Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Desenvolvimento de um biomarcador vocal

Análise de áudio para a estimativa do nível de SpO_2 usando Redes Neurais

Natália Hitomi Koza e Ricardo Mikio Morita

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Prof. Dr. Marcelo Finger

São Paulo
2022

"Whatever it is you're seeking won't come in the form you're expecting." — Haruki Murakami

Agradecimentos

It's still magic even if you know how it's done.
— Terry Pratchett - "A Hat Full of Sky"

Dedico este trabalho à minha família, que esteve comigo ao longo de toda esta jornada. Não sei até onde ela me levará, mas estou feliz de ter tido sua companhia comigo desde o início até o fim. Agradeço também a ajuda da Professora Nina, do Professor Marcelo Finger ao longo deste ano para a elaboração de nosso trabalho. Este não seria possível sem o apoio contínuo de vocês.— Ricardo Mikio Morita

Agradeço aos meus pais, ao Nathan, aos meus amigos Ricardo G., Luiz Gabriel e Amauri, que me apoiaram emocionalmente ao longo dos anos, aos professores Nelson Uehara e Guilherme França que me motivaram e inspiraram a seguir esse caminho e, ao professor Marcelo Finger pela oportunidade e supervisão na realização deste projeto. — Natália Hitomi Koza

Resumo

Natália Hitomi Koza e Ricardo Mikio Morita. **Desenvolvimento de um biomarcador vocal: *Análise de áudio para a estimativa do nível de SpO_2 usando Redes Neurais***. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2022.

Insuficiência respiratória é um sintoma comum em casos de COVID-19 e pode ser diagnosticada quando a saturação de oxigênio do sangue (SpO_2) de um paciente é inferior a 92%. Nosso trabalho analisa a eficiência de Redes Neurais Convolucionais para determinar o nível de SpO_2 a partir de amostras de voz. Através de etapas de pré-processamento, estas amostras de áudio passam por um processo de janelamento para serem então convertidas em MFCCs, as quais foram usadas em diversos experimentos para tentar determinar uma configuração ótima da Rede Neural. A investigação incluiu a variação de hiper parâmetros da rede, adição de ruído, comparação do desempenho de MFCCs e Mel-espectrogramas, dentre outros. As técnicas aplicadas nos permitiram alcançar uma acurácia em valor absoluto de estimativas próxima da estimativa real com variação de dois pontos. Também foi feita uma avaliação do desempenho clínico e notou-se que o modelo possui uma acurácia de 80% na predição de pacientes com insuficiência, apesar de que em certos intervalos nosso modelo é capaz de predizer corretamente com uma acurácia ao redor de 90%, comparável ao trabalho anterior do grupo SPIRA. Acreditamos que é necessário obter mais amostras de pacientes com insuficiência respiratória para que possamos analisar com maior profundidade estes resultados e possivelmente melhorar as nossas métricas.

Palavras-chave: Biomarcador. Insuficiência Respiratória. Redes Neurais Convolucionais.

Abstract

Natália Hitomi Koza e Ricardo Mikio Morita. **Development of a voice biomarker: *Audio analysis for estimating SpO_2 through the use of Neural Networks***. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2022.

Respiratory failure is a common symptom in cases of COVID-19 and can be diagnosed when a patient's blood oxygen saturation (SpO_2) is less than 92%. Our work analyzes the efficiency of Convolutional Neural Networks to determine the SpO_2 level from voice samples. Through pre-processing steps, these audio samples go through a windowing process to be then converted into MFCCs, which were used in several experiments to try to determine an optimal configuration of the Neural Network. The investigation included changing hyper parameters of the network, adding noise, comparing the performance of MFCCs and Mel-spectrograms, among others. The applied techniques allowed us to reach an accuracy in absolute value of estimates close to the actual value within a variation of two points. An evaluation of the clinical performance was also carried out and it was noted that the model has an accuracy of 80% in predicting patients with respiratory failure, although in certain intervals our model is able to predict correctly with an accuracy of around 90% , comparable to the previous work by the SPIRA project. We believe that it is necessary to obtain more samples from patients with respiratory failure so that we can analyze these results more seriously and also possibly improve our metrics.

Keywords: Biomarker. Respiratory Failure. Convolutional Neural Networks.

Lista de Abreviaturas

IME	Instituto de Matemática e Estatística
USP	Universidade de São Paulo
SUS	Sistema Único de Saúde
DFT	Transformada Discreta de Fourier (*Discrete Fourier Transform*)
MFC	*Mel-Frequency Cepstrum*
MFCC	*Mel-Frequency Cepstrum Coefficients*
STFT	Transformada de Fourier de tempo reduzido (*Short-Time Fourier Transform*)
DCT	*Discrete Cosine Transform*
CNN	Rede Neural Convolucional (*Convolutional Neural Network*)
COVID-19	Doença do Coronavírus (*Coronavirus disease*)

Lista de Símbolos

SpO_2	Saturação da pressão de oxigênio no sangue
$\hat{S}pO_2$	Estimativa ou predição da SpO_2
$Tanh$	Tangente hiperbólica
$ReLU$	*Rectified Linear Unit*

Sumário

Introdução **1**

Motivação . . . 1

Objetivos . . . 1

1 Considerações Preliminares **3**

1.1 Biomarcadores Vocais . . . 3

1.2 Análise no domínio do Tempo versus Análise no domínio da Frequência 4

1.3 Short-Time Fourier Transform (STFT) . . . 5

1.4 Espectrogramas . . . 7

1.5 Escala mel . . . 8

1.6 Mel-Frequency Cepstral Coefficients (MFCCs) . . . 10

1.7 Redes Neurais . . . 12

1.7.1 Perceptrons . . . 12

1.7.2 Neurônios Sigmoides . . . 14

1.7.3 Retropropagação . . . 15

1.7.4 Redes Neurais Convolucionais . . . 16

1.7.5 Avaliação de Desempenho . . . 18

2 Metodologia **23**

2.1 Dados . . . 24

2.1.1 Janelamento . . . 25

2.1.2 Formato do arquivo de entrada . . . 26

2.1.3 Processamento do áudio . . . 26

2.1.4 *Split* dos dados . . . 26

2.2 Rede Neural . . . 27

2.2.1 Hiper-parâmetros . . . 28

3 Análise dos resultados **29**

3.1 Regressão . . . 29

3.1.1 Resultados 29
3.1.2 Performance 41
3.2 Classificação 42
3.2.1 Resultados 43
3.3 Análise do Erro Relativo 44

4 Considerações finais **47**
4.1 Desafios enfrentados 48
4.2 Próximos passos 49

Referências **51**

Introdução

Motivação

Com a pandemia da doença respiratória COVID-19, que vem ocorrendo desde 2020 e continua vigente em 2022, vimos o número de óbitos e internações crescer rapidamente, inclusive levando a recomendação de procura dos hospitais somente em caso de sintomas graves devido à falta de leitos hospitalares, decorrente do alto número de pacientes com insuficiência respiratória que necessitavam de ventiladores e respiradores mecânicos.

Podemos citar como seus sintomas a tosse seca, dores de garganta, voz excessivamente aerada, anomalias no padrão de respiração e também a chamada *hipóxia silenciosa*, na qual acontece a queda da saturação de oxigênio no sangue (SpO_2) sem dispneia, ou seja, sem que o indivíduo apresente dificuldades de respiração. Esta representa um ponto de risco para a COVID-19 (Saúde - SUS, 2020) que pode levar ao diagnóstico tardio e a uma piora acelerada do quadro clínico do paciente.

Muitos estudos surgiram nesta área, tanto para o desenvolvimento de respiradores de baixo custo e autotestes, como também pesquisas que buscavam identificar a COVID-19 em seus estados iniciais através de sons respiratórios como tosse, respiração e voz.

Um desses estudos é o realizado pelo grupo SPIRA[1], que estimulado por este quadro de hipóxia silenciosa (Casanova *et al.*, 2021), propôs a criação de um biomarcador a partir da voz que apresentou bons resultados acerca da possibilidade de se identificar pacientes que estão abaixo do limiar de 92% de SpO_2 com uma acurácia de 91.66% e perspectivas de melhorar este número.

Nesta mesma linha, propomos no presente trabalho a criação de um biomarcador para estimar os níveis de SpO_2 de forma não invasiva através do uso da voz, com baixo custo e que possa auxiliar no diagnóstico nos casos de COVID-19. Na conclusão, questionamos a viabilidade desta abordagem para outras doenças do sistema respiratório.

Objetivos

Nosso trabalho visa estimar o nível de SpO_2 de um indivíduo através da análise da gravação de um pequeno trecho de fala acompanhado de um intervalo de confiança, de

[1] Grupo formado por pesquisadores da Universidade de São Paulo, criado em meio à pandemia com foco na criação de um sistema capaz de fazer triagem de pacientes com insuficiência respiratória motivado pela COVID-19, utilizando-se de modelos de aprendizado de máquina e processamento de áudio.

forma a analisar seu desempenho como biomarcador.

Fazendo uso de redes neurais e de dispositivos acessíveis e com baixo custo como *smartphones* e computadores para a coleta do áudio, tem-se a esperança de que esta ferramenta possa auxiliar na triagem hospitalar ou mesmo o público em geral, na avaliação da necessidade de intervenção médica acerca da presença de insuficiência respiratória.

Para isto, iremos trabalhar com os dados coletados do projeto SPIRA, que consistem em áudios de voz e planilhas de dados com informações adicionais, como os batimentos cardíacos e SpO_2. Iremos tratar e processar as gravações de vozes usando técnicas de processamentos de sinais e, com redes neurais, iremos estimar numericamente o nível de $Sp\mathit{O}_2$.

Finalmente, para relacionar nosso trabalho com sua principal aplicação clínica, iremos metrificar o quão bem a nossa rede detecta casos de insuficiência respiratória, usando o limiar de 92% de SpO_2.

Na conclusão resumiremos nossos achados e opiniões sobre a experiência do projeto de modo geral.

Capítulo 1

Considerações Preliminares

Este capítulo visa apresentar os princípios envolvidos no projeto, o que inclui explicações sobre os termos médicos presentes, pré-processamento dos áudios, noções de redes neurais e as métricas para avaliar o desempenho da rede.

Vamos também exemplificar com código a obtenção e visualização de alguns dos conceitos, usando a linguagem de programação *Python* e as bibliotecas *Librosa* para o processamento do áudio e *Matplotlib* para os gráficos.

Os detalhes de implementação da rede serão cobertos no capítulo 2.

1.1 Biomarcadores Vocais

Biomarcadores são indicações objetivas de características biológicas em um dado momento e são comumente usados na área médica. Eles podem ser divididos em categorias de acordo com seu método de obtenção e como exemplos podemos citar a taxa de glicose sanguínea, que é um biomarcador molecular, e as imagens de raio-X e tomografias computadorizadas, que são biomarcadores radiográficos. Algumas de suas aplicações se dão na monitoração de processos biológicos naturais, na determinação de um estado patogênico ou no grau de resposta a um tratamento farmacológico.

A voz pode ser utilizada como um biomarcador (se, assim como biomarcadores tradicionais, for validada analiticamente e qualificada com evidências) para auxiliar no diagnóstico de doenças que afetam o coração, cérebro, pulmões, músculos, pregas vocais e em outros elementos relacionados que possam causar alterações na fala.

Um biomarcador vocal é um conjunto de *features* vocais, que podem ser acústicas, como o *pitch* (tom), ou linguísticas, que caracterizam algum quadro clínico. Alguns exemplos de biomarcadores vocais que estão sendo investigados atualmente são para detecção de depressão (Mundt *et al.*, 2012), Parkinson (Tracy *et al.*, 2020) e mais recentemente COVID-19 (Despotovic *et al.*, 2021).

Usualmente são desenvolvidos através do treinamento de modelos de inteligência artificial usando processamento de sinais de áudio, onde os sinais passam por um pré-processamento para remoção de ruídos (que podem enviesar o modelo) e extração de

features, que no caso de Redes Neurais Convolucionais são normalmente Mel espectrogramas ou MFCCs.

1.2 Análise no domínio do Tempo versus Análise no domínio da Frequência

A análise de áudio no domínio do tempo representa a amplitude de um sinal em relação ao tempo. Uma forma de visualizar sinais de áudio nesse domínio é através de um osciloscópio. Arquivos *.wav*, por exemplo, utilizam essa forma de representação.

Em contraste, a análise de áudio no domínio da frequência representa a amplitude do sinal de áudio em relação a frequência, ou seja, neste domínio a onda de áudio é uma composição de sinais de diferentes frequências. Comumente se utiliza um analisador de espectro como ferramenta para essa visualização.

É importante distinguir estas duas formas de representação pois trabalhar com áudios na dimensão do tempo, mesmo se fixado o tamanho de janelas no áudio para manter a duração constante, tende a ter uma dimensionalidade muito alta, dificultando a aplicação de certas técnicas para o estudo destas dimensões na área de *Information Retrieval*, e também faz com que sons perceptivamente similares possuam sua representação no vetor espaço — isto é, a representação espacial do som, no caso dispondo as amplitudes em função do tempo — com valores de similaridades inadequados para comparação (Stowell e Plumbley, 2014), levando à possível perda de informações relevantes.

Logo, a representação de áudio no domínio da frequência é a mais adequada para o nosso problema e vamos abordá-la com mais detalhes nas próximas seções.

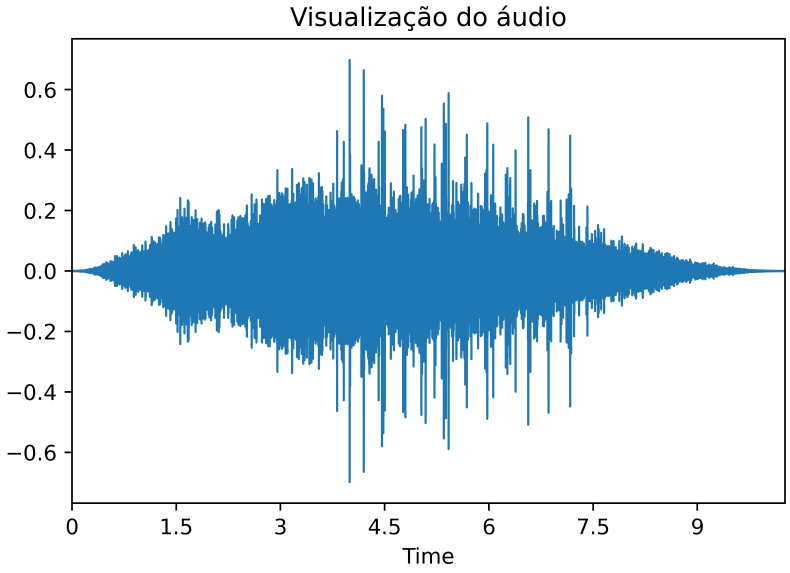
O Programa 1.1 exemplifica como carregar um arquivo de áudio[1] e visualizar o formato de sua onda, na representação amplitude por domínio do tempo.

Notar que os trechos de código deste capítulo, apesar de serem apresentados como diferentes programas, seguem uma sequência lógica e podem depender entre si. Então, recomendamos ao interessado em replicar o código que interprete-o como um único arquivo.

Programa 1.1 Carregando e visualizando um arquivo de áudio

```
1   import librosa
2   import librosa.display
3
4   y, sr = librosa.load('mixkit-stadium-crowd-light-applause-362.wav')
5   crowd_sound, _ = librosa.effects.trim(y)
6   fig1 = plt.figure(dpi = 900)
7   librosa.display.waveplot(crowd_sound, sr=sr);
8   plt.title("Visualização do áudio")
9   plt.show()
10  fig1.savefig("grafico1.png")
```

[1] Este áudio é *royalty-free* e está disponível no site https://mixkit.co/free-sound-effects/



A imagem obtida é uma representação bidimensional do áudio com suas amplitudes de onda ao longo do tempo. O exemplo se trata do som de uma plateia aplaudindo.

1.3 Short-Time Fourier Transform (STFT)

A conversão de áudio entre o domínio do tempo e o domínio da frequência é feita através de operações denominadas transformadas, dentre as quais a transformada de Fourier é uma das mais populares.

Estas operações são funções, também conhecidas como decomposições, que traduzem a amplitude de um áudio ao longo do tempo em uma composição de diferentes frequências.

Para fins computacionais, estamos interessados na transformada discreta de Fourier (DFT). Conceitualmente, esta transforma uma sequência de N números complexos $\{x_n\} := x_0, x_1, ..., x_{N-1}$ em outra sequência de números complexos, $\{X_k\} : X_0, X_1, ..., X_{N-1}$, a qual é definida por:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N} kn}$$

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \left[\cos\left(\frac{2\pi}{N} kn\right) - i \cdot \sin\left(\frac{2\pi}{N} kn\right) \right]$$

Esta operação requer $O(N^2)$ operações: temos N termos X_k e cada operação requer uma soma de N termos. Com o algoritmo *Fast Fourier Transform* (FFT), entretanto, é possível reduzir o custo desta transformada para $O(N \log N)$. Este algoritmo é extremamente

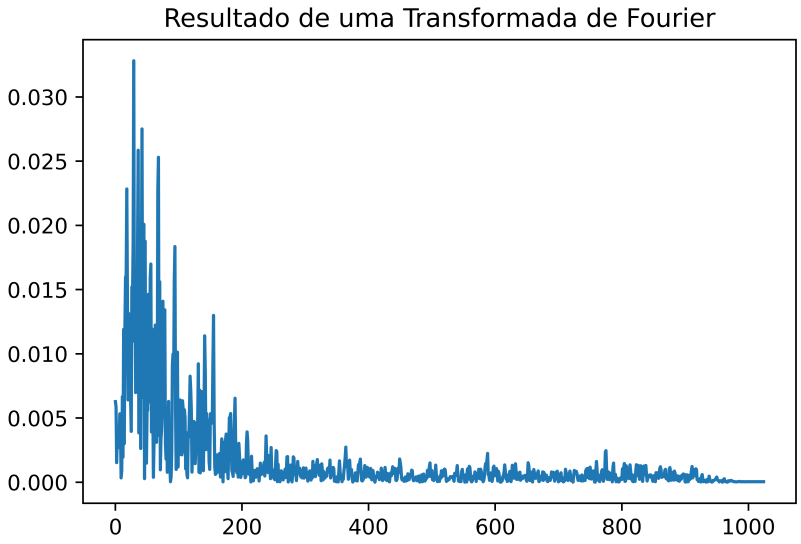
importante em diversas áreas como tratamento de imagens, processamento de sinais, multiplicação de polinomiais, cálculo de convoluções entre outras.

