UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM DE SISTEMAS COMPLEXOS

MILTON DOS SANTOS

Classificação de áudio musical a partir dos coeficientes da Transformada Wavelet utilizando Redes Neurais Convolucionais

São Paulo
2022

MILTON DOS SANTOS

Classificação de áudio musical a partir dos coeficientes da Transformada Wavelet utilizando Redes Neurais Convolucionais

Versão corrigida

Dissertação apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Modelagem de Sistemas Complexos.

Área de concentração: Sistemas Complexos

Orientador: Prof. Dr. Camilo Rodrigues Neto

São Paulo

2022

Ficha catalográfica elaborada pela Biblioteca da Escola de Artes, Ciências e Humanidades,
com os dados inseridos pelo(a) autor(a)
Brenda Fontes Malheiros de Castro CRB 8-7012; Sandra Tokarevicz CRB 8-4936

Santos, Milton dos
Classificação de áudio musical a partir dos coeficientes da Transformada Wavelet utilizando Redes Neurais Convolucionais / Milton dos Santos; orientador, Camilo Rodrigues Neto. -- São Paulo, 2022.
115 p: il.

Dissertacao (Mestrado em Ciencias) - Programa de Pós-Graduação em Modelagem de Sistemas Complexos, Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, 2022.
Versão corrigida

1. Processamento de Sinais. 2. MIR. 3. Transformada Wavelet. 4. Coeficientes Wavelet. 5. Rede Neural Convolucional. I. Rodrigues Neto, Camilo, orient. II. Título.

Dissertação de autoria de Milton dos Santos, sob o título **"Classificação de áudio musical a partir dos coeficientes da Transformada Wavelet utilizando Redes Neurais Convolucionais"**, apresentada à Escola de Artes, Ciências e Humanidades da Universidade de São Paulo, para obtenção do título de Mestre em Ciências pelo Programa de Pós-graduação em Modelagem de Sistemas Complexos, na área de concentração Sistemas Complexos, aprovada em _____ de _____ de _____ pela comissão julgadora constituída pelos doutores:

_____

Prof. Dr. _____

Instituição: _____

Presidente

_____

Prof. Dr. _____

Instituição: _____

_____

Prof. Dr. _____

Instituição: _____

_____

Prof. Dr. _____

Instituição: _____

Dedico à minha mãe, Maria Lúcia dos Santos e à memória de meu pai, Cícero Romão de Almeida.

Agradecimentos

Agradeço à minha amada esposa Isabella Tavares que me apoiou durante esta jornada e, contudo, ainda nos presenteou com a nossa maior riqueza, a nossa filha Beatriz Tavares.

Ao meu orientador, Camilo Rodrigues Neto, que com sabedoria e paciência me auxiliou nesta fase de minha vida. Tenho certeza que sem o seu apoio eu não teria conseguido.

Aos meus irmãos, Paulo Santos de Almeida e Silvio Santos que me apoiaram desde o início nesta jornada com sábios conselhos e incentivos.

Aos colegas que obtive no programa de pós graduação em Modelagem de Sistemas Complexos, Murilo Mazzotti Silvestrini, Lam Chong Hang Lee, Fábio Reale, Guilherme Giovanini e aos queridos Ludmila Deute e Luiz Amato Neto que nos deixaram cedo demais.

Aos demais professores do Programa de Pós Graduação em Modelagem de Sistemas Complexos, pela dedicação ao ensino de excelente qualidade e por me ajudarem a visualizar os problemas de uma forma que eu consiga apresentar soluções eficientes.

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

"Tudo o que é possível exige existir."

"Omne possibile exigit existere."

(Gottfried Wilhelm Leibniz)

Resumo

SANTOS, Milton dos. **Classificação de áudio musical a partir dos coeficientes da Transformada Wavelet utilizando Redes Neurais Convolucionais**. 2022. 114 f. Dissertação (Mestrado em Ciências) – Escola de Artes, Ciências e Humanidades, Universidade de São Paulo, São Paulo, 2022.

