Artigo recebido em 05/10/2021.
Publicado em 20/01/2022.

Revista SODEBRAS – Volume 17
N° 193 – JANEIRO/ 2022

DOI: https://doi.org/10.29367/issn.1809-3957.17.2022.193.20

PRE-PROCESSING TECHNIQUES FOR RESPIRATORY SOUNDS IN IA CLASSIFIERS

TÉCNICAS DE PRÉ-PROCESSAMENTO PARA SONS RESPIRATÓRIOS EM CLASSIFICADORES IA

César Augusto Menezes Nepomuceno[1]
Moisés Pereira Bastos[2]
João Paulo Campos Mendes[3]
Victor Vermehren Valenzuela[4]
Daniel Akio Chen[5]
Vitoriano Medeiros Casas[6]

Abstract - An effort in the research and development of autonomous systems for classification of breath sounds, aiming to assist in the early detection and treatment of pathologies, has become increasingly recurrent. In this sense, this work studied pre-processing techniques that power autonomous classifiers, in order to leverage such technologies in our country. To this end, research was conducted to survey the main methods of data preparation as well as the practical implementation of various techniques using python, a common language in the field of artificial intelligence. As a result, we obtained a guide that aims to facilitate the theoretical and practical understanding for researchers in this area.

Keywords: Artificial Intelligence. Preprocessing. Classifiers.

Resumo – Um esforço na pesquisa e desenvolvimento de sistemas autônomos de classificação de sons respiratórios, visando auxiliar na detecção e tratamento precoce de patologias, tem se tornado cada vez mais recorrente. Nesse sentido, este trabalho estudou técnicas de pré-processamento que alimentam classificadores autônomos, no sentido de alavancar tais tecnologias em nosso país. Para tal, foram realizadas pesquisas no sentido de levantar os principais métodos de preparação de dados como também realizar a implementação prática de diversas técnicas utilizando python, linguagem comum no ramo da inteligência artificial.

[1] Universidade do Estado do Amazonas. Contato: camn.lic@uea.edu.br;
[2] Universidade do Estado do Amazonas. Contato: mpbastos@uea.edu.br;
[3] Universidade do Estado do Amazonas. Contato: jpcmendes@uea.edu.br;
[4] Universidade do Estado do Amazonas. Contato: vvalenzuela@uea.edu.br;
[5] Universidade do Estado do Amazonas. Contato: dac.eng17@uea.edu.br;
[6] Universidade do Estado do Amazonas. Contato: vmc.eai18@uea.edu.br.

Como resultado, obteve-se um guia que visa facilitar o entendimento teórico e prático para pesquisadores embasados nesta área.

Palavras-chave: Inteligência artificial. Pré-processamento. Classificadores.

I. INTRODUÇÃO

As doenças respiratórias têm causado cada vez mais problemas para os sistemas de saúde uma vez que representam a terceira causa de mortes em todo o mundo. Com isso, grandes esforços na pesquisa têm focado no diagnóstico precoce e monitoramento contínuo em pacientes com problemas respiratórios nos últimos anos (MARQUES, OLIVEIRA E JÁCOME, 2014).

O estetoscópio é o principal método de ausculta pulmonar, que deve ser manuseado por um médico especialista, e apesar de ter se tornado digital, ainda apresenta algumas desvantagens como por exemplo a falta de um monitoramento contínuo, a necessidade de um especialista para detectar os sons de maneira adequada, entre outras incapacidades dessa ferramenta (ROCHA et. al., 2019). Logo, a possibilidade de detectar de maneira precisa sons adventícios, como sibilos e estalos, é um fator determinante em um exame médico para diagnosticar e tratar doenças respiratórias (GRZYWALSKi et. al, 2019).

Métodos automatizados podem ser úteis na detecção precoce de sons com anomalia, podendo auxiliar no tratamento precoce futuramente. Neste cenário, técnicas de machine learning, uma área da IA (Inteligência Artificial), foram introduzidas obtendo um desempenho robusto na detecção de sons em geral. Direcionado para a classificação de sons pulmonares, métodos desta área computacional alimentam os recursos do espectrograma em classificadores autônomos que exploram arquiteturas de rede poderosas como por exemplo a CNN (Convolutional Neural Network) (GRZYWALSKI et. al, 2019).

Com isso, o objetivo deste trabalho foi estudar de maneira teórica e prática as técnicas mais comuns de pré-processamento que alimentam tais classificadores no ramo da IA.