O Programa 1.2 mostra a aplicação da Transformada de Fourier em uma pequena amostra do áudio, onde a convertemos do domínio do tempo para o domínio da frequência.

Programa 1.2 Aplicando Transformada de Fourier a uma pequena janela do áudio

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3
4   n_fft = 2048
5   fig2 = plt.figure(dpi = 900)
6   D = np.abs(librosa.stft(crowd_sound[:n_fft], n_fft=n_fft, hop_length=n_fft+1))

7   plt.plot(D)
8   plt.title("Resultado de uma Transformada de Fourier")
9   plt.show()
10  fig2.savefig("grafico2.png")
```



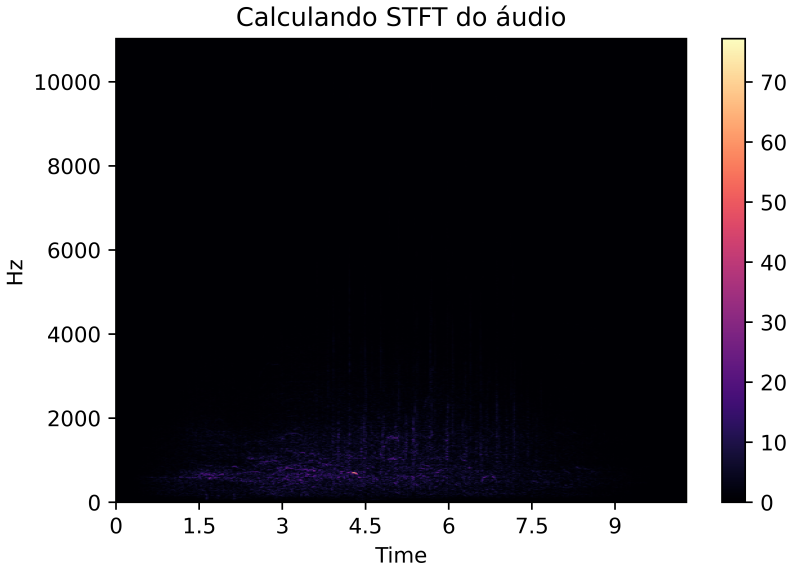
A imagem obtida ilustra o que acontece à representação 2D — amplitude *vs* tempo — de um áudio quando aplicamos a Transformada de Fourier em um pequeno intervalo do áudio. Nota-se uma prevalência de ondas de frequência inferior a 200 Hz.

Agora, se dividirmos o áudio em janelas de mesmo tamanho (com sobreposições, caso conveniente), calcularmos a DFT de cada uma delas e somarmos os resultados numa matriz, teremos um registro da magnitude e fase de cada ponto no tempo e frequência. Denominamos este processo como *Short-Time Fourier Transform (STFT)* e para dados discretos de tempo chamamos de *Discrete-Time STFT*.

O Programa 1.3 mostra como obter uma STFT, a qual internamente é calculada se dividirmos o áudio em intervalos e aplicarmos DFTs em cada um destes.

Programa 1.3 Obtendo STFT

```
hop_length = 512
D = np.abs(librosa.stft(crowd_sound, n_fft=n_fft, hop_length=hop_length))
fig3 = plt.figure(dpi = 900)
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='linear');
plt.colorbar();
plt.title("Calculando STFT do áudio")
plt.show()
fig3.savefig("grafico3.png")
```



Esta representação obtida ainda é muito sutil e de difícil interpretação, pois a maioria dos sons audíveis por humanos se concentram num pequeno intervalo de frequência. A visualização pode ser melhorada através da mudança de escala da imagem, do uso de espectrograma, entre outros métodos.

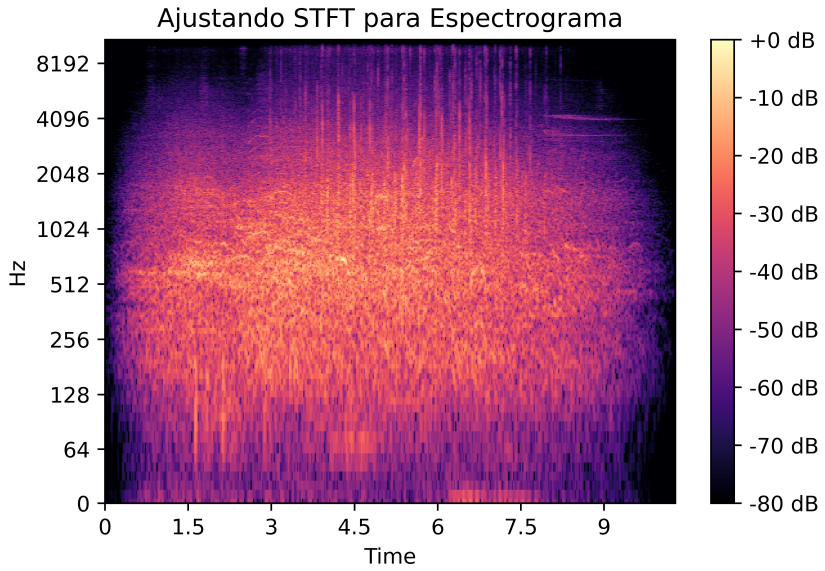
1.4 Espectrogramas

Um espectrograma é uma composição dos dois domínios (tempo e frequência), onde é possível visualizar, no nosso caso em que tratamos de som, a variação da intensidade de frequências de um áudio ao longo do tempo, embora também seja possível usá-lo para visualizar frequências de ondas eletromagnéticas, como a luz, ao longo do tempo. Apesar de poder ser visto como um gráfico de superfície, usualmente é apresentado de forma planar, como um mapa de calor, onde as cores representam a intensidade da frequência (Network, 2016).

Para melhorar a representação obtida pelo programa 1.3, podemos converter as frequências da STFT no eixo y para uma escala logarítmica e colorir o eixo da amplitude com o esquema de decibéis, como pode ser visto no Programa 1.4. Esta representação, não coincidentemente, é denominada de espectrograma.

Programa 1.4 Usando STFT para encontrar o Espectrograma

```
1   fig4 = plt.figure(dpi = 900)
2   DB = librosa.amplitude_to_db(D, ref=np.max)
3   librosa.display.specshow(DB, sr=sr, hop_length=hop_length, x_axis='time',
        y_axis='log');
4   plt.colorbar(format='%+2.0f dB');
5   plt.title("Ajustando STFT para Espectrograma")
6   plt.show()
7   fig4.savefig("grafico4.png")
```



Com este ajuste, podemos observar mais facilmente variações na imagem. Nem sempre um ser humano pode ser capaz de distinguir dois espectrogramas de áudios distintos, mas computacionalmente o resultado obtido é imediatamente superior do que a simples utilização de *STFTs* para a extração de *features* em Redes Neurais Convolucionais.

Através de uma transformação não-linear, podemos converter a frequência de um espectrograma para a escala mel, obtendo um *Mel-Frequency Cepstrum* (MFC) ou Mel espectrograma.

1.5 Escala mel

Mel é uma escala que visa representar frequências sonoras baseando-se na percepção humana de tons. Na escala usual de frequências, tons consecutivos tem distâncias diferentes

em relação à sua frequência, mas na escala mel tons consecutivos estão posicionados à mesma distância. A escala se baseia no fato de que o ouvido humano percebe facilmente variações em baixas frequências, enquanto que tem dificuldades em discernir mudanças em altas frequências, sugerindo uma escala logarítmica de percepção (Smyth, 2019).

Foi definido como referência que 1000 mels = 1000 Hz e é possível, através de uma transformação não-linear, converter frequências f (em Hertz) para a escala mel, obtendo o valor m, em mels, com a fórmula:

$$m = 2595 \times \log_{10}\left(1 + \frac{f}{700}\right)$$

A figura 1.1 ilustra a relação entre a escala mel e a escala usual de frequências.

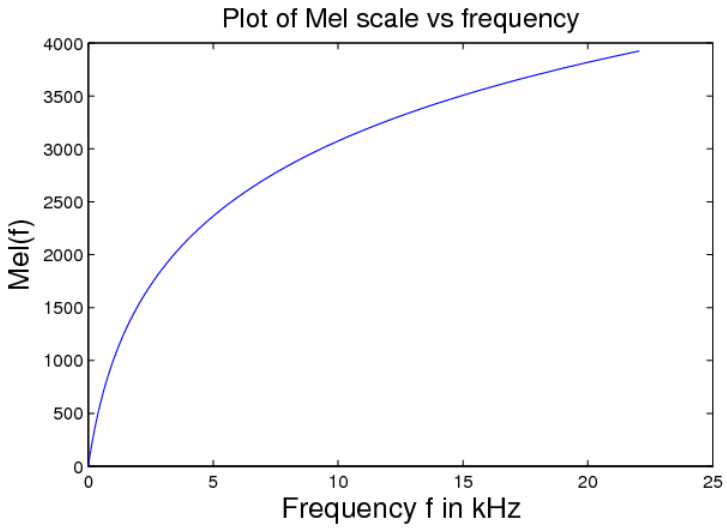


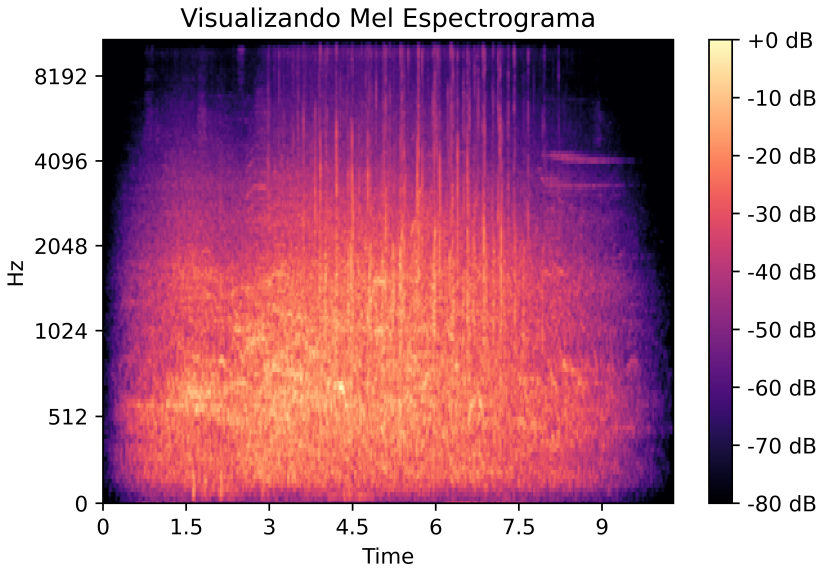
Figura 1.1: *Escala mel vs escala de frequência usual*

Fonte: Ramaseshan, 2013

Para nosso trabalho, estaremos mais interessados em Mel espectrogramas e MFCCs, do que em espectrogramas. O Programa 1.5 mostra como obter um Mel espectrograma.

Programa 1.5 Encontrando Mel Espectrograma

```
1   n_mels = 128
2   fig5 = plt.figure(dpi = 900)
3   S = librosa.feature.melspectrogram(crowd_sound, sr=sr, n_fft=n_fft,
        hop_length=hop_length, n_mels=n_mels)
4   S_DB = librosa.power_to_db(S, ref=np.max)
5   librosa.display.specshow(S_DB, sr=sr, hop_length=hop_length, x_axis='time',
        y_axis='mel');
6   plt.colorbar(format='%+2.0f dB');
7   plt.title("Visualizando Mel Espectrograma")
8   plt.show()
9   fig5.savefig("grafico5.png")
```



O resultado é uma imagem similar ao espectrograma, mas com as frequências convertidas para a escala mel. Com isso, podemos trabalhar com MFCCs.

1.6 Mel-Frequency Cepstral Coefficients (MFCCs)

MFCCs são coeficientes que coletivamente compõem uma MFC. De posse de um Mel Espectrograma, podemos encontrar estes MFCCs com uma transformação denominada *Discrete Cosine Transform (DCT)* seguida de uma normalização. Dados muito correlacionados (i.e. com alto grau de relação linear) tem melhor performance em MFCCs do que com Mel Espectrogramas.(Fayek, 2022)

Esquematicamente a geração das MFCCs se dá por (Gordillo, 2014):

- Pré-ênfase: passagem do sinal de áudio por um filtro que amplifica as altas frequências, as quais foram atenuadas pelo mecanismo de produção da voz.

- Segmentação: divisão do sinal de fala em trechos menores, geralmente com duração entre 20 e 40 ms.
- Janelamento: usualmente se utiliza o janelamento de Hamming para diminuir a amplitude da descontinuidade nos extremos do *frame*, o qual é feito multiplicando cada segmento do sinal pelas amplitudes da janela de Hamming com mesma duração do segmento. A janela de Hamming, mostrada na figura 1.2, possui um pico no centro e tende a zero nas extremidades.

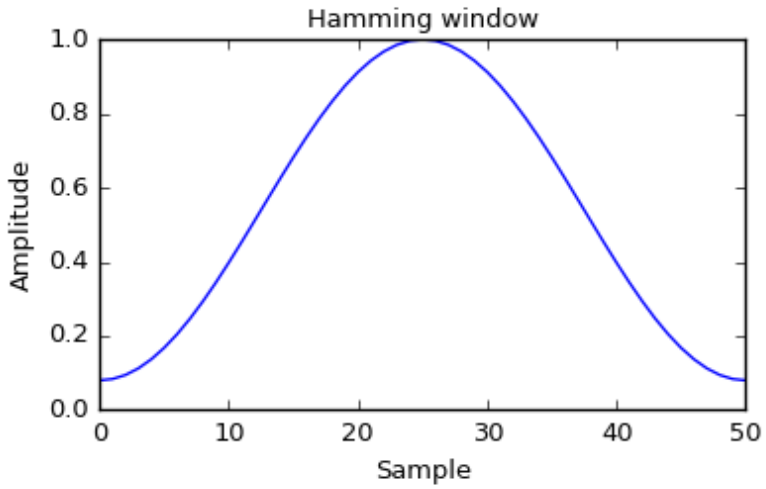


Figura 1.2: *Gráfico da Janela de Hamming com duração de 50* samples

- Cálculo da FFT em cada janela: conversão do sinal complexo no domínio do tempo em componentes mais simples no domínio da frequência.
- Passagem pelo *mel filterbank*: conversão para a escala mel com uso de filtros triangulares.
- Log: usado como função de suavização.
- DCT: conversão do espectro log Mel no domínio do tempo.
- *Lifter*: normalização dos coeficientes.

Para calcularmos MFCCs, temos a fórmula:

$$c_i = \sum_{n=1}^{N_f} Sn \cos\left[i(n-0.5)\left(\frac{\pi}{N_f}\right)\right], i = 1, 2, ..., L$$

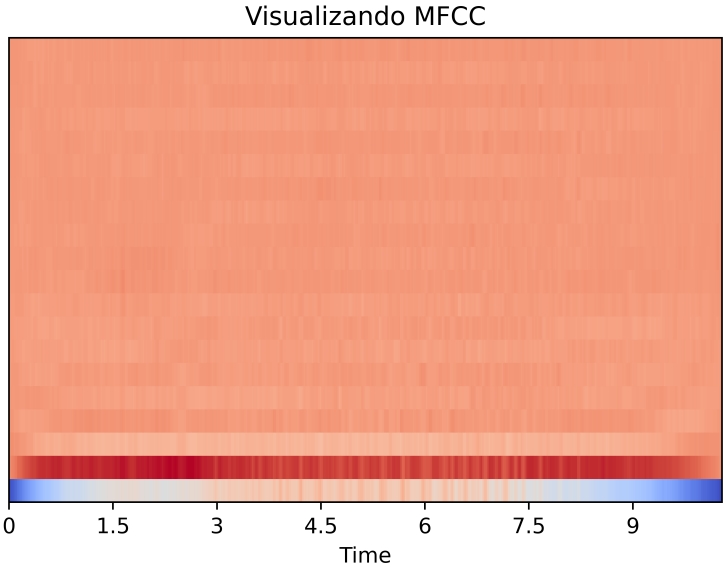
Onde $c_i = c_y(i)$ = i-ésimo coeficiente de MFCC, N_f é o número de filtros triangulares no banco de filtros[2], S_n é o log da energia de saída do n-ésimo coeficiente e L é o número de coeficientes de MFCC que queremos calcular.

[2] um arranjo de filtros passa-faixa que decompõe o sinal em diversas componentes, cada uma carregando apenas uma sub-banda de frequência do sinal original.

O Programa 1.6 exemplifica a obtenção de MFCCs.

Programa 1.6 Encontrando MFCCs