A identificação do estilo musical a que pertence uma música é uma tarefa relativamente simples para um humano, mesmo com pouco treinamento musical. Entretanto, é uma tarefa bastante difícil de ser realizada de forma automatizada. Neste trabalho utilizamos a Transformada Wavelet, que consegue representar uma música em suas componentes de frequência em função do tempo, gerando uma imagem denominada espectrograma. A partir do espectrograma, geramos imagens para treinar uma Rede Neural Convolucional com o objetivo de classificar os sinais de áudio em seus estilos musicais. Apenas os primeiros 15 segundos de cada música são utilizados para gerar o espectrograma, 6.075 músicas no conjunto de treinamento e 2.025 no conjunto de teste, pertencentes a 10 estilos musicais – Blues, Clássico, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae e Rock. O procedimento é repetido 10 vezes, com o conjunto de treinamento e teste escolhidos aleatoriamente. A média das taxas de acerto ficou entre 70% e 94%, bem acima dos 10% esperados se a classificação fosse por puro acaso.

Palavras-chaves: Processamento de Sinais. MIR. Transformada Wavelet. Coeficientes Wavelet. Rede Neural Convolucional.

Abstract

SANTOS, Milton dos. **Classification of musical audio from the coefficients of the Wavelet Transform using Convolutional Neural Networks**. 2022. 114 f. Dissertation (Master of Science) – School of Arts, Sciences and Humanities, University of São Paulo, São Paulo, 2022.

Identifying the musical style to which a song belongs is a relatively simple for a human, even with little musical training. However, it is a task quite difficult to be performed in an easy way. In this work we use the Wavelet Transform, which manages to represent a song in its frequency as a function of time, generating an image called spectrogram. From grass, we generate images of the behavior spectrum a Convolutional Neural Network with the purpose of classifying audio signals into their musical styles. only the first 15 seconds of each song used to generate the spectrogram, 6,075 songs in training set and 2025 in the test set, belonging to 10 musical styles – Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae and Rock. The procedure Repetition 10 times, with the training and test set randomly chosen. One average hit rates were between 70% and 94%, well above the 10% expected if the classification were by pure chance.

Keywords: Signal Processing. MIR. Wavelet Transform. Wavelet Coefficient. Convolutional Neural Network

Lista de figuras

Lista de algoritmos

Lista de tabelas

Lista de abreviaturas e siglas

MIR	*Music Information Retrieval*
DNN	Rede Neural Profunda
CNN	Rede Neural Convolucional
LIT	Lineares Invariantes no Tempo
TF	Transformada de Fourier
DTFT	Transformada de Fourier em Tempo Discreto
DFT	Transformada de Fourier Discreta
STFT	Transformada de Fourier de Curta Duração
WT	Transformada Wavelet
cA	Coeficientes Aproximados
cD	Coeficientes Detalhados
SVM	*Support Machine Vector*
LDA	*Linear Discriminat Analysis*
KNN	*K-Nearest Neighbors*
RR	*Round Robin*

Lista de símbolos

$\int$	Símbolo matemático de integral.
$\sum$	Símbolo matemático de somatório.
∞	Símbolo matemático de infinito.
$\in$	Símbolo matemático de pertence.
$:=$	Símbolo de "é definido como".
ϕ	Símbolo de Wavelet Mãe
$< \cdot, \cdot >$	Símbolo de Produto Interno
∂	Derivada Parcial

Sumário

1 Introdução e Justificativa

No campo da Visão Computacional as Redes Neurais Convolucionais (do inglês "Convolutional Neural Network", CNN) tem conseguido destaque por resultados obtidos em diversas tarefas, se tornando uma técnica eficaz para reconhecimento, restauração e geração de imagens entre outras. Porém, a imagem necessariamente não precisa estar associada a uma figura gerada a partir de uma máquina fotográfica ou de um "smartphone", mas também é possível ter imagens criadas a partir de uma série temporal, por exemplo, um sinal de áudio que tenha os seus valores contidos em um vetor e a partir disto criar uma imagem.

Entretanto, a ideia de utilizar uma base de dados com diversas imagens e teinar uma Rede Neural Convolucional para obter resultados com um alto percentual de acerto em sua previsão, não é uma tarefa simples. Há limitações que estão relacionados à quantidade e a qualidade dessas imagens. A quantidade pode interferir na forma como a rede irá aprender com os dados, com pouca quantidade certamente ela terá um pico de aprendizagem generalizando algumas propriedades para as demais imagens. A falta de qualidade na imagem pode causar uma baixa capacidade de discernir as principais características de cada figura que pode causar uma generalização com as demais (DODGE; KARAM, 2016).

O processamento de áudio tem um papel importante para o bom desempenho da análise desse tipo de sinal (referência), havendo uma melhora significativa nos resultados principalmente com a combinação de processamento de sinais e algoritmos modernos de aprendizagem de máquina (Machine Learning) e também com aprendizagem de máquina profunda (Deep Learning).