II. METODOLOGIA

Neste artigo são apresentados fundamentos teóricos e exemplos práticos na utilização de técnicas de pré-processamento para classificação de sons respiratórios utilizando IA. Tais técnicas são comumente utilizadas na preparação do espectrograma que alimenta o classificador.

2.1 – Base de Dados

Todo classificador necessita de uma quantidade muito grande de dados de entrada para realizar treinamento e posterior classificação, sendo comumente encontrados em base de dados que possuem informações diversas do tipo imagem, sons e etc. Neste estudo foi utilizada uma base de dados de sons respiratórios frequentemente usada em classificadores de sons pulmonares, sendo desenvolvida para um desafio científico na Conferência Internacional de Informática Biomédica e em Saúde (ICBHI) que aconteceu em 2017 (ROCHA et. al., 2019).

O conjunto de dados supracitado representa uma coletânea de amostras de sons respiratórios, coletadas de maneira independente por duas equipes de pesquisadores em dois países diferentes. Possui 920 áudios respiratórios e para tal contou com a participação de 126 pessoas, sendo 77 adultos e 49 crianças. A tabela 1 apresenta mais informações relevantes sobre a base de dados (ROCHA et. al., 2019).

Tabela 1 – Informações relevantes da base de dados (NA: Não Disponível).

Número de Gravações	920
Frequência de Amostragem	4 KHz (90); 10 KHz (6); 44.1 (824)
Bits por amostra	16
Duração Média de Gravação	21.5 s
Número de Participantes	126 sendo 77 adultos e 49 ciranças
Gênero	79 masculino e 46 feminino (NA:1)
Idade (desvio médio padrão)	43.0 +- 32.2 anos (NA:1)
Idade dos Participantes adultos	67.6 +- 11.6 anos (NA:1)
Idade das Crianças	4.8 +- 4.6 anos
Indice de Massa Corporal dos Adultos	27.2 +- 5.4 kg m^{-2} anos (NA:2)
Peso das Crianças	21.4 +- 17.2 kg (NA:5)
Altura das Crianças	104.7 +- 30.8 cm (NA:7)

Fonte: Rocha et. al., 2019.

2.2 – Exploração de Dados

Visando iniciar os estudos, foi realizada uma exploração dos dados contidos na base de dados do ICHBI 2017. Este banco de dados possui sons respiratórios normais e adventícios, sendo estes últimos divididos em sibilos, estalos ou uma combinação de ambos.

Foi estudada então uma rotina em *python* presente em Kaggle (2019), apresentada na figura 1, que gera um plot. Plots são formas de categorizar os resultados obtidos ao compilar o código. Essa categorização tem o intuito de facilitar futuras análises desses resultados por apresentá-los de uma maneira alternativa, na forma de gráficos ou outras representações visuais.

No código apresentado na figura 2 pode-se perceber que há um acesso à base de dados de áudios e uma verificação das informações de duração de cada áudio e, em seguida, é implementada a plotagem de um histograma utilizando a função *plt.hist*.

A figura 2 ilustra o plot gerado pelo trecho de código supracitado, que tem o objetivo de dividir o conteúdo da base de dados por tipos de sons respiratórios apresentando em um gráfico de barras que mostra a distribuição de durações dos áudios existentes na base de dados utilizada com a seguinte organização: O eixo vertical demonstra a quantidade de áudios com determinada duração e o eixo horizontal demonstra as durações de áudio presentes na base de dados.

Figura 1 – Trecho do código que possui a rotina

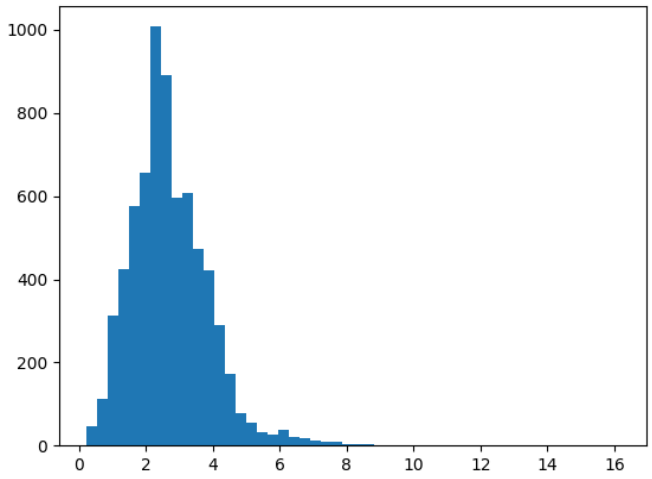
```
duration_list = []
for i in range(len(rec_annotations)):
    current = rec_annotations[i]
    duration = current['End'] - current['Start']
    duration_list.extend(duration)

duration_list = np.array(duration_list)
plt.hist(duration_list, bins = 50)
print('longest cycle:{}'.format(max(duration_list)))
print('shortest cycle:{}'.format(min(duration_list)))
threshold = 5
print('Fraction of samples less than {} seconds:{}'.format(threshold,
                                                           np.sum(duration_
list < threshold)/len(duration_list)))
```

```
longest cycle:16.163
shortest cycle:0.20000000000000284
Fraction of samples less than 5 seconds:0.9660771238040011
```