```
1   crowd_mfcc = librosa.feature.mfcc(crowd_sound, sr=sr, n_fft=n_fft, hop_length
        =hop_length, n_mels=n_mels)
2   fig6 = plt.figure(dpi = 900)
3   librosa.display.specshow(crowd_mfcc, sr=sr, x_axis='time')
4   plt.title("Visualizando MFCC")
5   plt.show()
6   fig6.savefig("grafico6.png")
```



É possível notar diferenças visuais claras entre MFCCs e Mel espectrogramas, mas não é evidente quais informações cada representação carrega. Assim, ambas as formas são comumente usadas como *features* em Redes Neurais Convolucionais para problemas envolvendo áudio. O nosso trabalho inclui uma comparação do desempenho entre elas quando utilizadas como entrada nos nossos modelos.

1.7 Redes Neurais

Esta seção busca dar uma breve introdução sobre os modelos de Redes Neurais, suas componentes e formas de se avaliar o desempenho de Redes.

1.7.1 Perceptrons

Perceptrons foram criados em meados de 1950 e 1960 pelo cientista Frank Rosenblatt (Rosenblatt, 1962), inspirado no trabalho de Warren McCulloch e Walter Pitts (McCulloch e Pitts, 1943). Inicialmente foram desenvolvidos para existir fisicamente como

máquinas, com avanços na área levando ao refinamento de sua lógica e possibilitando a implementação destes em programas.

Um perceptron recebe entradas binárias (i.e. apenas aceita 0 ou 1) $x_1, x_2, \ldots, x_n$ e produz uma saída binária (a qual também pode ser apenas 0 ou 1). A figura 1.3 ilustra um exemplo de perceptron com 3 entradas.

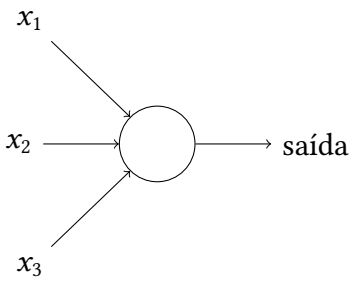


Figura 1.3: *Exemplo de um perceptron simplificado com três entradas: x_1, x_2 e x_3*

Rosenblatt propôs uma regra simples para calcular a saída: ele introduziu números reais como pesos $w_1, w_2, \ldots, w_n$ que expressam a importância de cada entrada. Para descobrir se a saída é 0 ou 1, é necessário calcular se $\sum_j w_j x_j$ é menor ou maior do que um determinado limiar (*threshold*) que também é um número real. Em termos algébricos, temos:

$$saída = \begin{cases} 0 & \text{se } \sum_j w_j x_j \leq \text{limiar} \\ 1 & \text{se } \sum_j w_j x_j > \text{limiar} \end{cases} \tag{1.1}$$

Um perceptron por si só é capaz apenas de resolver problemas simples. Mas decisões mais elaboradas podem ser processadas aumentando o número de perceptrons e criando conexões mais complexas, como exemplificados na figura 1.4.

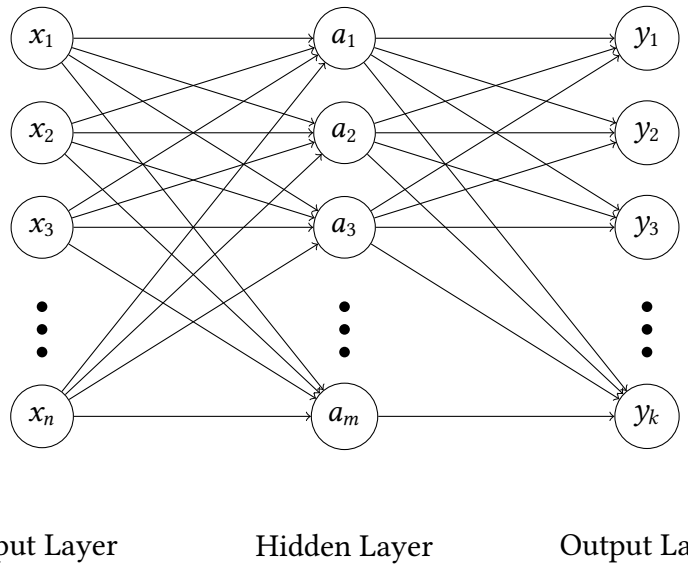


Figura 1.4: *Ilustração de um perceptron multicamadas*

Nesse modelo a primeira camada recebe as entradas e toma decisões de acordo com seus pesos. Há uma ou mais camadas ocultas, formadas por neurônios, que tomam decisões baseadas nas decisões da camada anterior. E de modo similar, a camada de saída recebe a saída da camada anterior para decidir e devolver um resultado.

Podemos descrever os perceptrons de um modo que será útil em outros modelos, reescrevendo a condição $\sum_j w_j x_j > limiar$. Primeiramente podemos escrever $\sum_j w_j x_j$ como um produto escalar, $w \cdot x = \sum_j w_j x_j$, onde w e x são os vetores cujos componentes são os pesos e as entradas, respectivamente. Se movermos o produto escalar para o outro lado da desigualdade e substituir o limiar por um número real b que representará viés de forma que $b \equiv -limiar$, então teremos:

$$saída = \begin{cases} 0 & \text{se } w \cdot x + b \leq 0 \\ 1 & \text{se } w \cdot x + b > 0 \end{cases} \tag{1.2}$$

Um viés positivo facilita o disparamento do perceptron, enquanto que um viés negativo implica numa dificuldade maior do neurônio devolver uma saída 1.

1.7.2 Neurônios Sigmoides

Similar à figura 1.3 que representa o perceptron, podemos pensar em um neurônio sigmoide também contendo entradas $x_1, x_2, \ldots, x_n$, mas que agora em vez de só assumir valores 0 ou 1, estas entradas podem assumir quaisquer valores entre 0 e 1. Além disso, a saída não é mais 0 ou 1, mas sim $\sigma(w \cdot x + b)$, onde σ é a função sigmoide, definida por:

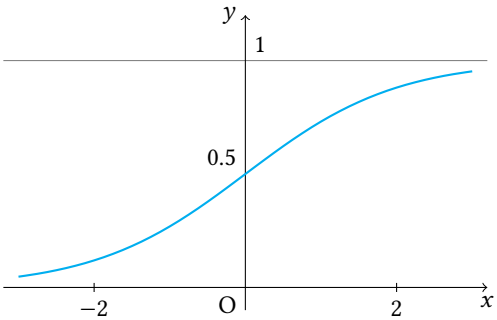


Figura 1.5: *Visualização da função sigmoide.*

$$\sigma(z) \equiv \frac{1}{1 + \exp^{-z}} \tag{1.3}$$

De forma mais explícita, a saída de um neurônio sigmoide com entradas $x_1, x_2, \ldots, x_n$, pesos $w_1, w_2, \ldots, w_n$ e viés b é:

$$\sigma(w \cdot x + b) = \frac{1}{1 + exp(-\sum_j w_j x_j - b)} \tag{1.4}$$

A função sigmoide é um exemplo de função de ativação. Uma das importâncias de se usar uma função de ativação é conseguir abranger problemas não lineares e sem elas a

rede só é capaz de tratar de conjuntos linearmente separáveis (Olah, 2014), reduzindo a sua expressividade e, em turno, seu desempenho. Diferentes funções de ativações podem ser usadas, como por exemplo:

- Tanh ou Tangente hiperbólica: $f(x) = tanh(x)$
- ReLU (*Rectified Linear Unit*): $f(x) = max(0, x)$
- *Softplus* (suavização da ReLU): $f(x) = \frac{1}{\beta} \times log(1 + \exp(\beta \times x))$, onde β é modificado para alterar a suavização aplicada.
- Mish: $f(x) = x \times Tanh(Softplus(x))$

Uma diferença importante do modelo de perceptron para o modelo de neurônio sigmoide está na "suavização" da resposta. Agora, pequenas mudanças Δw_j nos pesos e Δb no viés irão produzir uma pequena mudança $\Delta saida$ na saída do neurônio. Esta saída é aproximadamente:

$$\Delta saida \approx \sum_j \frac{\partial saida}{\partial w_j} \Delta w_j + \frac{\partial saida}{\partial b} \Delta b \tag{1.5}$$

Onde a soma é sobre todos os pesos w_j e $\partial saida/\partial w_j$ e $\partial saida/\partial b$ denotam as derivadas parciais da saída em relação à w_j e b, respectivamente. Desta fórmula observa-se que $\Delta saida$ é aproximadamente uma função linear das mudanças Δw_j e Δb dos pesos e vieses.

1.7.3 Retropropagação

Backpropagation, ou retropropagação, uma abreviação do termo "Propagação retrógrada de erros", é um algoritmo usado numa Rede Neural associado à uma função de perda, que irá calcular o gradiente da função erro com respeito aos pesos dessa rede usando o método do gradiente.

O termo retrógrada vem do fato de que o cálculo deste gradiente procede de forma reversa através da rede, com o gradiente das camadas finais sendo calculado primeiro e com o das primeiras camadas sendo calculado por último. Computações parciais do gradiente de uma camada são reusadas na computação da camada anterior. Este fluxo retrógrado da informação do erro permite uma computação eficiente do gradiente de cada camada, muito mais eficiente do que a computação ingênua de cada gradiente isoladamente.

No contexto de retropropagação, três pré-requisitos são necessários (Brilliant.org, 2022):

- Um *dataset* com pares entrada-saída (x_i, y_i), onde x_i é a entrada e y_i o *target* correspondente à x_i. Denotemos o conjunto de pares entrada-saída de tamanho n por $X = \{(x_1, y_1), ..., (x_n, y_n)\}$.
- Uma Rede Neural *feedforward*, que é uma rede cujas camadas são totalmente conexas (i.e. todos os neurônios de cada camada estão conectados a todos os neurônios da camada seguinte) e que não tem conexões entre neurônios da mesma camada, fazendo com que a informação sempre caminhe para a frente, sem *loops*.

- Uma função de perda entre a saída desejada y_i e a saída prevista $\hat{y}_i$ da rede neural com entrada x_i, para cada par em X e um valor inicial de parâmetros θ, onde θ é o conjunto de todos pesos e vieses da rede. Denotemos essa função como $E(X, \theta)$. Alguns exemplos de funções de perda incluem MAE, MSE e RMSE, propícias para problemas de regressão, e que serão discutidas em na subseção 1.7.5.

Consideremos w_{ij}^k o peso entre o nó i da camada $k-1$ e o nó j da camada k e b_i^k o viés do nó i na camada k.

O treinamento da rede neural com o método do gradiente requer o cálculo do gradiente da função de perda $E(X, \theta)$ com respeito aos pesos w_{ij}^k e vieses b_i^k. Denotando por θ^t o valor dos pesos e vieses da rede no instante t. Usando uma taxa de aprendizado α, cada iteração do método do gradiente atualiza os pesos e vieses segundo:

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E(X, \theta^t)}{\partial \theta} \tag{1.6}$$

Através da *Backpropagation*, a Rede Neural pode se adaptar de acordo com as entradas, encontrando a representação interna mais apropriada para estes dados, com o menor erro. Espera-se atingir maior generalização da informação, fazendo com que novos dados sejam familiares à rede e também possuam erros pequenos.

Porém, há casos em que isso não ocorre, como no *overfitting* e *underfitting* da Rede. A seção 1.7.5 discute com mais detalhes estes e outros termos relacionados à avaliação de desempenho de uma Rede Neural.

1.7.4 Redes Neurais Convolucionais

Redes Neurais Convolucionais (*Convolutional Neural Networks, Convnets* ou CNNs) são um tipo de rede neural muito usado para classificação de imagens (Krizhevsky *et al.*, 2017). Uma das características de CNNs é sua capacidade de detectar *features* das imagens independentemente de sua posição (*location invariant*).

Em CNNs é comum o uso de tensores, que são objetos que armazenam dados e operações algébricas que definem transformações lineares entre vetores, escalares e até mesmo outros tensores.

Tipicamente, *Convnets* são compostas por três tipos diferentes de camada:

Convolutional layers

A entrada de uma CNN é um tensor de forma: (número de entradas) × (altura da entrada) × (largura da entrada) × (número de canais). Depois de passar por uma camada convolucional, a entrada é abstraída em um mapa de *feature*, também denominado de mapa de ativação, com formato: (número de entradas) × (altura do mapa de *feature*) × (largura do mapa de *feature*) × (canais do mapa de *feature*).

Devido às técnicas usadas em CNNs, o processamento de dados executado numa camada convolucional é capaz de tratar entradas com muitos dados de forma mais eficiente do que Redes Neurais *feedforward* convencionais pela redução do número de parâmetros livres.

Por exemplo, uma rede totalmente conexa para uma matriz bi-dimensional de tamanho 100×100, possui dez mil pesos para cada neurônio da segunda camada. Se usássemos uma CNN, entretanto, podemos usar uma região de *tiling* de 5×5 (uma técnica que consiste em cortar a imagem em pedaços menores para auxiliar no processamento que também pode ser denominada dimensão do *kernel*), agora só seriam necessários 25 parâmetros por neurônio. Assim, é possível criar redes neurais com mais camadas (Aghdam e Heravi, 2017) e por um custo menor.

Pooling layers

Pooling layers podem acompanhar camadas convolucionais e serem locais ou globais. Sua função é de reduzir o número de dimensões dos dados combinando *clusters* de neurônios de uma camada em um único neurônio da próxima camada. *Pooling Layers* locais combinam *clusters* pequenos, como por exemplo de 2×2. Já *Pooling Layers* globais agem em todos os neurônios de um mapa de *feature*. Dois métodos são populares para este tipo de *pooling*: *Max Pooling*, que usa o valor máximo de cada *cluster* de neurônios do mapa, enquanto que *Average pooling* utiliza o valor médio.

Fully connected layers

São camadas que contêm neurônios onde cada um deles está conectado a todos os neurônios de uma outra camada. É comum nas CNNs que a matriz de entrada achatada passe por pelo menos uma camada totalmente conexa.

Hiper-parâmetros

Uma CNN também possui diversos hiper-parâmetros que controlam o processo de aprendizado:

- *Kernel Size*: controla o número de *pixels* processados ao mesmo tempo. Valores comuns para a dimensão do *kernel* são 2×2 e 3×3, 1×3 e 3×1. Representa a janela de análise.
- *Padding*: é a adição de *pixels* de valor 0 nas bordas da imagem, para manter o tamanho do mapa na saída. O número de *pixels* para adicionar em cada lado é $\lfloor \frac{k}{2} \rfloor$ *pixels*, onde k é o tamanho do lado da dimensão do *kernel* escolhida. Por exemplo, uma camada convolucional usando *kernel* de 3×3 adicionaria 1 *pixel* para cada lado da imagem, isto é, usaria um *padding* de 2 *pixels*.
- *Stride*: refere-se ao número de *pixels* que são movidos para cada deslocamento da janela de análise, ou seja, um *stride* de 2 significa que a janela de análise se move 2 *pixels* em relação ao seu antecessor.
- *Dilation*: envolve ignorar *pixels* em um *kernel*, diminuindo o custo em processamento e memória com redução de perda do sinal. Uma dilatação de 2 em um *kernel* 3×3 expande-o para 5×5, mas continuamos mantendo o processamento de apenas 9 *pixels*. Uma dilatação de 4 neste *kernel* expandiria-o para 9 x 9, mantendo o processamento apenas dos 9 *pixels* centrais.

Regularização

Regularização é o processo de introduzir informações adicionais de modo a diminuir *overfitting* ou lidar com efeitos colaterais de determinadas escolhas de design da rede. Diversos métodos existem na literatura, mas mencionaremos apenas aqueles que foram usados em nosso trabalho.

- *Dropout*: esta técnica, em cada época de treinamento, faz com que nós individuais sejam ignorados com uma probabilidade $1-p$ ou mantidos com uma probabilidade p. Além de diminuir efeitos de *overfitting*, este método também acelera o treinamento.
