploradas sejam aplicadas em outros campos, como a detecção de pragas em plantações através do som.

REFERÊNCIAS

3M Littmann Cardiology IV. **3M Loja Oficial**. Disponível em: <https://www.loja3m.com.br/estetoscopio-3m-littmann-cardiology-iv-6179-preto-e-champagne-7891040241750/p>. Acesso em: 11 Maio 2021, il.

AYKANAT, M. et al. **Classification of lung sounds using convolutional neural networks**. EURASIP Journal on Image and Video Processing, 11 Setembro 2017.

CHAUDHARI, G. et al. **Virufy: Global Applicability of Crowdsourced and Clinical Datasets for AI Detection of COVID-19 from Cough**, 2020.

CHILDERS, D. G.; SKINNER, D. P.; KEMERAIT, R. C. The Cepstrum: A Guide to Processing. **Proceeding of the IEEE**, 65, n. 10, Outubro 1977. 1428-1443.

CHRISTENSEN, M. G. **Introduction to Audio Processing**. 1. ed. Cham: Springer International Publishing, 2019.

DINIZ, P. S. R.; DA SILVA, E. A. B.; NETTO, S. L. **Processamento Digital de Sinais:** Projeto e Análise de Sistemas. 2. ed. Porto Alegre: Bookman, 2014.

DOSHI, K. **Audio Deep Learning Made Simple (Part 2)**: Why Mel Spectrograms perform better. towards data science, 2021. Disponível em: <https://towardsdatascience.com/audio-deep-learning-made-simple-part-2-why-mel-spectrograms-perform-better-aad889a93505>. Acesso em: 15 Junho 2021.

GÉRON, **A. Mãos à obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow**. Tradução de Rafael Contatori. Rio de Janeiro: Alta Books, 2019.

GRUS, J. **Data Sciente do Zero**. Tradução de Wellington Nascimento. Rio de Janeiro: Alta Books, 2016.

HAFKE-DYS, H. et al. **The accuracy of lung auscultation in the practice of physicians and medical students**. PLoS ONE, 12 Agosto 2019.

JAKE. **Fourier Transform**. PGFplots.net, 2021. Disponível em: <https://pgfplots.net/fourier-transform/>. Acesso em: 28 Julho 2021, il.

JEON, H. et al. **Area-Efficient Short-Time Fourier Transform Processor for Time–Frequency Analysis of Non-Stationary Signals**. Applied Sciences, 21 Outubro 2020, il.

LAENNEC'S stethoscope. Wellcome Collection. Disponível em: <https://wellcomecollection.org/works/a9r7h4py>. Acesso em: 11 Maio 2021, il.

NAVES, R. **UM ESTUDO DE RECONHECIMENTO DE SONS PULMONARES BASEADO EM TÉCNICAS DE INTELIGÊNCIA COMPUTACIONAL**, Lavras, 2015.

NGUYEN, T.; PERNKOPF, F. **Lung Sound Classification Using Snapshot Ensemble of Convolutional Neural Networks**. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Montreal, 2020. 760-763.

OPPENHEIM, A. V.; SCHAFER, R. W. **Processamento em tempo discreto de sinais**. Tradução de Daniel Vieira. 3. ed. São Paulo: Person, 2012.

PALANIAPPAN, R.; SUNDARAJ, K.; AHAMED, N. U. **Machine learning in lung sound analysis:** A systematic review. Biocybernetics and Biomedical Engineering, 33, n. 3, 2013. 129-135.

PORTO, C. C.; PORTO, A. L. (Eds.). **EXAME Clínico**. 7. ed. Rio de Janeiro: Guanabara Koogan, 2016.

PRIFTIS, K. N.; HADJILEONTIADIS, L. J.; EVERARD, M. L. (Eds.). **Breath Sounds:** From Basic Science to Clinical Practice. 1. ed. [S.l.]: Springer, 2018.

RAO, K. S.; VUPPALA, A. K. **Speech Processing in Mobile Environments**. 1. ed. [S.l.]: Springer, 2014.

ROCHA, B. M. et al. **A Respiratory Sound Database for the Development of Automated Classification**. Precision Medicine Powered by pHealth and Connected Health, Singapura, 17 Novembro 2017.

SANAR RESIDÊNCIA MÉDICA. **Pneumologia:** residência, áreas de atuação, rotina e mais! Sanarmed. Disponivel em: <https://www.sanarmed.com/pneumologia-residencia-areas-de-atuacao-rotina-e-mais>. Acesso em: 11 Maio 2021, il.

SCUDILIO, J. **Qual a melhor métrica para avaliar os modelos de Machine Learning?** flai, 2020. Disponivel em: <https://www.flai.com.br/juscudilio/qual-a-melhor-metrica-para-avaliar-os-modelos-de-machine-learning/>. Acesso em: 10 Dezembro 2021.

SECRETARIA DE VIGILÂNCIA EM SAÚDE. **Principais causas de morte**. Secretaria de Vigilância em Saúde, 2017. Disponível em: <http://svs.aids.gov.br/dantps/centrais-de-conteudos/paineis-de-monitoramento/mortalidade/gbd-brasil/principais-causas/>. Acesso em: 30 Maio 2021.

SENGUPTA, N.; SAHIDULLAH, M.; SAHA, G. **Lung sound classification using cepstral-based statistical features**. Computers in Biology and Medicine, 20 Maio 2016.

SHARMA, G.; UMAPATHY, K.; KRISHNAN, S. **Trends in audio signal feature extraction methods**. Applied Acoustics, 1 Setembro 2019.

STEVENS, S. S.; VOLKMANN, J.; NEWMAN, E. B. **A Scale for the Measurement of the Psychological Magnitude Pitch**, 4 Agosto 1936.

THOM, R. A. **Laennec and the stethoscope**. U.S. National Library of Medicine, 1960. Disponível em: <https://collections.nlm.nih.gov/catalog/nlm:nlmuid-101651398-img>. Acesso em: 11 Maio 2021, il.

WORLD HEALTH ORGANIZATION. **The top 10 causes of death**. World Health Organization, 2020. Disponível em: <https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death>. Acesso em: 30 Maio 2021.