Fonte: Kaggle, 2019.

Figura 2 – Plot gerado após exploração da base de dados ICHBI



Fonte: Kaggle, 2019.

2.3 – Reamostragem

Pode-se definir os sons como vibrações que se propagam pelo ar transmitindo energia. Os arquivos de áudio são geralmente armazenados no formato .wav e precisam ser digitalizados, usando o conceito de amostragem. A frequência de amostragem (ou taxa de amostragem) é o número de amostras por segundo em um som (BAHETI, 2020).

Na base de dados utilizada nesse estudo, os sons respiratórios têm diferentes taxas de amostragem. Com isso, como uma etapa fundamental de pré-processamento, foi realizada uma reamostragem de cada áudio de som pulmonar para 4000 Hz. No trecho de código apresentado na figura 3, os ciclos de áudio são armazenados em pastas que correspondem a sua categoria e neste processo ocorre a reamostragem, estabelecendo o valor de 4000 Hz com a variável *rate*, utilizando a função *write* da biblioteca *Wave*.

Figura 3 – Trecho de código referente a reamostragem

```
def extract_all_training_samples(filenames, annotation_dict, root, target_rate, desired_length):
    cycle_list = []
    rate = 4000
    target_rate = 4000

    for file in filenames:
        data = get_sound_samples(annotation_dict[file], file, root, target_rate)
        cycles_with_labels = [(d[0], d[3], d[4]) for d in data[1:]]
        cycle_list.extend(cycles_with_labels)
        count = 0
        for s in cycles_with_labels:
            if ((s[1] == 0) & (s[2] == 0)):
                write('C:/Users/cesar/PycharmProjects/Artigo/WAV/Normal/'+ file + str(count) + '.wav', rate, s[0])
                count = count +1
            elif ((s[1] == 1) & (s[2] == 0)):
                write('C:/Users/cesar/PycharmProjects/Artigo/WAV/Crackles/' + file + str(count) + '.wav', rate, s[0])
                count = count +1
            elif ((s[1] == 0) & (s[2] == 1)):
                write('C:/Users/cesar/PycharmProjects/Artigo/WAV/Wheezes/' + file + str(count) + '.wav', rate, s[0])
                count = count +1
            elif ((s[1] == 1) & (s[2] == 1)):
                write('C:/Users/cesar/PycharmProjects/Artigo/WAV/Both/' + file + str(count) + '.wav', rate, s[0])
                count = count +1
```

Fonte: Autoria própria.

2.4 – Normalização

A normalização é uma técnica utilizada para ajustar o volume dos arquivos de som a um nível padrão, pois muitas vezes esses áudios de uma base de dados podem ter variações em seus níveis com referência ao volume do som. Se não for aplicada essa técnica o nível pode variar muito de palavra para palavra e o classificador pode processar de maneira inadequada.

Neste estudo foi utilizada uma função retirada da plataforma *Kaggle* que realiza cálculos para determinar qual o intervalo de valores que será atribuído às frequências dos arquivos de áudio que serão utilizados para treinamento do classificador que ao final da compilação do código irá gerar os espectrogramas de mel, conforme apresenta a Figura 4.

III. RESULTADOS

Os métodos estudados neste artigo geralmente são utilizados como base para geração de um espectrograma em uma determinada arquitetura. Espectrogramas são representações visuais que possuem a capacidade de representar tempo, frequência e amplitude em um único gráfico. Este espectrograma vai representar muitas vezes a entrada do classificador em projetos de classificação de sons autônomos. Utilizando esta mesma técnica, mas adicionando os elementos de cálculo da escala de Mel, que é uma escala perceptual que mede tons considerados iguais em distância um do outro, existe o espectrograma de Mel.