- *Early Stopping*: é um dos métodos mais simples de prevenir *overfitting* e envolve simplesmente em parar de treinar a rede antes que isto ocorra. Uma estimativa desse momento pode ser feita com os dados separados para a validação — uma elevação da *loss* de validação enquanto que a *loss* de treinamento continua diminuindo pode indicar que estamos lidando com *overfitting* e que possa ser melhor interromper o treino no ponto de mínimo local desta *loss* de validação.
- *Weight decay*: é uma técnica que consiste em acrescentar um termo à função de perda, proporcional à soma absoluta dos pesos (*L1 Norm*) ou à soma quadrática dos pesos (*L2 Norm*) em cada nó. *L2 norm* tende a encorajar a rede a usar todas as entradas, enquanto que a regularização L1 propicia à redução do efeito de ruído sobre os dados. Ambas podem ser usadas na mesma rede.
- Uso da Função de Ativação *Mish*: Em relação à outras funções de ativação, existem trabalhos (Misra e Diganta, 2019) que afirmam que, devido à forma da primeira derivada, estas são mais fáceis de minimizar e mais robustas à mudanças de hiperparâmetros e da arquitetura da rede.

1.7.5 Avaliação de Desempenho

Para analisar o desempenho da rede, precisamos definir uma forma de avaliação. Primeiramente vamos considerar os dados, estes devem ter valores para entrada na rede e suas respectivas respostas corretas/esperadas, normalmente feitas pela supervisão de profissionais ou com algum tipo de semi-automação. Dividimos eles em três conjuntos disjuntos: Treino, Validação e Teste.

- Treino: dados de treino são usados para ajustar os pesos e vieses do modelo, acompanhados de uma função de *loss* e um mecanismo de retropropagação. Também é possível avaliar *underfitting* nessa fase.

  O *underfitting* é caracterizado por erros grandes nas predições já no treino, indicando que o modelo não é capaz de identificar um padrão nos dados do treino e falhará em generalizar novos dados. Algumas das possíveis causas são a falta de dados, muito ruído nos dados, modelo enviesado e arquitetura de modelo muito simples para o problema.

  Possíveis soluções incluem o uso de mais *features*, melhoria no pré-processamento dos dados, aumento da complexidade do modelo e treinar o modelo por mais épocas[3].

[3] Época ou *epoch* é um termo utilizado para representar um ciclo de treinamento de uma Rede Neural.

- Validação: usado para refinar hiper-parâmetros do modelo, também estimar quando o modelo está sofrendo de *overfitting* dos dados, determinando uma época para finalizar o treinamento.

  Overfitting é o termo usado para descrever o comportamento de um modelo que apresenta pouco erro nos dados de treino, mas erros grandes nos demais dados. Isso acontece quando o modelo aprende padrões muito únicos no treinamento, de forma a afetar a capacidade de generalização para novos dados. Possíveis causas são a alta complexidade da arquitetura do modelo, ruído nos dados, pouca quantidade de dados e muita variância no modelo.

  Para resolver este problema algumas abordagens são o aumento da quantidade de dados de treinamento, o uso de técnicas de regularização e aplicação do *K-Fold cross validation*, que é um procedimento em que se criam K partições mutualmente exclusivas e de mesmo tamanho dos dados, escolhe-se de forma circular uma delas para teste e as demais para treino.

- Teste: dados reservados que não foram utilizados para treino ou validação. Após o treino e escolha do modelo, utiliza-se os dados de teste para avaliar a performance do modelo sobre como a rede se sairia em dados nunca visto antes.

Quando uma rede computa um valor a partir de entradas, é necessário definir qual o objetivo de nossa rede, ou seja, definir uma função objetiva. Esta pode ser de maximizar uma classe de probabilidade em aprendizado Bayesiano, maximizar uma função de desempenho de um algoritmo genérico, ou minimizar uma função de custo/perda (*cost/loss*).

Existem diversas funções que são capazes de computar perda, mas para nosso problema de regressão iremos usar três: *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE) e *Root Mean Squared Error* (RMSE).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \qquad MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \qquad RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{1.7}$$

Nas equações referenciadas em 1.7, n representa o número de amostras, $\hat{y}$ é o valor previsto e y o valor esperado.

Para a avaliação final do desempenho do modelo foram utilizadas métricas de regressão e classificação sobre o erro nos dados de teste. Os resultados da utilização destas métricas serão apontados no capítulo 3.

Métricas para regressão

Neste projeto as métricas utilizadas para a avaliação da regressão foram:

Erro Absoluto: Diferença absoluta entre o valor predito e o *target*.

$$\epsilon_a = |y_i - \hat{y}_i| \tag{1.8}$$

Erro Relativo: Calculado como a razão do erro absoluto entre o *target* e a previsão, em relação ao *target*. Mostra em porcentagem a distância entre a estimativa e o valor real.

$$\epsilon_r = \frac{|y_i - \hat{y}_i|}{y_i} \tag{1.9}$$

Métricas para classificação

Nesse projeto foi feita a classificação binária, em que se define uma classe para as amostras que possuem uma característica de interesse (classe positiva) e outra para as que não a possuem (classe negativa). Um conceito importante para a avaliação de seu desempenho é a matriz de confusão, que é composta por quatro valores:

- *True Positive* (TP): amostra da classe positiva que foi corretamente classificada;
- *False Positive* (FP): amostra da classe negativa que foi erroneamente classificada como positiva;
- *True Negative* (TN): amostra da classe negativa que foi corretamente classificada;
- *False Negative* (FN): amostra da classe positiva que foi erroneamente classificada como negativa.

A partir da contagem de amostras nessas classes é possível obter outras métricas, as que foram utilizadas nesse projeto são:

Acurácia: mede o quão bem as classes foram corretamente classificadas e assume valores entre $[0, 1]$. Sendo que quanto mais perto de 1, melhor é a acurácia. É dada por:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1.10}$$

Sensitividade (SN): Também chamada de *recall* (REC) ou *True Positive Rate* (TPR), é a razão entre o número de amostras da classe positiva que foram corretamente classificadas e a quantidade total de amostras da classe positiva. Se refere à probabilidade de que resultados obtidos como positivos *realmente* sejam positivos. Seus valores variam em $[0, 1]$, sendo números mais próximos a 1 considerados melhores. Pode ser dada pela equação:

$$\text{Sensitividade} = \frac{TP}{TP + FN} \tag{1.11}$$

Especificidade (SP): Conhecida também como *True Negative Rate* (TNR). Indica a razão entre o número de amostras classificadas corretamente na classe negativa e a quantidade total de amostras da classe negativa. Indica a probabilidade que um resultado negativo obtido seja *realmente* negativo. Possui valores em $[0, 1]$ e quanto mais próximo a 0, pior é a especifidade.

$$\text{Especifidade} = \frac{TN}{TN + FP} \tag{1.12}$$

Precisão (PREC): Ou *Positive Predictive Value* (PPV), é a razão entre o número de amostras

da classe positiva que foram corretamente classificadas e a quantidade total de amostras que foram classificadas na classe positiva. Os valores variam em [0, 1], sendo que quanto mais próximo a 1, melhor a precisão.

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{1.13}$$

F_1-score: média harmônica da precisão e *recall*. Assume valores entre $[0, 1]$, sendo 1 o caso de precisão perfeita, já o valor 0 indica que a precisão ou o *recall* é nulo. Definida por:

$$F_1 = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{1.14}$$

Coeficiente de Correlação de Matthews (MCC): Diferentemente da acurácia e F_1-score, esse coeficiente considera as quatro categorias da matriz de confusão, levando em conta também o número de amostras positivas e negativas do *dataset* (Chicco e Jurman, 2020) e é usado para medir a qualidade da classificação. Assume valores entre $[-1, 1]$, onde +1 e -1 indica uma estimação perfeita ou totalmente incorreta, respectivamente, e 0 indica que o método não tem performance melhor do que escolher aleatoriamente os resultados. A fórmula da MCC é:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \tag{1.15}$$

Capítulo 2

Metodologia

Para nosso projeto utilizaremos dados da primeira fase de estudo do projeto SPIRA. Estes dados incluem trechos de fala coletados de pacientes com COVID-19 que compareceram à enfermaria de dois hospitais durante o período de junho à julho de 2020, além de pacientes saudáveis que foram adicionados como grupo de controle (Casanova *et al.*, 2021). Junto destes, temos também para cada paciente a sua frequência cardíaca e saturação de oxigênio, obtidos através do uso de um oxímetro.

Os sinais de áudio passarão por pré-processamento como filtros, janelamento, adição de ruídos e FFT. Em seguida extrairemos *features* através de MFCCs ou mel espectrogramas e utilizaremos diferentes modelos de redes neurais em busca daquela que melhor minimiza o erro para previsão dos níveis de SpO_2.

Existem diversas questões que serão discutidas e refinadas ao longo do estudo. Observamos como pontos de interesse os seguintes:

- Conhecimento das representações de sinais de áudio no computador;
- Conhecimento de técnicas de processamento de sinais;
- Conhecimento das bibliotecas computacionais usadas no projeto SPIRA;
- Decisões em como lidar com o ruído de fundo durante as coletas para evitar a geração de vieses nas redes neurais;
- Decisões em como lidar com as amostras de ruído ambiente que foram coletadas de modo a reduzir os vieses nas amostras de áudio da fala;
- Manuseio do *dataset* contendo o áudio dos pacientes coletados;
- Determinação das *features* mais adequadas para uma boa acurácia;
- Análise e determinação da viabilidade de nosso modelo para o problema proposto;

2.1 Dados

Os arquivos de áudio que iremos utilizar foram gravados através do aplicativo WhatsApp[1], possuindo originalmente a extensão OGG[2] e que posteriormente foram convertidos para WAV[3] antes de serem processado pela rede. A razão desta conversão do áudio é porque este último formato não comprime o áudio, diminuindo possíveis inconsistências em nossos experimentos. Eles podem ser encontrados sob licença CC BY-SA 4.0 na página do Projeto SPIRA no Github.

Há dois tipos de áudios disponíveis: gravações de fala e ruídos hospitalares. Dentre os áudios de fala temos aqueles de pacientes com COVID-19, coletados por um profissional da área médica na enfermaria, e vozes doadas via interface web de pessoas consideradas como grupo de controle, ou seja, saudáveis. Neste trabalho vamos apenas nos atentar aos áudios de voz da primeira categoria (de pacientes com COVID-19), pois são para os quais temos medições da SpO_2 provenientes do oxímetro, coletados juntamente com os áudios.

Abaixo está representada uma visão geral dos áudios de fala. Vemos um desbalanceamento das classes de SpO_2, sendo que algumas não possuem nenhuma amostra. Também nota-se uma menor quantidade de áudios de vozes femininas e com menor idade, além do fato de que a maioria dos casos graves de insuficiência foi do público masculino.

No total temos 218 amostras, um número relativamente baixo de amostras para trabalhos com Redes Neurais, mas esta é uma situação muito comum em projetos científicos relacionados à área da Saúde.

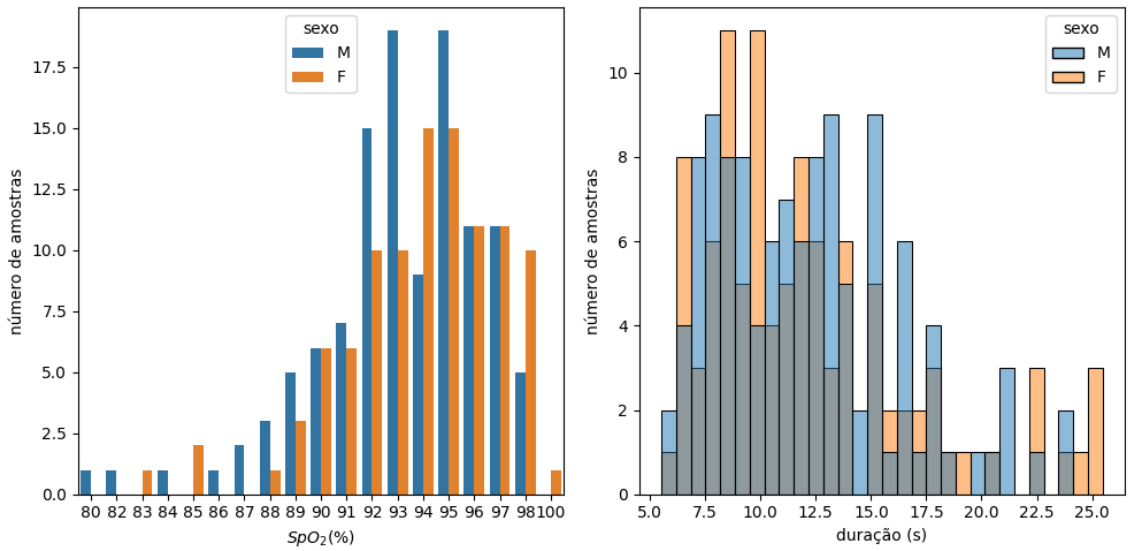


Figura 2.1: *Histogramas mostrando o número de amostras de áudio por classe SpO_2 (%) e duração (s)*

[1] Aplicativo multiplataforma de mensagens instantâneas e chamadas de voz.

[2] Formato de áudio mantido pela Xiph.Org Foundation. É utilizado principalmente para streaming.

[3] *Waveform Audio File Format* é um formato de áudio.

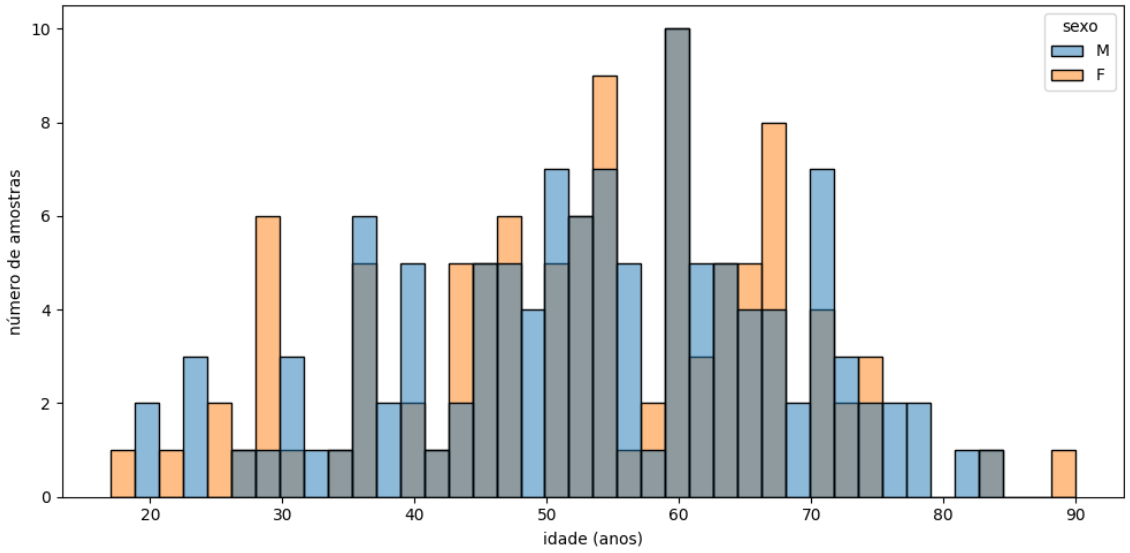


Figura 2.2: *Histograma mostrando o número de amostras de áudio por idade (anos completos)*

2.1.1 Janelamento

Como forma de amenizar o desbalanceamento e a falta de dados, podemos usar uma técnica de *data amplifying* janelando os áudios em tamanho correspondente a 4 segundos, com um salto de 1 segundo durante o treinamento, validação e teste dos modelos de Redes Neurais.

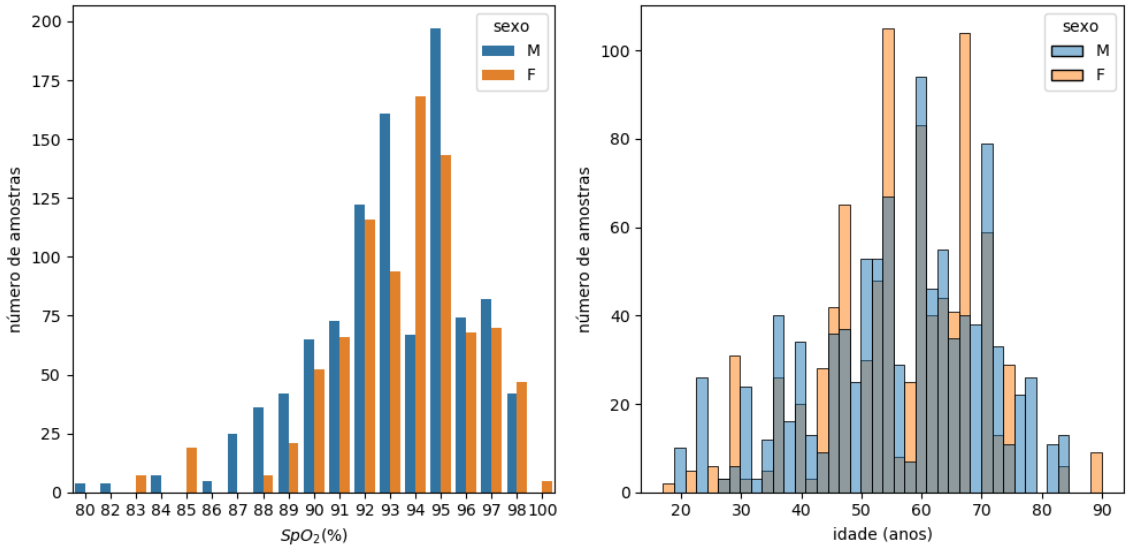


Figura 2.3: *Histograma mostrando o número de amostras de áudio por SpO_2 ()% e idade (anos completos) após os janelamentos de 4 segundos*

Como podemos ver na figura 2.3, o uso do janelamento diminuiu a disparidade entre os sexos e amplificou a quantidade de dados, totalizando agora 1889 amostras.

2.1.2 Formato do arquivo de entrada

Para o programa desenvolvido no projeto, o formato aceito de arquivos de metadados é CSV[4] onde as informações de caminho do áudio, idade, sexo (Feminino/Masculino) e nível de SpO_2 no sangue estão organizados da seguinte forma:

```
caminho_do_arquivo/audio.wav,sexo,idade,spO2
```

2.1.3 Processamento do áudio

O processamento dos áudios é feito com o auxílio da biblioteca Librosa[5]. Os áudios são carregados com taxa de amostragem de 16kHz e transformados em entrada para a rede.

Durante a fase dos experimentos testou-se o número de canais do áudio (mono ou estéreo), o tipo de entrada da rede (MFCC ou mel espectrograma) e os parâmetros ligados à MFCC, como o número de amostras entre *frames* sucessivos (*hop length*), tamanho dos *frames* (*window length*), comprimento da janela da FFT, quantidade de MFCC, quantidade de *mel bands*, implementação da escala mel (HTK ou Slaney) e a adição de ruídos hospitalares. Os resultados estão no capítulo 3.

2.1.4 *Split* dos dados

Os dados foram divididos antes do janelamento em três partições: treinamento, validação e teste, seguindo as proporções de 80%, 10% e 10%, respectivamente. Dividir os pacientes antes do janelamento garante que as janelas de áudio de um paciente encontram-se somente em um destes conjuntos de dados, diminuindo os vieses dos resultados, que poderiam ser causados por padrões específicos de cada amostra de fala.

[4] *Comma-Separated Values*, é um formato de arquivo de texto, representando geralmente dados tabulares.

[5] Biblioteca Python para análise de áudio e música.

2.2 Rede Neural

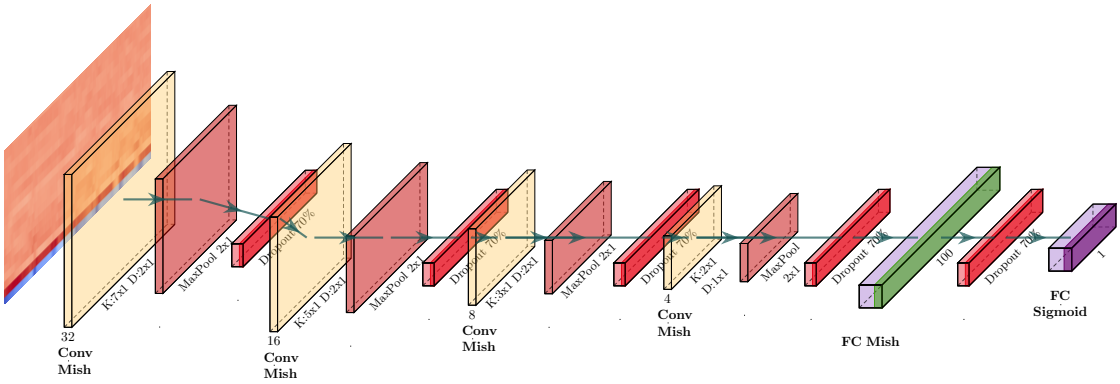


Figura 2.4: *Representação da CNN (Casanova et al., 2021)*

Baseando-se nos estudos realizados previamente pelo grupo SPIRA, a rede na qual a maioria dos nossos experimentos se encontra é uma adaptação da descrita na figura 2.4, graciosamente cedida pelos autores, com quatro camadas convolucionais, duas camadas totalmente conexas e com a saída de um número real.

A figura também descreve os filtros, *kernels*, número de neurônios e as funções de ativação envolvidas. As siglas representadas são: K para o tamanho do *kernel*, D para o tamanho da dilatação convolucional e FC para as camadas totalmente conexas (*fully connected*). Notar também as camadas de *Batch Normalization* (Zhang *et al.*, 2018) após as convolucionais.

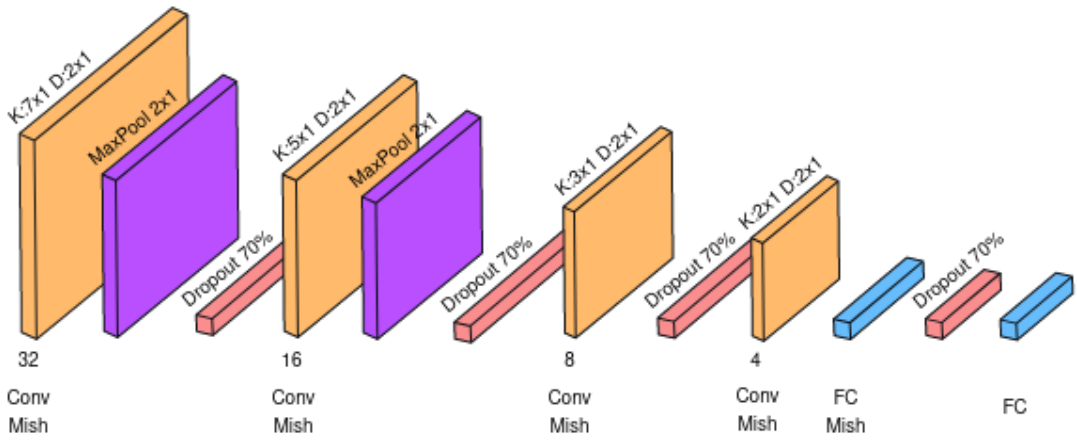


Figura 2.5: *Representação da CNN usada em nosso trabalho*

Na figura 2.5, temos uma representação simplificada da nossa rede. As duas redes são muito similares, diferindo basicamente na última camada, onde usamos uma Função de Ativação Linear diferente da Função Sigmoide. Isto por que nosso trabalho lida com regressão em vez de classificação binária.

A entrada da rede é uma MFCC ou um Mel espectrograma, que pode ser representada por uma imagem, passando por uma camada convolucional que expande o número de canais e três camadas convolucionais que diminuem sua resolução (totalizando quatro camadas convolucionais), passando então por duas camadas totalmente conexas para a saída que representa a previsão da SpO_2.

Foram testados, com o auxílio do Optuna[6], modelos com menos camadas, mas no geral eles não obtiveram desempenho melhor, além de necessitar de mais neurônios para obter resultados equiparáveis, elevando o custo computacional e sendo assim avaliados inferiormente ao modelo original.

Outro desafio enfrentado pelas redes com menos camadas foi a queda muito acelerada do erro no treinamento, rápida estabilização e *overfitting*, sugerindo que estes modelos chegam à conclusões errôneas causadas pela variação muito rápida do gradiente.

2.2.1 Hiper-parâmetros

Em relação aos hiper-parâmetros, testou-se os otimizadores *Adam*, *AdamW* e RMS, preferindo-se o primeiro. Como função de *loss* foram comparadas MSE, MAE e RMSE. Na maioria dos experimentos usou-se 1000 épocas, mas também foi testado 500 épocas. O tamanho do *batch* na maioria dos experimentos foi 5, mas também experimentou-se usar um *batch* de 10 amostras.

Como mitigação de *overfitting*, as seguintes medidas foram tomadas: usar *Mish* como função de ativação, *weight decay* de 0.01 e taxa de *dropout* de 70% em todas as camadas exceto a última, como no estudo original.

[6] *Framework* para *fine-tuning* de hiperparâmetros. Ela busca automatizar a busca por hiperparâmetros.

Capítulo 3

Análise dos resultados

Cada modelo foi treinado com os dados alocados para treinamento e, ao final de cada época, foi avaliado o seu desempenho na tarefa de predizer os *targets* do conjunto de validação, onde foi possível identificar o *overfitting* e escolher o número de épocas adequado para treinar esta rede. A rede treinada selecionada foi aquela que apresentou o menor erro no conjunto de validação e para estas redes foi avaliado seu desempenho no conjunto de teste.

Realizou-se cada experimento pelo menos 5 vezes com sementes aleatórias, pois foi devido à mudança de sementes que obtivemos grandes variações na saída da rede, nos levando a repetir o experimento diversas vezes para atingir consistência nos resultados. A decisão em permitir esta aleatoriedade foi feita porque ao fazer um gradiente que relaciona a variação do erro *versus* a variação das épocas, existem diversos mínimos locais para os quais sementes distintas podem levar o modelo a convergir. Por isso, o uso de sementes predeterminadas pode adicionar vieses inesperados em nosso trabalho.

Neste capítulo todos os resultados são referentes ao desempenho médio do modelo nos experimentos usando os dados de teste.

3.1 Regressão

Dado o problema de regressão da previsão do nível de SpO_2, esta seção apresenta os resultados obtidos utilizando-se as métricas descritas em 1.7.5.

3.1.1 Resultados

Comparações MAE vs MSE com diferentes *Learning Rates*

A tabela 3.1 ilustra o efeito do *Learning Rate* sobre as funções de *loss* MSE e MAE. As colunas *diff mean, diff median e avg std dev* se referem à média do erro absoluto, a mediana do erro absoluto e o desvio padrão médio dos erros absolutos de cada experimento, respectivamente. Já *diff std dev* se refere ao desvio padrão dentre os erros absolutos médios de cada experimento.

No.	Learning Rate	Loss_fn	Scheduler	diff mean	diff median	diff std dev	avg sdtdev	epoch
1	1E-03	MSE	Noam	2.1953	2.1002	0.5162	1.8718	1000
11	1E-03	MAE	Noam	8.3005	5.5037	6.4857	2.7548	1000
2	1E-02	MSE	Noam	3.0781	3.1182	0.1357	1.8549	1000
12	1E-02	MAE	Noam	2.9445	0.0081	2.9427	1.8463	1000
3	1E-04	MSE	Noam	8.6708	9.7579	5.5056	2.9327	1000
13	1E-04	MAE	Noam	3.0744	3.0838	0.0177	2.3827	1000
4	5E-06	MSE	None	3.0529	3.0434	0.4365	1.9150	1000
14	5E-06	MAE	None	5.1653	3.1105	3.6610	2.6109	1000
5	1E-05	MSE	None	4.3657	3.9688	1.1770	2.7699	1000
15	1E-05	MAE	None	3.3536	3.0983	0.8787	2.3777	1000
6	5E-06	MSE	Noam	3.3188	3.3510	0.3022	2.0707	1000
16	5E-06	MAE	Noam	3.6960	3.1928	0.9283	2.3470	1000
7	1E-05	MSE	Noam	3.3938	3.3664	1.0293	2.1522	1000
17	1E-05	MAE	Noam	3.6738	3.4029	0.7697	2.4328	1000
8	1E-03	MSE	None	3.0428	3.1218	0.2390	1.8822	1000
18	1E-03	MAE	None	3.0913	3.0842	0.0344	1.8613	1000
9	1E-02	MSE	None	3.1318	3.1157	0.0548	1.8748	1000
19	1E-02	MAE	None	3.3642	3.4142	0.1791	1.9773	1000
10	1E-04	MSE	None	4.4271	3.0028	3.5758	2.5536	1000
20	1E-04	MAE	None	3.1266	3.1261	0.0573	2.0114	1000

Tabela 3.1: *Experimentos MSE x MAE, Learning Rates e Schedulers*

Os dois desvios padrões citados têm significados distintos: *avg std dev* reflete as **variações entre o valor previsto e o real em cada época dos experimentos**, enquanto que *diff std dev* diz respeito às **variações dos resultados dentre as repetições do mesmo experimento**. Esta última foi adicionada porque alguns experimentos possuíam grandes variações entre as repetições, e a média e mediana não pareciam refletir estas diferenças com a devida clareza.

Nestes experimentos o *scheduler* utilizado é o Noam, com 10 passos de aquecimento. Inicialmente, tentamos observar a influência da *Learning rate* sobre os modelos.

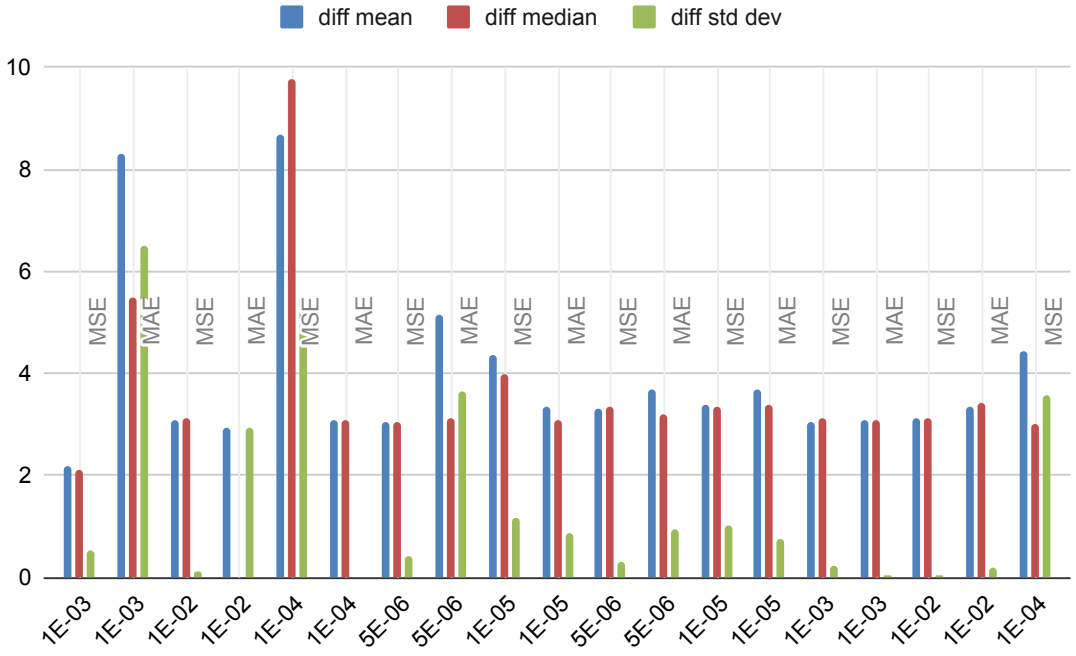


Figura 3.1: *Gráficos comparando MAE e MSE com diferentes learning rates*

A figura 3.1 ilustra os resultados da tabela 3.1. Destes experimentos iniciais é difícil tirar conclusões devido ao erro similar entre a maioria dos testes, mas pode-se observar o bom desempenho do experimento 1, de *learning rate* 10^{-3}, usando-se MSE e *scheduler* Noam.

Experimentos em relação ao número de dados e de batches

No.	Learning Rate	Loss_fn	Scheduler	diff mean	diff median	diff std dev	avg sdtdev	epoch	
3	1E-04	MSE	Noam	8.6708	9.7579	5.5056	2.9327	1000	
29	1E-04	MSE	Noam	5.3151	4.9493	1.8921	2.8407	500	batch=10
34	1E-04	MSE	Noam	7.6557	7.1684	4.6506	2.9791	500	Metade dos dados
39	1E-04	MSE	Noam	6.1440	6.2624	1.8961	3.0864	500	batch=10 e metade dos dados
11	1E-03	MAE	Noam	8.3005	5.5037	6.4857	2.7548	1000	
30	1E-03	MAE	Noam	14.1646	14.5545	7.7100	3.3306	500	batch=10
35	1E-03	MAE	Noam	6.0444	3.0857	4.0808	2.3808	500	Metade dos dados
40	1E-03	MAE	Noam	7.2507	3.2336	8.0005	2.4715	500	batch=10 e metade dos dados
13	1E-04	MAE	Noam	3.0744	3.0838	0.0177	2.3827	1000	
31	1E-04	MAE	Noam	5.0091	3.1907	3.1771	2.3820	500	batch=10
36	1E-04	MAE	Noam	5.5177	3.1208	5.3832	2.1434	500	Metade dos dados