A etapa que antecede o treinamento do modelo de aprendizagem de máquina é a obtenção de características, considerada uma das mais importantes partes para definir a melhor técnica para o processamento do sinal de áudio, levando-se em conta as informação contidas tanto no domínio do tempo quanto ao da frequência (SHARMA; KARTIKEYAN; KRISHNAN, 2019).

A motivação deste trabalho surge da necessidade de entender como um classificador de áudio se comportaria com as séries temporais desses sinais, com imagens geradas a partir dos coeficientes da transformada Wavelet, utilizando uma neural convolucional com estas figuras e fazer com que o algoritmo conseguisse prever o estilo musical correto. Uma

contribuição inicial deste trabalho seria a utilização da aprendizagem de máquina profunda usando imagens geradas pelos coeficientes Wavelets, ao invés da transformada de Fourier.

1.1 Organização do Trabalho

Este trabalho está dividido nas seguintes partes.

O **Capítulo 2** traz os problemas encontrados nesta área de pesquisa, por exemplo, os tipos de transformadas necessárias para mudar o domínio da função de um sinal $x(t)$ para o domínio da freqüência $X(j\omega)$ e também são apresentadas algumas limitações para este tipo de transformada. A apresentação da hipótese do presente trabalho encontra-se no Capítulo 2.

O **Capítulo 3** descreve a revisão da literatura com as principais referências bibliográficas que fundamentaram a teoria desta pesquisa. A **seção 3.1** traz uma breve introdução sobre Processamento de Sinais, onde são apresentados os conceitos de discretização de um sinal contínuo. A **seção 3.2** e **seção 3.3** destinam-se à explicação do que é Transformada de Fourier, como esta ferramenta matemática atua num sinal contínuo e não períodico no tempo e num sinal discreto infinito e finito. Na **seção 3.4** e na **seção 3.5** são apresentados os conceitos teóricos da Transformada Wavelet, seus coeficientes aproximados e detalhados, e qual é o tipo de Wavelet utilizada para o tratamento de sinais de áudio. Da **seção 3.6** à **seção 3.9** concentramos as definições de Redes Neurais de Aprendizagem Profunda e de Redes Neurais Convolucionais, também são apresentados os métodos de avaliação de uma Rede Neural e qual o método que escolhemos para avaliar a acurácia da Rede Neural Convolucional deste trabalho. Na **seção 3.10** são apresentadas uma breve revisão de alguns trabalhos referentes à previsão de estilo musical.

O **Capítulo 4** é destinado ao método utilizado para o tratamento dos sinais de áudio, como as imagens foram geradas a partir desses sinais, qual o tipo de Rede Neural Convolucional utilizada para ser treinada e avaliada para poder prever o estilo musical de uma nova imagem.

No **Capítulo 5** é descrito os resultados gerais obtidos com o treinamento da Rede Neural Convolucional através dos coeficientes da Tranformada Wavelet. A **seção 5.1** contém os resultados gerais, a **seção 5.2** apresenta os principais resultados obtidos pela validacão da acurácia após o treinamento da Rede e uma comparação entre os resultados

iniciais e finais. A **seção 5.3**, a **seção 5.4** e a **seção 5.5** foram dedicadas para comparar os resultado da Rede Neural Convolucional deste trabalho com outros tipos de configurações de redes neurais e traz à discussão resultados obtidos.

O **Capítulo 6**, apresenta a conclusão e as considerações finais sobre melhorias e trabalhos futuros a partir dos resultados do presente trabalho.

2 Estrutura da Pesquisa

2.1 Problema da Pesquisa

A análise de sinais de áudio consiste em extrair automaticamente caraterísticas de partes da música e, desses fragmentos, rastrear aspectos de tempo, harmonia, melodia, ritmo, assinatura de chave, tipo de instrumento ou o que ainda mais puder ser recuperado no aúdio estudado (Weihs et al., 2019). O campo de pesquisa dedicado à análise de conteúdo de áudio musical é chamado de Recuperação da Informação da Música (MIR), pois contempla tanto a análise do áudio com complicadas melodias quanto da música escrita em partituras na forma digital, este tipo compressão é conhecido como Music Instrument Digital Interface(MIDI) (LERCH, 2012).

A abordagem tradicional para extrair propriedades de um sinal de áudio musical utiliza as técnicas baseadas na Transformada de Fourier que converte um sinal que depende do tempo para uma representação com dependência da frequência. Esta é uma das mais importantes ferramentas em processamento de sinais (MULLER, 2015).