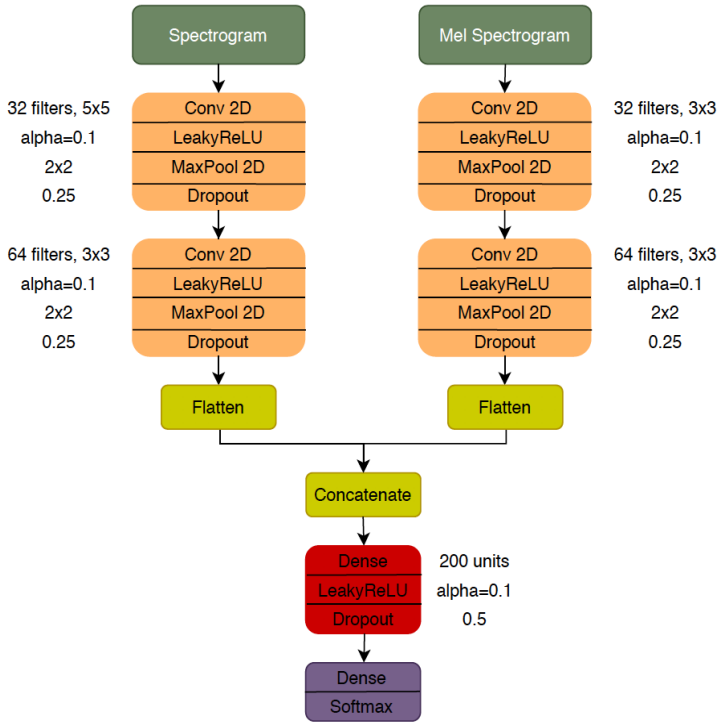
No trabalho de Rocha (2021) tem-se, por exemplo, uma arquitetura com dupla entrada, isto é, composta de espectrograma e mel espectrograma, conforme mostra a Figura 5.

Figura 4 – Rotina responsável pela normalização.

```
def sample2MelSpectrum(cycle_info, sample_rate, n_filters, vtlp_params):
    n_rows = 175  # 7500 cutoff
    n_window = 512  # ~25 ms window
    (f, t, Sxx) = scipy.signal.spectrogram(cycle_info[0], fs=sample_rate, nfft=n_window, nperseg=n_window)
    Sxx = Sxx[:n_rows, :].astype(np.float32)  # sift out coefficients above 7500hz, Sxx has 196 columns
    mel_log = FFT2MelSpectrogram(f[:n_rows], Sxx, sample_rate, n_filters, vtlp_params)[1]
    mel_min = np.min(mel_log)
    mel_max = np.max(mel_log)
    diff = mel_max - mel_min
    norm_mel_log = (mel_log - mel_min) / diff if (diff > 0) else np.zeros(shape=(n_filters, Sxx.shape[1]))
    if (diff == 0):
        print('Error: sample data is completely empty')
    labels = [cycle_info[1], cycle_info[2]]  # crackles, wheezes flags
    return (np.reshape(norm_mel_log, (n_filters, Sxx.shape[1], 1)).astype(np.float32),  # 196x64x1 matrix
            label2onehot(labels))
```

Fonte: Kaggle, 2019.

Figura 5 – Arquitetura CNN com dupla entrada



Fonte: Rocha, 2021.

3.1 – Geração do Espectrograma

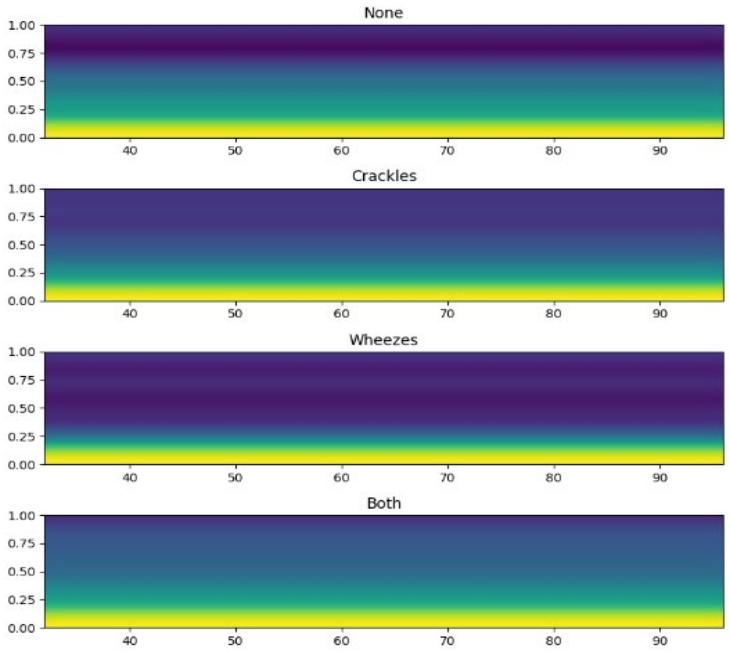
Ao fim da análise dos áudios e todo o processo de classificação, é possível gerar espectrogramas das classes as quais os ciclos respiratórios foram divididos. Este processo ocorre utilizando a função *plt.specgram* da biblioteca *matplotlib.pyplot*, conforme trecho de código apresentado na figura 6, assim como na figura 7 é possível observar os espectrogramas gerados.