41	1E-04	MAE	Noam	10.0900	10.6203	5.6004	3.1650	500	batch=10 e metade dos dados
14	5E-06	MAE	None	5.1653	3.1105	3.6610	2.6109	1000	
32	5E-06	MAE	None	3.6958	3.2278	0.8466	2.5316	500	batch=10
37	5E-06	MAE	None	2.9541	3.1143	0.4687	1.9337	500	Metade dos dados
42	5E-06	MAE	None	3.5617	3.8537	0.7244	2.1924	500	batch=10 e metade dos dados
20	1E-04	MAE	None	3.1266	3.1261	0.0573	2.0114	1000	
33	1E-04	MAE	None	8.0226	9.3375	4.5712	3.0211	500	batch=10
38	1E-04	MAE	None	7.7025	4.3882	6.1741	2.5607	500	Metade dos dados
43	1E-04	MAE	None	6.8529	3.1490	5.5178	2.4614	500	batch=10 e metade dos dados

Tabela 3.2: *Experimentos variando batches e número de dados*

Nos experimentos da tabela 3.2 comparamos os resultados quando variamos *batches*

de 5 para 10 e também quando reduzimos o número de dados de treinamento pela metade. Cada linha em azul representa um experimento que foi selecionado para ser repetido, variando o número de dados e o número de *batches*.

A intenção em reduzir a quantia de dados de treinamento foi investigar se as grandes variações dos resultados estavam diretamente relacionados com o baixo número de dados e/ou se o problema seria agravado, o que nem sempre ocorreu.

Uma característica que estes experimentos da linha azul têm em comum é a grande variação dentre as repetições. Ao compararmos os valores de *diff mean* dos resultados das linhas em verde com os da linha azul relacionada, não é possível tirar conclusões elaboradas. Ficou evidente a necessidade de repensar este problema com novos experimentos ou até uma nova abordagem nas comparações para que seja possível analisar mais profundamente a questão de variância *versus* o número de dados e/ou o número de *batches*. Apesar disso, acreditamos que grande parte dessa inconsistência se deve à falta de dados e por isso não nos aprofundamos mais nesse ponto, deixando esta questão em aberto.

Experimentos com *weight decay, optimizer* e dimensão das camadas *fc*

No.	Learning Rate	Loss_fn	Scheduler	diff mean	diff median	diff std dev	avg sdtdev	epoch
1	1E-03	MSE	Noam	2.3166	2.3507	0.0999	1.8718	1000
21	same as 1 but fc1=200			1.9707	1.8938	0.3217	1.9394	1000
22	same as 1 but fc1=50			3.2201	2.7746	1.6272	1.8388	1000
23	same as 1 but optimizer=AdamW			2.3617	2.4989	0.4368	2.0790	1000
24	same as 1 but optimizer=RMS			3.0863	3.2002	0.3591	1.8980	1000
26	same as 1 but with weight_decay=0.1			2.7435	2.7729	0.3457	1.9053	1000
27	same as 1 but with weight_decay=0.2			2.8422	2.8093	0.2594	1.9266	1000
28	same as 1 but with weight_decay=0.005			2.8522	2.5507	0.5550	1.9492	500

Tabela 3.3: *Experimentos variando wd, optimizers e fcs*

Observando o bom desempenho do experimento 1, tentou-se criar variações ao redor desta configuração. Experimentou-se dobrar e dividir pela metade o número de nós da primeira camada totalmente conexa, usar otimizadores *AdamW* e *RMS* em vez de *Adam*, além de variar o valor de *Weight Decay*.

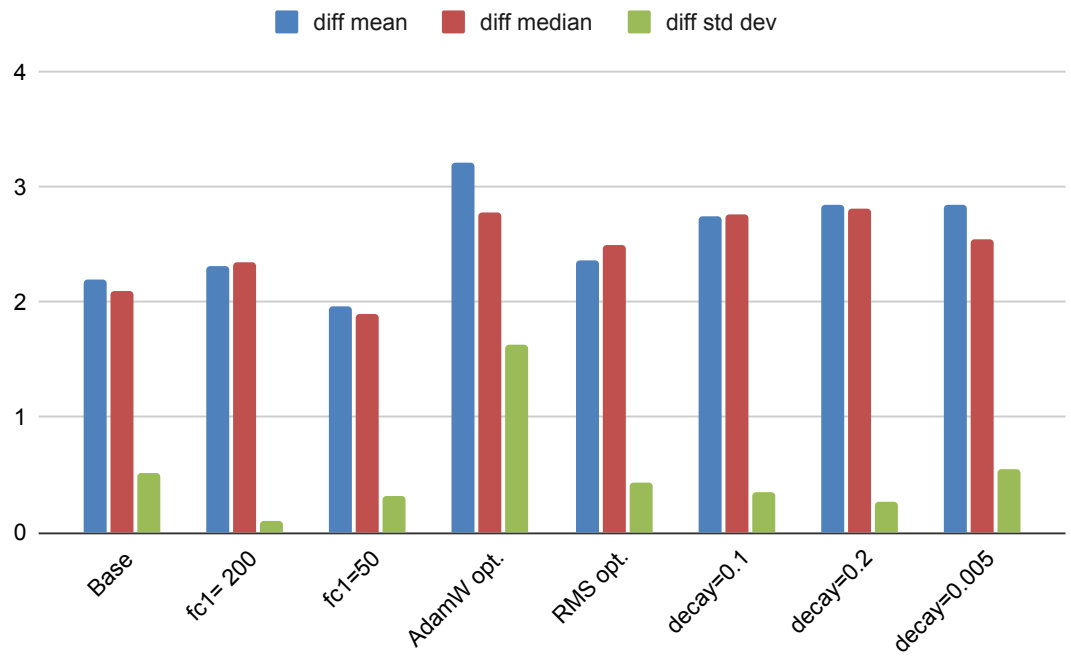


Figura 3.2: *Ilustração da Tabela 3.3*

Com a figura 3.2, vemos que não houve muita diferença com a mudança do número de nós na primeira camada totalmente conexa. Temos um desempenho inferior ao usarmos AdamW, mas RMS não teve um desempenho muito diferente de Adam. As variações de *weight decay* acima e abaixo da padrão de 0.01 não pareceram trazer bons resultados.

Experimentos com MFCC e Mel espectrogramas

No.	Learning Rate	Loss_fn	Scheduler	diff mean	diff median	diff std dev	avg sdtdev	epoch	Comentário
1	1E-03	MSE	Noam	2.1953	2.1002	0.5162	1.8718	1000	MSE
25	same as 1 but with melspectrogram			3.0863	3.2002	0.3591	1.9930	1000	
11	1E-03	MAE	Noam	8.3005	5.5037	6.4857	2.7548	1000	MAE
56	same as 11 but with melspectrogram			5.0918	3.3147	3.1582	4.8806	1000	
44	same as 1 but with loss_fn=RMSE			3.1796	3.1036	0.2352	2.1148	500	RMSE
57	same as 44 but with melspectrogram			8.0264	4.2873	8.7809	2.8505	1000	

Tabela 3.4: *Experimentos comparando MFCCs vs Mel espectrogramas*

Fizemos uma comparação entre usar MFCC e Mel espectrogramas como *features* de entrada para nossa rede, usando as três funções de loss.

Na figura 3.3, vemos um desempenho melhor ao usar MFCCs em vez de mel espectrogramas com exceção da função MAE. Entretanto, como MSE e RMSE obtiveram melhores resultados, preferimos manter nossa decisão em usar MFCCs.

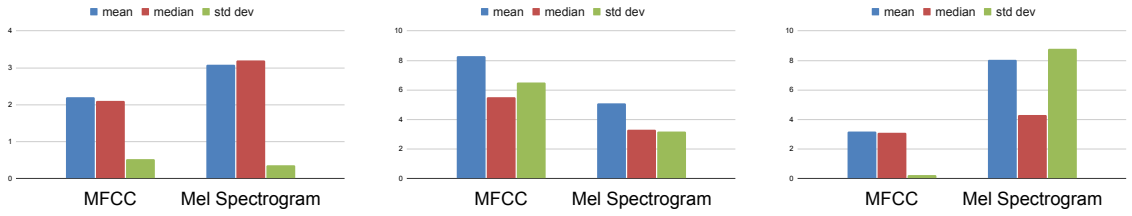


Figura 3.3: *Comparação Mel vs MFCC usando MSE, MAE e RMSE, respectivamente*

Áudio mono vs estéreo

A figura 3.4 mostra o erro nos testes para experimentos agrupados dois a dois, onde a única diferença entre suas configurações é o número de canais do áudio. Podemos observar que a quantidade de canais do áudio não parece interferir diretamente no desempenho da rede ou age em conjunto com outros hiper-parâmetros.

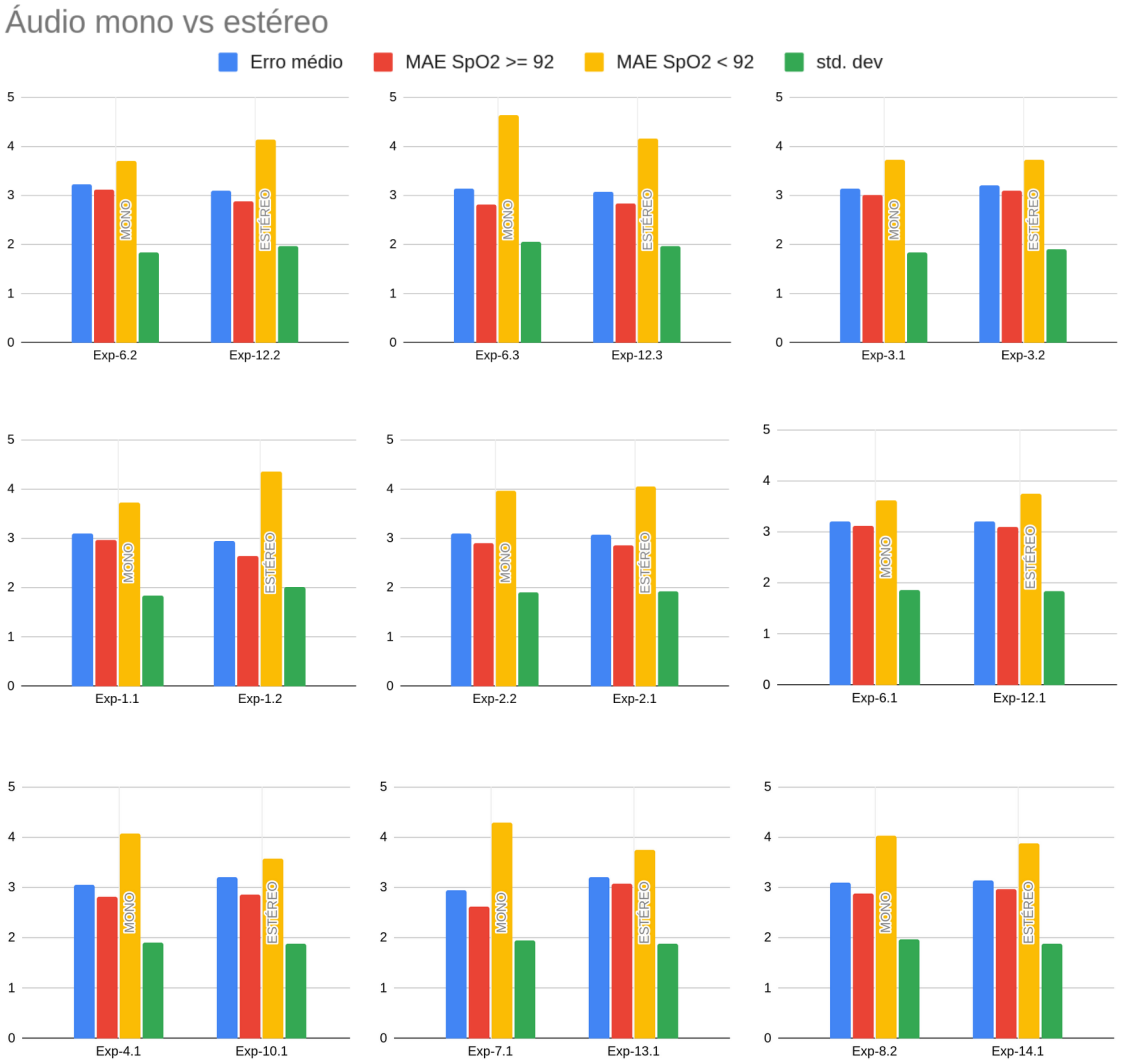


Figura 3.4: *Gráficos com erros dos testes para áudio mono e estéreo*

Para os experimentos 1.x, 2.x, 12.x e 6.x, o uso de áudio estéreo trouxe melhores resultados, para o experimento 3.x ele se mostrou indiferente e para os demais vemos que o áudio mono se sobressaiu.

O único padrão que pôde ser observado é o maior desvio padrão para o áudio estéreo.

Parâmetros para a MFCC

No geral a variação dos parâmetros foi irrelevante no desempenho do modelo. As configurações tiveram MAE em torno de 3 e erro relativo médio absoluto em torno de 3,4%. Os erros relativos não serão discutidos aqui, pois a diferença entre eles é baixa, mais de 80% dos experimentos está entre 3,1% e 3,5%. Abaixo a tentativa de estimar como e quanto cada um dos parâmetros influencia nos resultados:

- **HTK/Slaney**: implementações da escala mel

  Como mostrado na figura 3.5, a implementação da escala mel não teve efeito significativo sobre os resultados. Em média o erro permaneceu ao redor de 3 e o desvio padrão perto de 2.

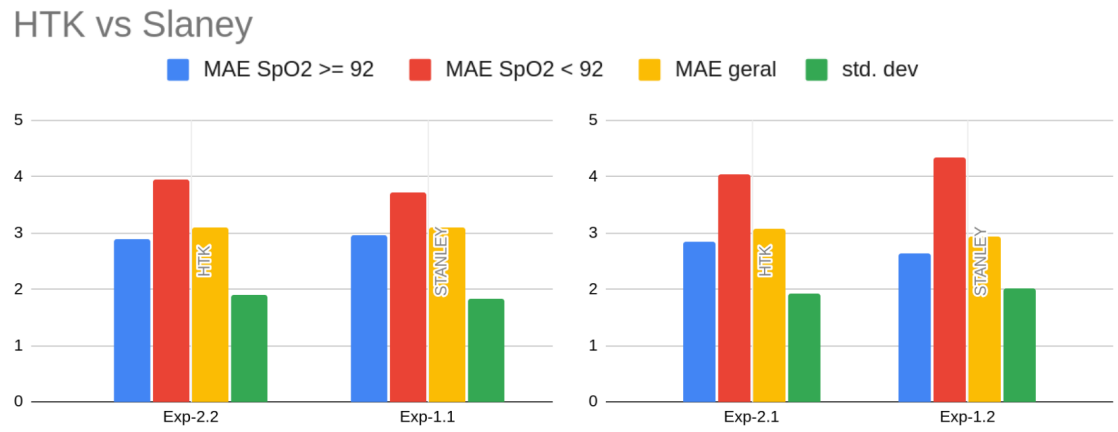


Figura 3.5: *Gráfico com erros absolutos do teste para a implementação HTK e Slaney da escala mel.*

- ***Window length***: tamanho da janela da FFT

  Observou-se que diminuir o tamanho das janelas reduziu o desvio padrão e erro para amostras com SpO_2 < 92%, mas levemente aumentou para as demais, tendo em média nenhum efeito no desempenho, como mostrado na figura 3.6.

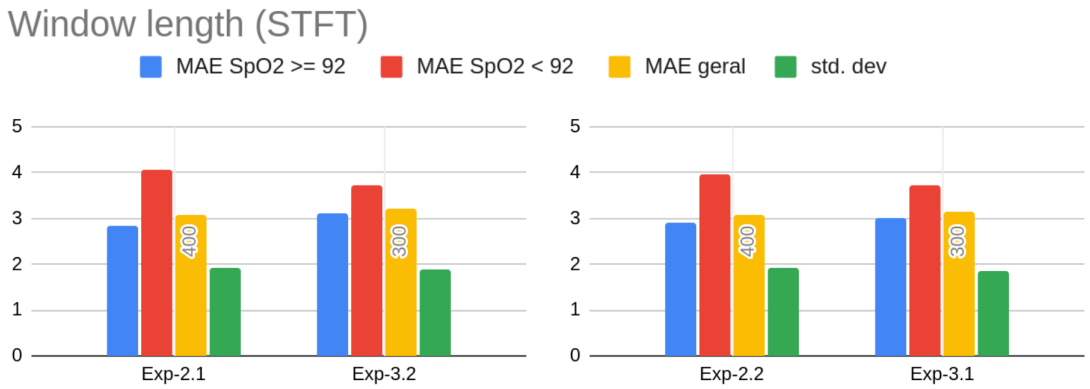


Figura 3.6: *Gráfico com erros absolutos do teste para diferentes tamanhos de janela.*

- **Hop length**: número de *frames* entre duas janelas consecutivas

  Não parece ter relação direta com o desempenho do modelo, como mostrado na figura 3.7. Há casos em que um menor *hop length* diminuiu o erro, mas há outros em que o erro aumentou; interpretamos isso como um comportamento aleatório.

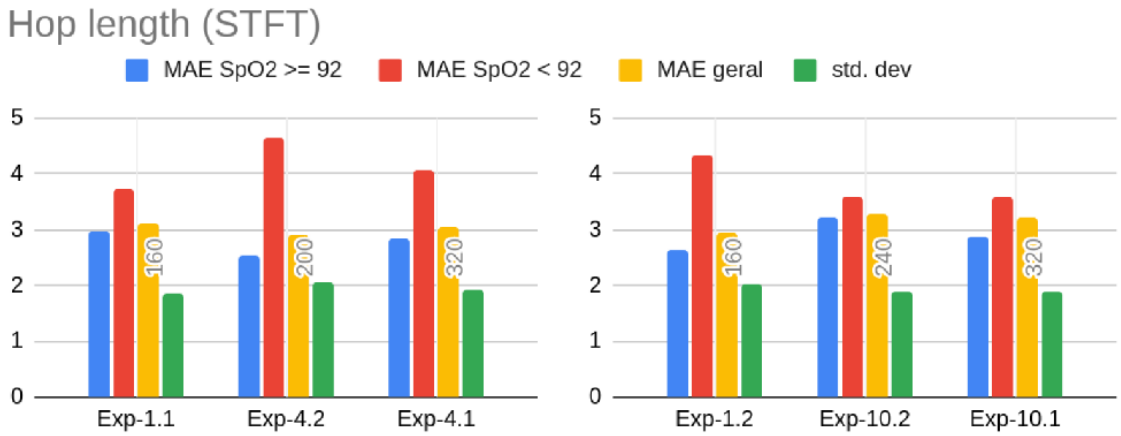


Figura 3.7: *Gráfico com erros absolutos do teste para diferentes valores de* hop length.

- **Tamanho da janela da FFT após o *padding***

  Nos testes realizados, menores tamanhos de janela diminuíram o erro para $SpO_2 <$ 92%. Porém em média o erro foi estável em torno de 3, como visto na figura 3.8.

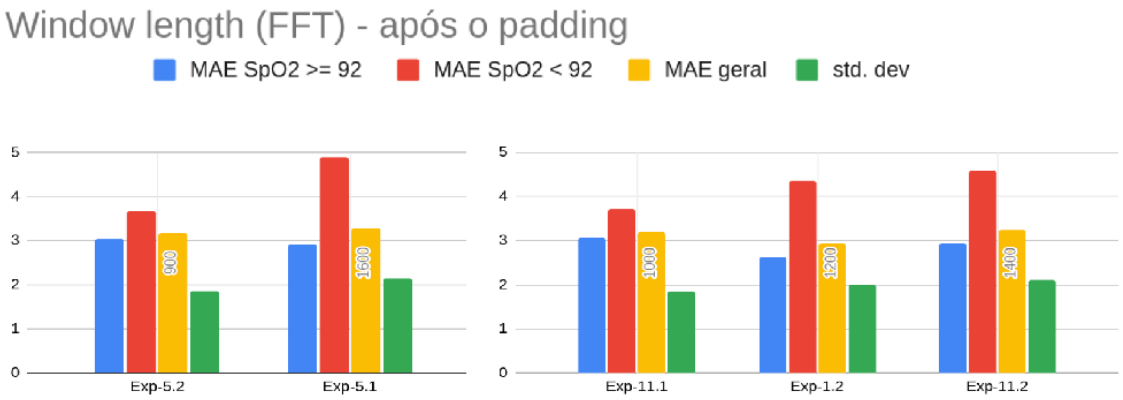


Figura 3.8: *Gráfico com erros absolutos do teste para diferentes valores de* window length *(após* padding*).*

- **Quantidade de MFCC**

  O erro médio sofreu pouca variação, exceto para 50 MFCCs. A quantidade ideal parece ser em torno de 30 e 40, como sugere a figura 3.9.

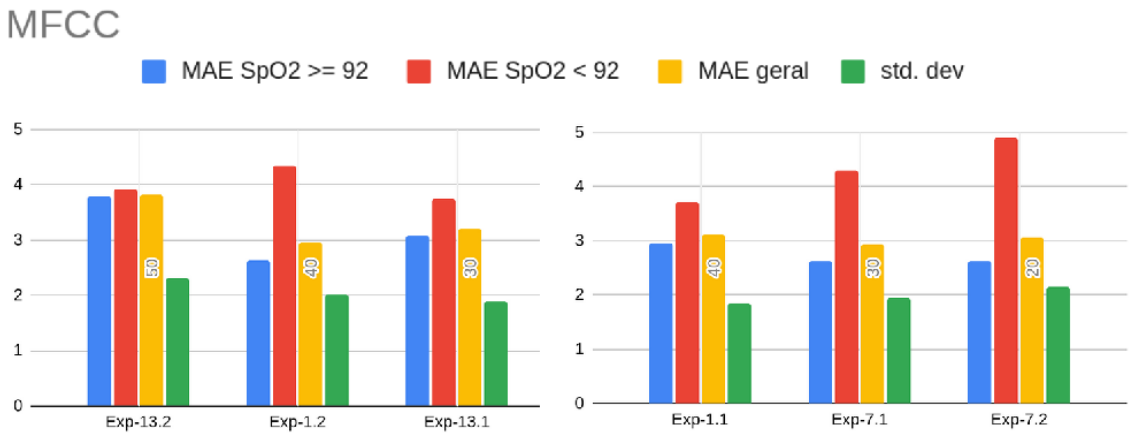


Figura 3.9: *Gráfico com erros absolutos do teste para diferentes quantidades de MFCC.*

- **Quantidade de *mel bands***

  Na figura 3.10 vemos também que o número de *mel bands* afeta pouco o desempenho em média, mas que 30 não foi um bom valor. Nos testes com 35 e 40 *mel bands* obtivemos resultados similares.

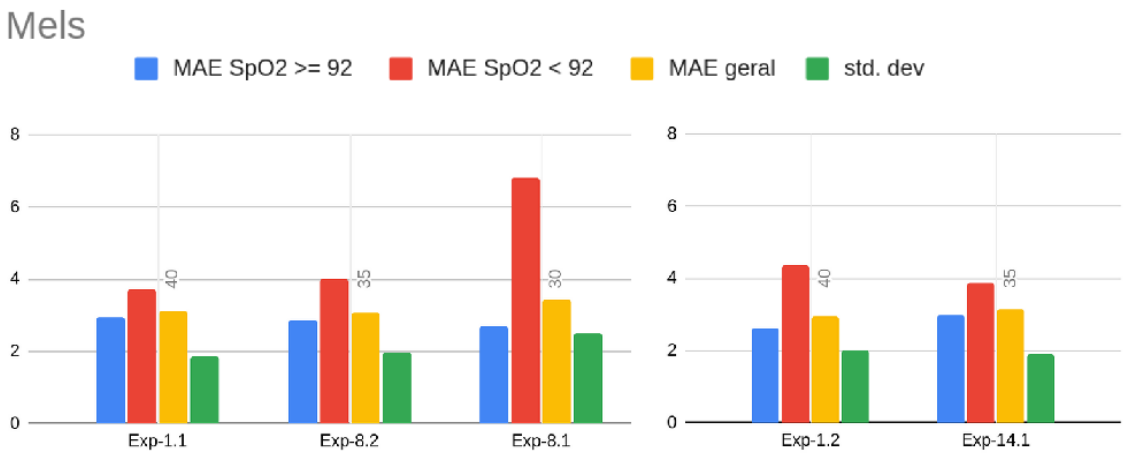


Figura 3.10: *Gráfico com erros absolutos do teste para diferentes quantidades de mel* bands.

Adição de ruídos

Uma das possíveis formas de viés em modelos que tem como entrada representações de áudio são os ruídos. Em nosso caso, estes seriam ruídos hospitalares, como tosses de outros pacientes ou sons de aparelhos presentes na UTI, que o nosso modelo poderia atribuir uma relação com os níveis de SpO_2 do paciente.

Para avaliar se nosso modelo possui este viés, nós realizamos experimentos inserindo ruído hospitalar apenas nas amostras de teste, como mostrado na figura 3.11.

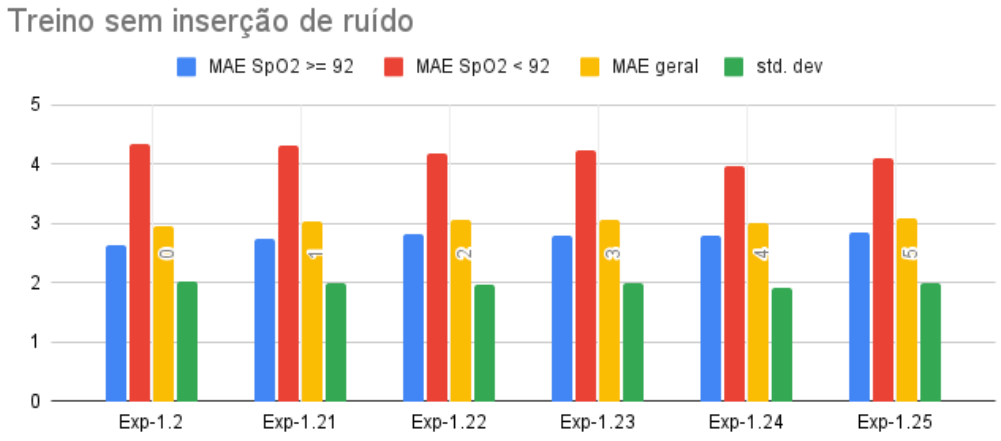


Figura 3.11: *Gráficos do erro absoluto do modelo e configuração de áudio para diferentes quantidades faixas de ruído inseridas no teste*

O gráfico mostra, da esquerda para a direita, o erro nas amostras de teste com inserção de 0 até 5 faixas de ruídos. Vemos que o erro médio absoluto está por volta de 3 em todos experimentos, mas que o erro absoluto para as amostras de $SpO_2 < 92\%$ está levemente decaindo, enquanto que para $SpO_2 \geq 92\%$ está aumentando. Há uma sugestão de um leve viés e de que o ruído está sendo associado com baixos níveis de SpO_2.