Na abordagem desses métodos, cada propriedade do som precisaria ser analisada, por exemplo, magnitude de potência. A Transformada Wavelet fornece um meio de manipular as escalas em ambos os domínios (tempo e frequência) para encontrar características que poderiam passar despercebidas (SCALVENZI; GUIDO; MARRANGHELLO, 2019).

Considerando os problemas descritos na pesquisa, foi encontrada um meio de classificar cada estilo musical – Blues, Clássico, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae e Rock – a partir apenas dos 15 segundos iniciais, utilizando apenas o espectrograma das Transformadas Wavelet e das Redes Neurais Convolucionais.

2.2 Objetivo geral

Este trabalho buscou explorar a eficiência da Transformada Wavelet em relação a Transformada de Fourier.

2.3 Objetivo específico

O objetivo específico da presente pesquisa inclui a coleta de uma base de dados no formato de áudio, organizada por estilo, extrair a série temporal de cada áudio para serem tratadas pela Transformada Wavelet e depois utilizadas para treinar uma Rede Neural Convolucional.

Esta pesquisa tem a contribuir com o desenvolvimento de técnicas de redes neurais convolucionais na classificação de áudio através de suas imagens.

3 Revisão da Literatura

O presente capítulo descreve as principais referências bibliográficas que formam a base teórica para o desenvolvimento dessa pesquisa e está dividido em cinco partes. A primeira seção seção 3.1 cobre os conceitos teóricos de processamento de sinais digitais e as transformações necessárias para sua análise. A segunda seção seção 3.2 descreve os conceitos da Série e da Transformada de Fourier. A terceira seção seção 3.4 compreende a explicação sobre a Transformada Wavelet. O quarto conjunto de seções, seção 3.6 à seção 3.8, abrange a explanação à aprendizagem de máquina profunda, conhecida como Rede Neural Profunda (do inglês, Deep Neural Network), juntamente com a Rede Neural Convolucional. A quinta seção seção 3.9 finaliza com a avaliação de modelo de Rede Neural Convolucional

3.1 Breve Introdução sobre Processamento de Sinais

Um sinal pode ser definido como uma quantidade física portadora de informação e varia de acordo com o tempo, espaço ou outra variável ou variáveis independente (DOWNEY, 2016).

Estes tipos de sinais são vistos em diversas áreas do conhecimento, por exemplo,os sinais elétricos gerados pelo coração ou pelo cérebro geram tensão elétrica.

Um sistema pode ser formalmente definido como um elemento que mapeia o sinal de entrada $x(t)$ dentro de um sinal de saída $y(t)$, matematicamente como a Equação 1 (RAO, 2018).

$$y(t) = \Re[x(t)], \tag{1}$$

onde, $\Re$ é um operador.

A Figura 1 ilustra um exemplo de um sistema de comunicação, onde, o transmissor envia um sinal da mensagem através de uma canal, isto chega no receptor que é uma estimativa do sinal original da entrada do sistema.

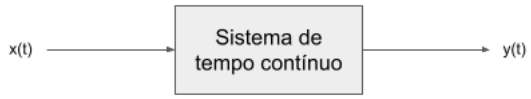
Figura 1 – Elaboração do autor com base em (HAYKIN; VEEN, 2002).

Esses sinais podem ser classificados em categorias que serão descritas a seguir estão restritas ao escopo desta pesquisa (RAO, 2018).

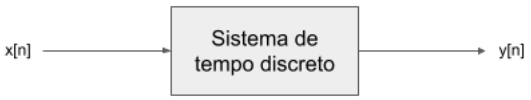
a) **Sinais de Tempo Contínuo e de Tempo Discreto**

O sinal de tempo contínuo tem a amplitude assumindo qualquer valor da variável independente, denotada como $x(t)$ e está ilustrada na Figura 3 (a). O sinal de tempo discreto tem a amplitude assumindo um conjunto de valores no tempo discreto da variável independente, denotada como $x[n]$ e está ilustrada na Figura 3 (b) (SPANIAS; PAINTER; ATTI, 2007).

Um sistema de tempo contínuo tem os seus sinais de entrada em tempo contínuo e são aplicados à saída com tempo contínuo ilustrado na Figura 2 (a). Um sistema de tempo discreto tem os seus sinais de entrada em tempo discreto e são aplicados à saída com tempo discreto ilustrado na Figura 2 (b) (OPPENHEIM; WILLSKY; HAMID, 2010).