Figura 6 - Plotagem de espectrogramas

```
plt.figure(figsize=(8, 8))
plt.subplot(4, 1, 1)
plt.specgram(training_clips['none'][ind][0].reshape(sample_height, sample_width))
plt.title('None')
plt.subplot(4, 1, 2)
plt.specgram(training_clips['crackles'][ind][0].reshape(sample_height, sample_width))
plt.title('Crackles')
plt.subplot(4, 1, 3)
plt.specgram(training_clips['wheezes'][ind][0].reshape(sample_height, sample_width))
plt.title('Wheezes')
plt.subplot(4, 1, 4)
plt.specgram(training_clips['both'][ind][0].reshape(sample_height, sample_width))
plt.title('Both')
plt.tight_layout()
plt.colorbar(format='%+2.0f dB')
```

Fonte: Autoria própria

Figura 7 - Espectrogramas gerados pela análise dos áudios



Fonte: Autoria própria

3.2 – Geração do Mel Espectrograma

Após os mesmos processos citados no item 3.1, também é possível gerar espectrogramas de mel, utilizando as funções *plt.imshow, plt.subplot* da biblioteca *matplotlib.pyplot* apresentadas na Figura 8 e obtendo os resultados de plotagem demonstrados na Figura 9.

Figura 8 - Trecho de código para plotagem de espectrograma de mel.

```
target_sample_rate = 22000
sample_length_seconds = 5
sample_dict = extract_all_training_samples(filenames, rec_annotations_dict, root, target_sample_rate, sample_length_seconds)
training_clips = sample_dict
sample_height = training_clips['none'][0][0].shape[0]
sample_width = training_clips['none'][0][0].shape[1]
ind = 1
plt.figure(figsize=(10, 10))
plt.subplot(4, 1, 1)
plt.imshow(training_clips['none'][ind][0].reshape(sample_height, sample_width))
plt.title('None')
plt.subplot(4, 1, 2)
plt.imshow(training_clips['crackles'][ind][0].reshape(sample_height, sample_width))
plt.title('Crackles')
plt.subplot(4, 1, 3)
plt.imshow(training_clips['wheezes'][ind][0].reshape(sample_height, sample_width))
plt.title('Wheezes')
plt.subplot(4, 1, 4)
plt.imshow(training_clips['both'][ind][0].reshape(sample_height, sample_width))
plt.title('Both')
plt.tight_layout()
plt.colorbar(format='%+2.0f dB')
```

Fonte: Kaggle, 2019.