Uma forma de tentar mitigar esse viés é o treinamento da rede com inserção de ruídos nas amostras. Então foram realizados experimentos com inserção de ruído para diferentes configurações de áudio. Abaixo veremos o comportamento do nosso modelo e configuração de áudio escolhido para essa inserção.

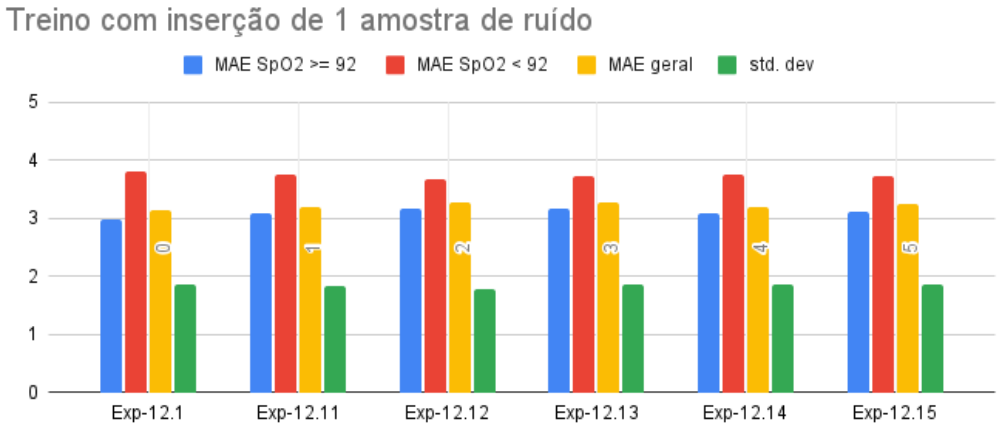


Figura 3.12: *Gráfico com os erros absolutos das amostras de teste para o modelo treinado com inserção de 1 amostra de ruído.*

Na figura 3.12 observa-se que em relação ao modelo treinado sem ruído, houve uma queda no erro absoluto para $SpO_2 < 92\%$ e aumento para $SpO_2 \geq 92\%$, mas o erro médio é levemente maior, o que é esperado, devido ao desbalanceamento do *dataset*.

Esse comportamento se repetiu com todos os demais experimentos, sendo bastante inconclusivo sobre a melhora ou não a respeito do viés, então tentou-se analisar os erros relativos.

Primeiro, vamos ilustrar o erro relativo do modelo sem ruído no treino, mostrado na tabela 3.5. Observa-se em média um erro de 1,78%.

Experimento	Ruído no teste	Erro relativo médio*	Mediana*	Desvio padrão*
Exp-1.2	0	0.0162	0.026335	0.0359975
Exp-1.21	1	0.0171325	0.0278225	0.0359575
Exp-1.22	2	0.0180225	0.02809	0.03601
Exp-1.23	3	0.018415	0.0286875	0.03569
Exp-1.24	4	0.0187875	0.02872	0.0352475
Exp-1.25	5	0.018545	0.0294825	0.0353375

Tabela 3.5: *Tabela dos erros relativos e desvio padrão dos testes para treinamento sem inserção de ruído. (*) Esses valores são a média de 5 repetições do experimento.*

Agora comparando com a inserção de ruídos no treino, temos para uma amostra a tabela 3.6.

Experimento	Ruído no teste	Erro relativo médio*	Mediana*	Desvio padrão*
Exp-12.1	0	0.016882	0.021303	0.035426
Exp-12.11	1	0.017954	0.027448	0.0355281
Exp-12.12	2	0.01859	0.02998	0.035486
Exp-12.13	3	0.018384	0.029048	0.03526
Exp-12.14	4	0.019554	0.032276	0.036312
Exp-12.15	5	0.018956	0.028964	0.035422

Tabela 3.6: *Tabela dos erros relativos e desvio padrão dos testes para treinamento com inserção de 1 amostra de ruído. (*) Esses valores são a média de 5 repetições do experimento.*

Pode-se notar que agora o erro relativo médio é de aproximadamente 1,8%, sugerindo que a adição de ruídos ainda está sendo associada com baixos níveis de SpO_2. Como o erro relativo é da forma $\frac{SpO2-\hat{SpO2}}{SpO2}$ vemos que a rede está predizendo valores mais baixos que o real.

Para a inserção de duas amostras de ruído nos testes, mostrado na figura 3.13 e na tabela 3.7, apesar do erro absoluto ter se mantido próximo a 3 (como no treinamento sem ruído), temos que o erro relativo médio aumentou para 1,99%.

Experimento	Ruído no teste	Erro relativo médio*	Mediana*	Desvio padrão*
Exp-12.2	0	0.01979	0.031055	0.0357525
Exp-12.21	1	0.020775	0.0314825	0.0367725
Exp-12.22	2	0.021005	0.031235	0.0355325
Exp-12.23	3	0.0191625	0.0300875	0.035675
Exp-12.24	4	0.0197225	0.0298625	0.035465
Exp-12.25	5	0.0190325	0.029405	0.02332

Tabela 3.7: *Tabela dos erros relativos e desvio padrão dos testes para treinamento com inserção de 2 amostra de ruído. (*) Esses valores são a média de 5 repetições do experimento.*

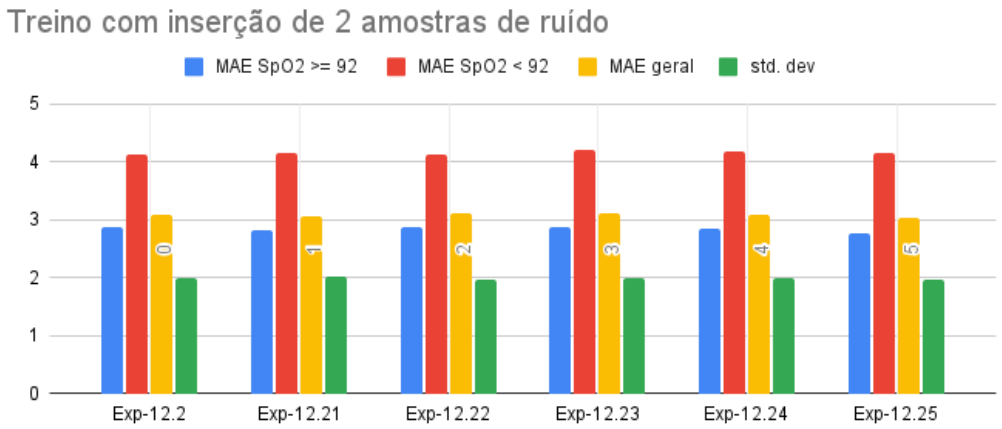


Figura 3.13: *Gráfico com os erros absolutos das amostras de teste para o modelo treinado com inserção de 2 amostras de ruído.*

Com 3 amostras de ruído inseridas em cada amostra do teste temos erro relativo de 1,97%, como mostrado na tabela 3.8 e figura 3.14.

Experimento	Ruído no teste	Erro relativo médio*	Mediana*	Desvio padrão*
Exp-12.3	0	0.01931	0.03081	0.035415
Exp-12.3	1	0.019125	0.02938	0.035455
Exp-12.3	2	0.019745	0.03134	0.03584
Exp-12.3	3	0.020015	0.031175	0.0357075
Exp-12.3	4	0.02	0.031775	0.0358125
Exp-12.3	5	0.02046	0.0327075	0.0357975

Tabela 3.8: *Tabela dos erros relativos e desvio padrão dos testes para treinamento com inserção de 3 amostra de ruído. (*) Esses valores são a média de 5 repetições do experimento.*

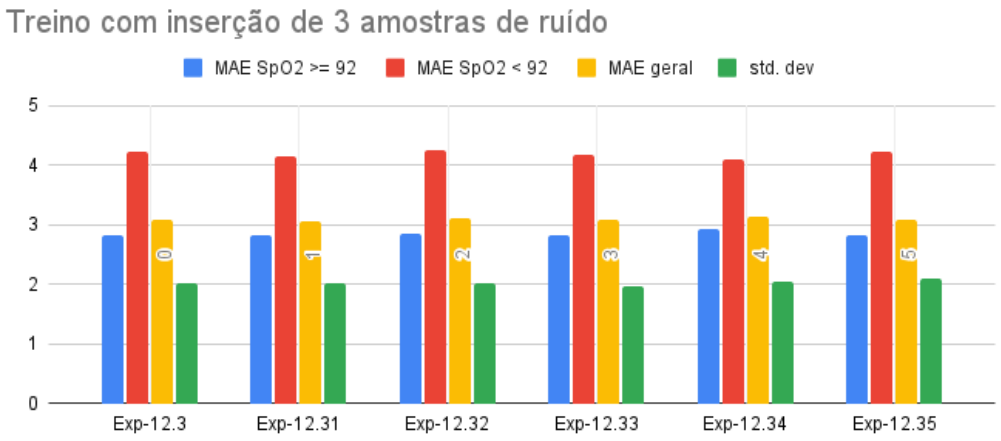


Figura 3.14: *Gráfico com os erros absolutos das amostras de teste para o modelo treinado com inserção de 3 amostras de ruído.*

O mesmo se repetiu para as demais inserções de ruído, ocorrendo o aumento do erro relativo apesar de em termos do erro médio permanecer igual. Concluímos então que a adição de ruídos acabou criando um viés na rede, que agora em média prevê abaixo do nível real de saturação de oxigênio no sangue.

3.1.2 Performance

No.	Learning Rate	Loss_fn	Scheduler	diff mean	diff median	diff std dev	avg sdtdev	epoch
22	same as 1 but fc1=50			1.9707	1.8938	0.3217	1.8388	1000
1	1E-03	MSE	Noam	2.1953	2.1002	0.5162	1.8718	1000
21	same as 1 but fc1=200			2.3166	2.3507	0.0999	1.9394	1000
24	same as 1 but optimizer=RMS			2.3617	2.4989	0.4368	1.8980	1000

Tabela 3.9: *Experimentos de melhor performance*

A tabela 3.9 mostra os resultados que trouxeram melhores resultados. As linha roxas indicam experimentos muito próximos do experimento da linha azul, com pequenas modificações.

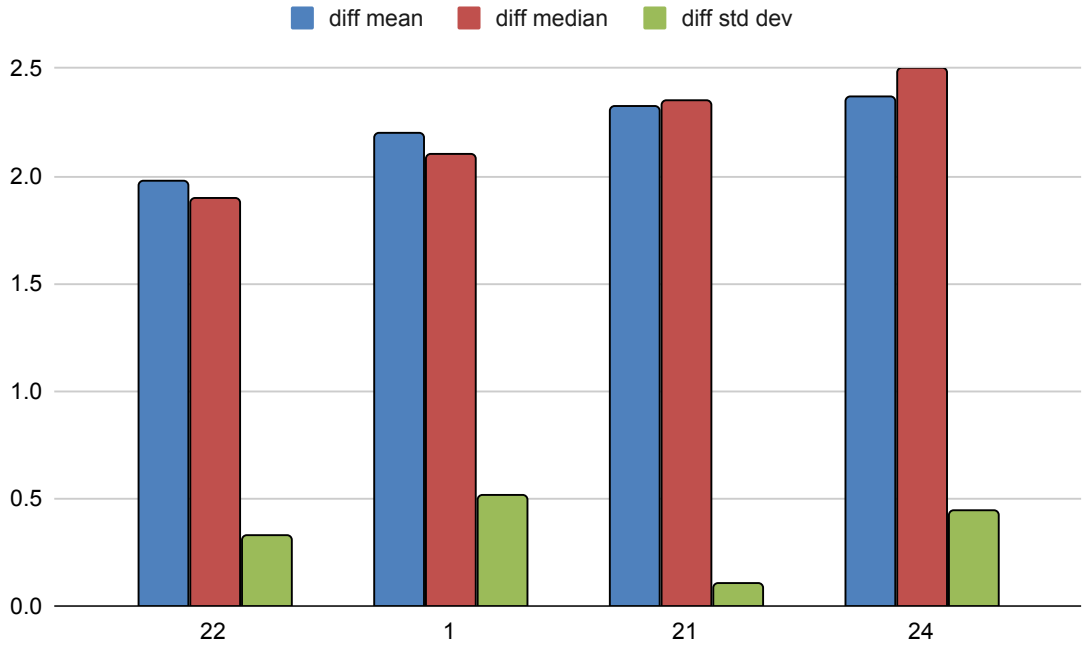


Figura 3.15: *Ilustração da Tabela 3.9*

Nossas melhores respostas estão estimando ao redor de dois pontos do valor real de SpO_2. Esta afirmação, entretanto, ainda é bastante analítica e de difícil interpretação prática. Vamos dedicar as próxima seções à uma análise mais detalhada dos dados, buscando ilustrar a performance de nossa Rede para seu principal fim clínico.

3.2 Classificação

Um dos principais usos práticos da medida da saturação de oxigênio no sangue é a determinação de insuficiência respiratória. Portanto, achamos necessário observar a performance do modelo para a tarefa de classificação. Aqui, a classe positiva representa presença de insuficiência, utilizando como limiar o valor de 92% para SpO_2, ou seja, todos os pacientes com valor de saturação de oxigênio abaixo ou iguais a 92% estão nessa categoria. Enquanto que a não presença da insuficiência são todos os demais casos ($SpO_2 > 92\%$).

Seja y_i o valor real correspondente a entrada x_i e $\hat{y}_i$ uma previsão para y_i.

Os termos TP, TN, FP e FN que aparecem a frente são respectivamente:

- **True Positive (TP):** $y_i, \hat{y}_i > 92\%$
- **True Negative (TN):** $y_i, \hat{y}_i \leq 92\%$
- **False Positive (FP):** $y_i > 92\%$ e $\hat{y}_i \leq 92\%$

- **False Negative (FN):** $y_i \leq 92\%$ e $\hat{y}_i > 92\%$

As definição das métricas utilizadas encontram-se descritas na subseção 1.7.5. Na tabela, usou-se um esquema de cores gradiente de branco até o verde escuro indicando a variação dentro do intervalo $[0, 1]$. Notar também que $TP + TN + FP + FN = 1$ para cada experimento.

3.2.1 Resultados

Experimento	TP	TN	FP	FN	Acurácia	F1 Score	MCC	sensitividade	especifidade	precisão
22	0.08	0.74	0.08	0.10	0.82	0.47	0.38	0.446	0.906	0.528
24	0.08	0.69	0.13	0.10	0.77	0.43	0.31	0.456	0.843	0.458
1	0.08	0.68	0.13	0.10	0.77	0.42	0.29	0.446	0.840	0.436
21	0.07	0.69	0.13	0.12	0.76	0.32	0.20	0.374	0.844	0.285
26	0.110	0.489	0.326	0.075	0.60	0.37	0.17	0.595	0.600	0.288

Tabela 3.10: *Matriz de confusão de experimentos por melhor acurácia*

A tabela mostra os cinco melhores experimentos, ordenados por maior acurácia. Das métricas mostradas, consideramos mais relevantes a Acurácia, o *F1 Score* e a *MCC*. Destes cinco, escolhemos os quatro melhores para criar um gráfico que resume os resultados obtidos com as métricas escolhidas.

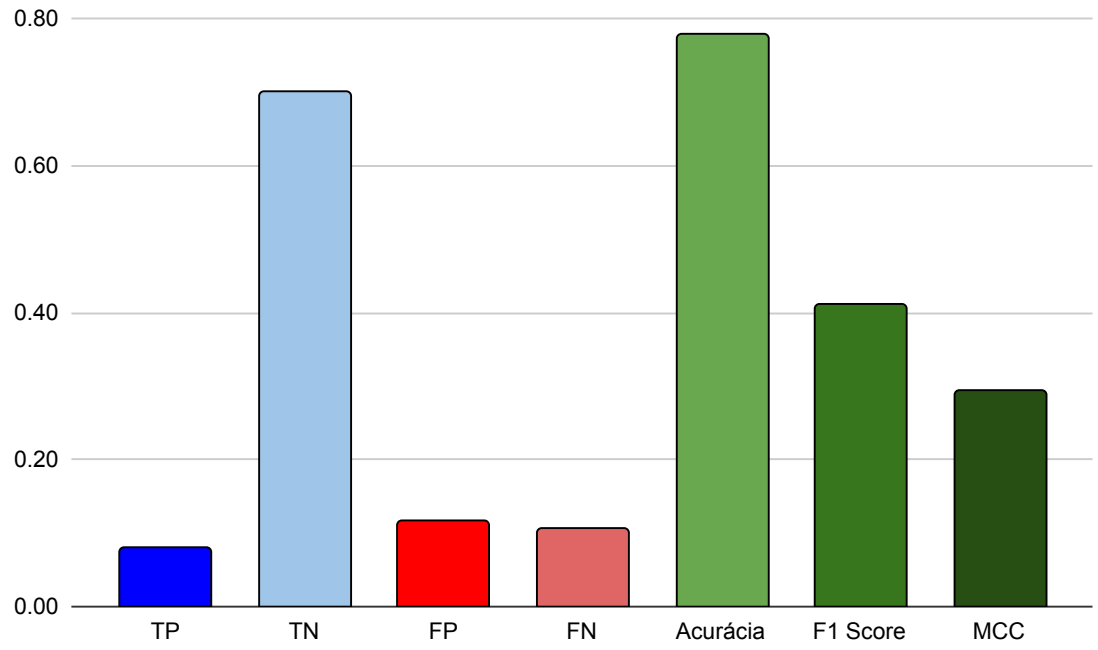


Figura 3.16: *Ilustração da Tabela 3.10*

A acurácia de nossa rede foi ao redor de 0.8 (ou 80%). Já era esperado que esta não seria tão boa quanto o trabalho original do grupo SPIRA (com acurácia de 91,66%) pois este lidava com uma comparação binária enquanto que nosso problema é de regressão. A *F1 Score* mostra que nosso modelo tem um problema na captura de casos positivos e na

acurácia destas capturas. Por fim temos a MCC, indicando que nosso modelo faz mais do que adivinhar valores e é capaz de estimar SpO_2 com relativa confiabilidade.

3.3 Análise do Erro Relativo

Agora vamos analisar os resultados do ponto de vista do erro relativo. A fórmula para obter o erro relativo se encontra na subseção 1.7.5 do capítulo de noções preliminares.

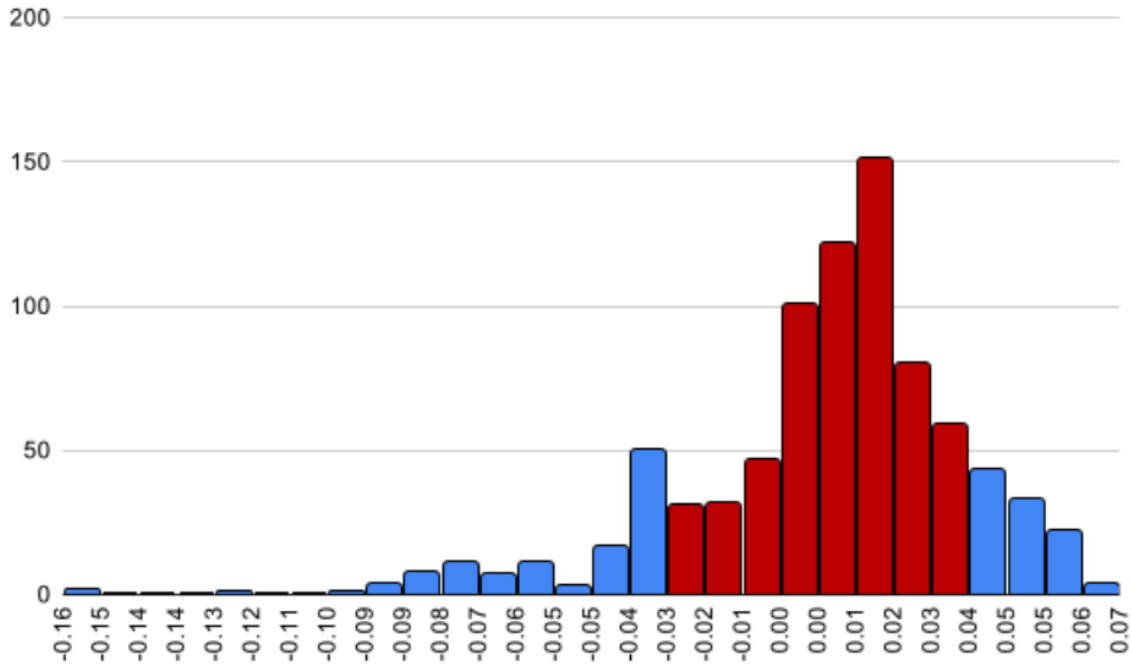


Figura 3.17: *Histograma dos erros relativos dos melhores experimentos*

Uma análise dos dados exemplificados no histograma mostra que 80% das estimativas estão no intervalo $[-3.34\%; 4.01\%]$ de erro relativo (representado pelas colunas em vermelho). Também é possível notar um viés do modelo de estimar valores acima da saturação real do que abaixo, fato esse que é esperado, já que os dados utilizados no projeto se concentram em altos valores de SpO_2.

Considerando a importância do valor de 92% para a SpO_2, já que esta é a que diferencia um paciente de saudável de um doente, e do fato de que nossos melhores experimentos estão ao redor de dois pontos do valor real, seria interessante analisar a taxa de acerto de nosso modelo nos intervalos ao redor de ± 2 pontos de 92%.

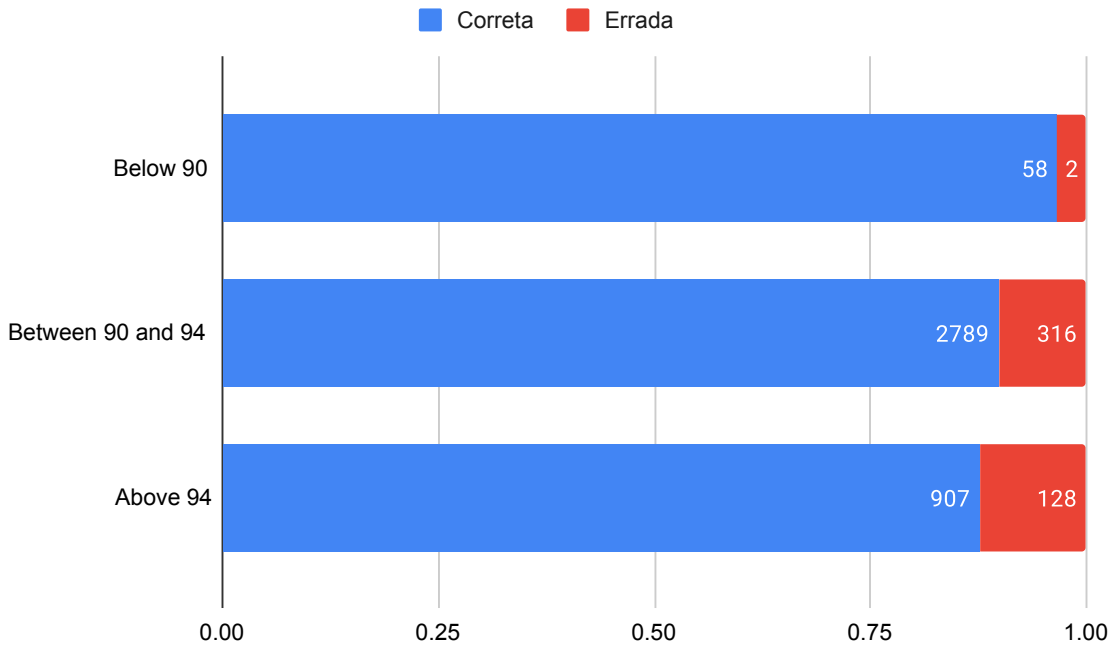


Figura 3.18: *Resultado das estimativas, separadas em faixas*