(a)

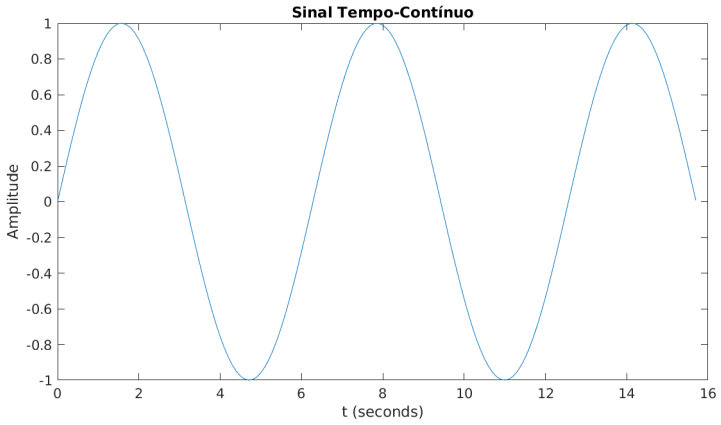


(b)

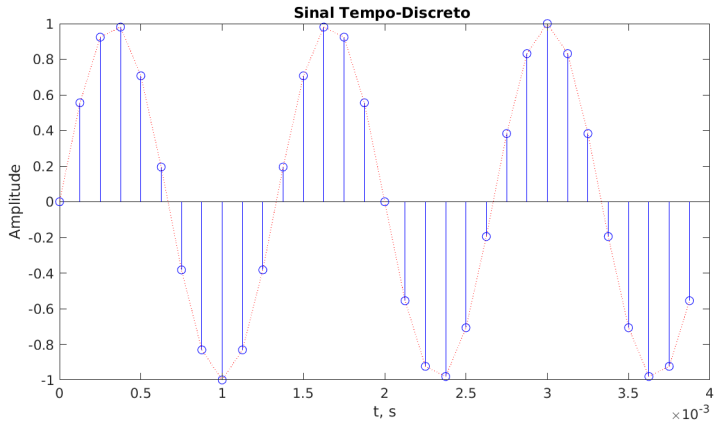
Figura 2 – Fonte: Elaborado pelo autor, 2022.

b) **Sinais Analógicos e Digitais**

O sinal analógico é um sinal de tempo contínuo, visto em Figura 3 (a), onde a função $sen(x)$ com $x = 5\pi t$ varia a amplitude de -1 a 1, podendo assumir qualquer valor da variável independente t, esta propriedade faz dela um sinal analógico. O sinal digital é um sinal de tempo discreto que possui somente um conjunto discreto de valores, visto em Figura 3 (b), onde a amplitude A desta função $Asen(2\pi ft)$ pode assumir somente um conjunto de valores amostrados ou um conjunto discreto de valores e, esta característica, faz dela um sinal digital.



(a)



(b)

Figura 3 – Conversão de um sinal em tempo-contínio para um sinal em tempo-discreto.

Fonte: Elaborado pelo autor, 2022.

Todo sinal analógico que precisa ser transformado para um sinal digital, passa por um conversor de sinal contínuo para sinal discreto (C/D). Este processo segue alguns estágios de tratamento feito pelo conversor, um para a geração de cada sinal e o outro para a extração de informações do sinal.

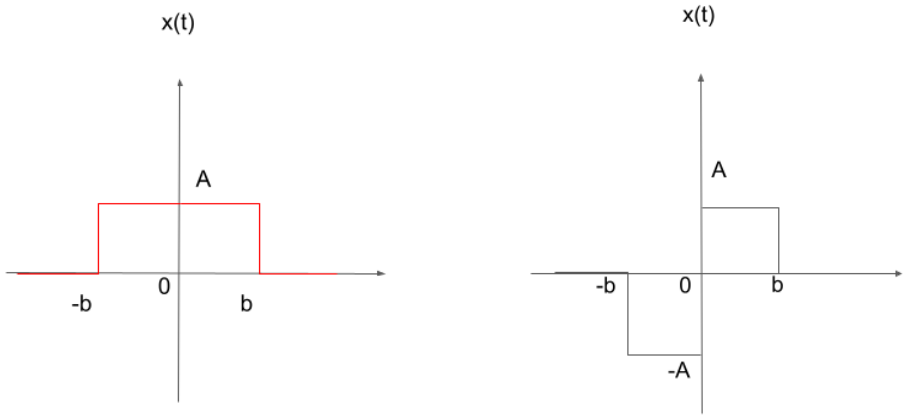
No processamento desses sinais são necessárias três etapas fundamentais: amostragem, quantização e a codificação. A amostragem atribui um valor numérico para cada valor da amplitude do sinal, a quantização dessa amostragem representa cada um desses valores num nıvel de amplitude, e a etapa de codificação faz o sequenciamento dos valores quantizados em um código binário (HAYKIN; VEEN, 2002).