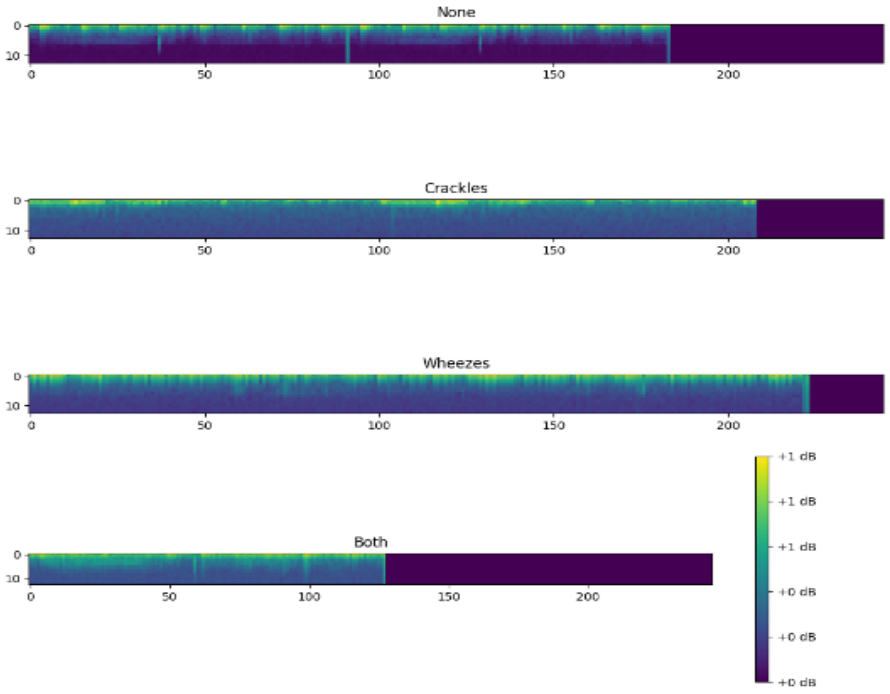
IV. CONCLUSÕES

Este estudo teve como objetivo principal apresentar as principais técnicas utilizadas no pré-processamento de sons respiratórios gerando espectrogramas muito utilizados como entrada em arquiteturas de classificadores IA.

O trabalho limitou-se a apresentar aspectos teóricos e práticos com implementação de códigos utilizando linguagem de programação python, uma das mais utilizadas na área de IA. Inicialmente foi apresentada a base de dados, que representa uma coleção robusta contendo amostras normais e anormais de sons respiratórios e muito utilizada nesse ramo de pesquisa. Posteriormente se realizou uma exploração desses dados e apresentadas as técnicas de reamostragem e normalização até a geração do espectrograma e mel espectrograma.

Com isso, esse estudo representa um guia para pesquisadores que estejam inseridos nesse ramo de classificação autônoma de sons sejam eles respiratórios ou não, uma vez que tais técnicas são largamente utilizadas na classificação de sons em geral.

Figura 9 – Espectrogramas de Mel gerados pela análise dos áudios.



Fonte: Kaggle, 2019.

V. REFERENCES

BAHETI, Pragati. **Working with Audio Data for Machine Learning in Python**. HeartBeat, 2020. Disponível em: <https://heartbeat.fritz.ai/working-with-audio-signals-in-python-6c2bd63b2daf> . Acesso em 20 jul 2021.

GRZYWALSKI, Tomasz; PIECUCH, Mateusz; SZAJEK, Marcin; BRĘBOROWICZ, Anna; HAFKE-DYS, Honorata; KOCIŃSKI, Jędrzej; PASTUSIAK, Anna; BELLUZZO, Riccardo. Practical implementation of artificial intelligence algorithms in pulmonary auscultation examination. **European Journal of Pediatrics** (2019) 178:883–890 https://doi.org/10.1007/s00431-019-03363-2

KAGGLE. **Respiratory Diseases**. 2019. Disponível em: <https://www.kaggle.com/sakshamrawal/respiratory-diseases/data>. Acessado em 10 ago 2021.

MARQUES, Alda; OLIVEIRA, Ana; JÁCOME, Cristina. Computerized Adventitious Respiratory Sounds as Outcome Measures for Respiratory Therapy: A Systematic Review. **Respitarory Care**, 2014, v. 59 v. 5. DOI: 10.4187/respcare.02765

ROCHA, Bruno Machado; FILOS, Dimitris; MENDES, Luís; SERBES, Gorkem; ULUKAYA, Sezer; KAHYA, Yasemin P; JAKOVLJEVIC, Niksa; TURUKALO, Tatjana L; VOGIATZIS, Ioannis M; PERANNTONI, Eleni. An open access database for the evaluation of respiratory sound classification algorithms. **Institute of Physics and Engineering in Medicine**, v. 40, n. 3, 2019.

ROCHA, Bruno Machado; PESSOA, Diogo; MARQUES, Alda; CARVALHO, Paulo; PAIVA, Rui Pedro. Automatic Classification of Adventitious Respiratory Sounds: A (Un)Solved Problem? **Sensors** 2021, 21, 57. https://dx.doi.org/10.3390/s21010057

VI. AGRADECIMENTOS

Este artigo é o resultado do projeto de PD&I IA_Machine Learning for Pneumonia Diagnose, realizado pela instituição Universidade do Estado do Amazonas, em parceria com a Samsung Eletrônica da Amazônia Ltda., usando recursos da Lei Federal nº 8.387/1991, estando sua divulgação e publicidade em conformidade com o previsto no artigo 39.º do Decreto nº 10.521/2020.

Os autores agradecem a Universidade do Estado do Amazonas – Escola Superior de Tecnologia, a Agência de Inovação da UEA - AGIN e a Samsung Eletrônica da Amazônia Ltda. pelo apoio.

VII. COPYRIGHT