No gráfico 3.18, estão dispostas as taxas de acerto agrupadas pelos valores previstos. Assumindo $\hat{S}pO2$ como o valor estimado de SpO_2 pela nossa rede, temos uma estimativa de Insuficiência Respiratória confiável, ao redor de 96,7%, para $\hat{S}pO2 < 90$ e confiança de 89% para $90 \leq \hat{S}pO2 \leq 94$. Estas são estimativas comparáveis com as previsões por classificação com CNNs do trabalho anterior do grupo SPIRA de acurácia 91%.

De um ponto de vista clínico, este gráfico também ilustra que estimativas abaixo de 90% são mais confiáveis do que estimativas acima de 94%. Entretanto, seriam necessários mais dados para que possamos confirmar estas conclusões.

Capítulo 4

Considerações finais

Primeiramente, foi possível determinar diversos hiper-parâmetros que tiveram boa performance e também hiper-parâmetros que não afetam significativamente o desempenho da rede, como os relacionados com a MFCC.

Dentre nossos testes, a configuração seguinte foi uma das que teve melhor eficiência:

- *Learning Rate*: 10^{-3}
- *Loss Function*: MSE
- Uso de *Scheduler* Noam com 10 passos de aquecimento
- Audio Estéreo
- Não adição de ruído
- MFCCs como *features* de entrada
- Hiper-parâmetros para MFCC:
  - Implementação *Slaney* da escala mel
  - Janela de 400 frames
  - *Hop length* de 160 frames
  - Janela de 1200 amostras para a FFT
  - 40 MFCC
  - 40 *Mel Bands*
- *batches* de tamanho 5

Nossos melhores resultados estão ao redor de dois pontos do valor real de SpO_2, que pode ser traduzido em uma variação em erro relativo ao redor de 3% a 4% ao redor do valor real. Se não incluirmos as observações práticas, este seria um dos resultados finais de nossos experimentos com o nosso modelo otimizado para os dados disponíveis.

Em relação aos testes relacionados à performance como ferramenta de diagnóstico de insuficiência respiratória, obtivemos, de forma geral, uma acurácia de 80%, F1-Score de 0.44 e MCC ao redor de 0.35. Com estes resultados da matriz de confusão vemos que nosso modelo tem dificuldades na classificação de pacientes com IR, o que é um *trade-off* que foi esperado acontecer ao trabalharmos com um problema de regressão ao invés de classificação binária.

Uma análise interessante foi comparar nossa acurácia nos intervalos de SpO_2 $[0, 90[$, $[90, 94[$ e $[94, 100]$, onde obtivemos uma acurácia de previsão próxima de 90% para valores abaixo de 94%. Isto quer dizer que **para valores abaixo de 94% fomos capazes de atingir uma acurácia próxima daquela alcançada por modelos de classificação binária**.

4.1 Desafios enfrentados

As maiores dificuldades enfrentadas vieram das tentativas de compreender a base teórica por trás do trabalho, como nas etapas de pré-processamento, modelagem da rede neural, implementações, forma de lidar com os hiper-parâmetros e de como avaliar o desempenho do modelo. Estes desafios levaram a um processo de aprendizado contínuo até o final do trabalho.

O fato da rede neural ser basicamente uma caixa preta nos causou dúvidas sobre a credibilidade do nosso trabalho. Tivemos dificuldade em compreender se estávamos na direção certa e até mesmo quando a rede estava funcionando nos questionamos como e o porquê dos valores obtidos. A insegurança só diminuiu quando pudemos analisar a performance dos resultados dos experimentos.

Podemos dizer também que a elaboração do pôster, da apresentação e da monografia nos deram uma visão ampla do nosso projeto e pudemos ver através deles o nosso progresso ao longo do ano, onde foi possível reunir todo o conhecimento adquirido e atribuir significado ao processo.

4.2 Próximos passos

É fundamental que haja mais dados, principalmente de pacientes com insuficiência respiratória. Assim espera-se poder aprimorar o desempenho da rede neural para ter uma melhor confiabilidade e levar à aplicações práticas no mundo real, com a criação de um biomarcador vocal de fato.

Também seria interessante ver aplicações deste modelo à outros problemas. Na época em que este documento foi escrito, existiam discussões no grupo SPIRA sobre dois temas interessantes: a detecção de níveis de monóxido de carbono emitido por fumantes através da voz e o acompanhamento de asma crônica em pacientes que fumam por um longo período de tempo. Com algumas modificações, talvez também pudéssemos realizar experimentos nestas áreas com nosso modelo.

Junto com as conclusões obtidas em nosso experimento, lançamos um convite a alunos interessados em continuar o desenvolvimento e pesquisa nesta área. Acreditamos que existe potencial para uso no mundo real deste tipo de trabalho, com a possível criação de aplicações de baixo custo, acessibilidade e confiabilidade.

Referências

[Aghdam e Heravi 2017] Hamed Habibi Aghdam e Elnaz Jahani Heravi. *Guide to convolutional neural networks : a practical application to traffic-sign detection and classification*. Springer, 2017 (citado na pg. 17).

[Brilliant.org 2022] Brilliant.org. *Backpropagation*. 2022. url: https://brilliant.org/wiki/backpropagation/ (citado na pg. 15).

[Casanova *et al.* 2021] Edresson Casanova *et al.* "Deep learning against covid-19: respiratory insufficiency detection in brazilian portuguese speech". Em: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 2021, pgs. 625–633 (citado nas pgs. 1, 23, 27).

[Chicco e Jurman 2020] Davide Chicco e Giuseppe Jurman. "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation". Em: *BMC Genomics* 21.1 (jan. de 2020). doi: 10.1186/s12864-019-6413-7 (citado na pg. 21).

[Despotovic *et al.* 2021] Vladimir Despotovic, Muhannad Ismael, Maël Cornil, Roderick Mc Call e Guy Fagherazzi. "Detection of covid-19 from voice, cough and breathing patterns: dataset and preliminary results". Em: *Computers in Biology and Medicine* 138 (nov. de 2021), pg. 104944. doi: 10.1016/j.compbiomed.2021.104944 (citado na pg. 3).

[Fayek 2022] Haytham Fayek. *Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between*. 2022. url: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html (citado na pg. 10).

[Gordillo 2014] Christian Dayan Arcos Gordillo. "Reconhecimento de voz contínua combinando os atributos MFCC e PNCC com métodos de robustez SS, WD, MAP E FRN". Tese de dout. Jun. de 2014. url: https://www.maxwell.vrac.puc-rio.br/colecao.php?strSecao=resultado&nrSeq=23090@1 (citado na pg. 10).

[Krizhevsky *et al.* 2017] Alex Krizhevsky, Ilya Sutskever e Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". Em: *Communications of the ACM Vol 50*. 2017, pgs. 84–90 (citado na pg. 16).

[McCulloch e Pitts 1943] Warren S. McCulloch e Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". Em: *Bulletin of Mathematical Biophysics Vol 5* (1943), pgs. 115–133. URL: http://comedudb6.knue.ac.kr/tykim/Myhome/AInote/MCP1943.pdf (citado na pg. 12).

[Misra e Diganta 2019] Misra e Diganta. *Mish: A Self Regularized Non-Monotonic Activation Function*. 2019. DOI: 10.48550/ARXIV.1908.08681. URL: https://arxiv.org/abs/1908.08681 (citado na pg. 18).

[Mundt *et al.* 2012] James C. Mundt, Adam P. Vogel, Douglas E. Feltner e William R. Lenderking. "Vocal acoustic biomarkers of depression severity and treatment response". Em: *Biological Psychiatry* 72.7 (out. de 2012), pgs. 580–587. DOI: 10.1016/j.biopsych.2012.03.015 (citado na pg. 3).

[Network 2016] Pacific Northwest Seismic Network. *What is a Spectrogram?* 2016. URL: https://pnsn.org/spectrograms/what-is-a-spectrogram (citado na pg. 7).

[Olah 2014] Christopher Olah. *Neural Networks, Manifolds, and Topology*. 2014. URL: https://colah.github.io/posts/2014-03-NN-Manifolds-Topology/ (citado na pg. 15).

[Ramaseshan 2013] Ajay Ramaseshan. "Application of Multiway Methods for Dimensionality Reduction to Music". Tese de dout. Dez. de 2013 (citado na pg. 9).

[Rosenblatt 1962] Frank Rosenblatt. *Principles of Neurodynamics*. Spartan Books, 1962 (citado na pg. 12).

[Saúde - SUS 2020] Ministério da Saúde - SUS. *Protocolo de Manejo Clínico da COVID-19 na Atenção Especializada*. 2020. URL: https://pesquisa.bvsalud.org/bvsms/resource/pt/biblio-1096764 (citado na pg. 1).

[Smyth 2019] Tamara Smyth. *Music 175: Pitch II*. 2019. URL: http://musicweb.ucsd.edu/~trsmyth/pitch2/pitch2.pdf (citado na pg. 9).

[Stowell e Plumbley 2014] Dan Stowell e Mark D. Plumbley. *Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning*. 2014. URL: https://peerj.com/articles/488/ (citado na pg. 4).

[Tracy *et al.* 2020] John M. Tracy, Yasin Özkanca, David C. Atkins e Reza Hosseini Ghomi. "Investigating voice as a biomarker: deep phenotyping methods for early detection of parkinson's disease". Em: *Journal of Biomedical Informatics* 104 (abr. de 2020), pg. 103362. DOI: 10.1016/j.jbi.2019.103362. URL: https://www.sciencedirect.com/science/article/pii/S1532046419302825 (citado na pg. 3).

[Zhang *et al.* 2018] Wei Zhang, Chuanhao Li, Gaoliang Peng, Yuanhang Chen e Zhujun Zhang. "A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load". Em: *Mechanical Systems and Signal Processing* (2018), pgs. 439–453. URL: https://www.sciencedirect.com/science/article/pii/S0888327017303369 (citado na pg. 27).