Os sinais analógicos são representados por parâmetros de frequência e tempo, mas dependendo do sinal existe a dificuldade no tratamento desta informação, por exemplo, um sinal randômico. Para resolver este tipo de problema, a

Transformada de Fourier surge como uma útil ferramenta matemática para a análise desse tipo de sinal, porém para isto o sinal é convertido do domínio do tempo t para o domínio da frequência ω.

c) **Sinais Pares e Ímpares**

O sinal de tempo contínuo é definido como par quando $x(t) = x(t)$ é ímpar quando $x(-t) = -x(t)$, esses tipos de sinais estão ilustrados na Figura 4. Essas duas funções são importantes para concluir que todo sinal contínuo é composto pela soma do sinal par x_{par} com o sinal ímpar x_{imp}.



(a) Sinal Par (b) Sinal Ímpar

Figura 4 – As propriedades de multiplicação entre elas gera propriedades importantes para a interação entre esses tipos de sinais para análise.

Fonte: Elaborado pelo autor, 2022.

Qualquer sinal pode ser expresso como a soma de ambos como descrita na Equação 2 e isto gera três importantes propriedades (RAO, 2018) que seguem descritas abaixo:

A parte par e a parte ímpar do sinal são:

$$x_{par}(t) = \frac{x(t) + x(-t)}{2} \tag{2}$$

$$x_{impar}(t) = \frac{x(t) - x(-t)}{2} \tag{3}$$

Da Equação 2 e Equação 3 temos as seguintes propriedades:

i. Multiplicação entre um sinal par e um sinal ímpar produzirá um sinal ímpar. Assumindo que

$$y(t) = \left(\frac{x(t) + x(-t)}{2}\right)\left(\frac{x(t) - x(-t)}{2}\right) \tag{4}$$

Temos

$$y(-t) = -y(t) \tag{5}$$

ii. Multiplicação entre um sinal par e um sinal par produzirá um sinal par. Assumindo que

$$y(t) = \left(\frac{x(t) + x(-t)}{2}\right)\left(\frac{x(t) + x(-t)}{2}\right) \tag{6}$$

$$y(t) = y(t) \tag{7}$$

iii. Multiplicação entre um sinal ímpar e um sinal ímpar produzirá um sinal par. Assumindo que

$$y(t) = \left(\frac{x(t) - x(-t)}{2}\right)\left(\frac{x(t) - x(-t)}{2}\right) \tag{8}$$

$$y(-t) = y(t) \tag{9}$$

d) **Sistema Linearmente Invariante no Tempo (LIT)**

Um sistema invariante no tempo pode ser definido se deslocamento no tempo do sinal de entrada resulta em um deslocamento igual no sinal de saída (OPPENHEIM; WILLSKY; HAMID, 2010).

No tempo contínuo, se $y(t)$ é o sinal de saída de um sistema invariante no tempo e $x(t)$ sendo o sinal de entrada, então este sistema terá $y(t - t_0)$ como a saída quando $x(t - t_0)$ como entrada.

No tempo discreto, se $y[n]$ é o sinal de saída de um sistema invariante no tempo e $x[n]$ sendo o sinal de entrada, então $y[n - n_0]$ é a saída quando $x[n - n_0$ for aplicado na entrada.

e) **Representação do Sinal Contínuo em Termos de Impulsos**

A aproximação de um sinal contínuo $x(t)$ pode ser expresso como a soma de todos os sinais de pulso (RAO, 2018) que será chamado de $\hat{x}(t)$ e está ilustrado

na Figura 5. A definição é dada por

$$\delta_\Delta(t) = \begin{cases} \frac{1}{\Delta}, & 0 < t < \Delta \\ \\ 0, & otherwise \end{cases}$$

Desde que $\delta_\Delta = 1$, $\hat{x}(t)$ pode ser expresso como

$$\hat{x}(t) = \sum_{k=-\infty}^{+\infty} x(k\Delta)\delta(t - k\Delta)\Delta \tag{10}$$

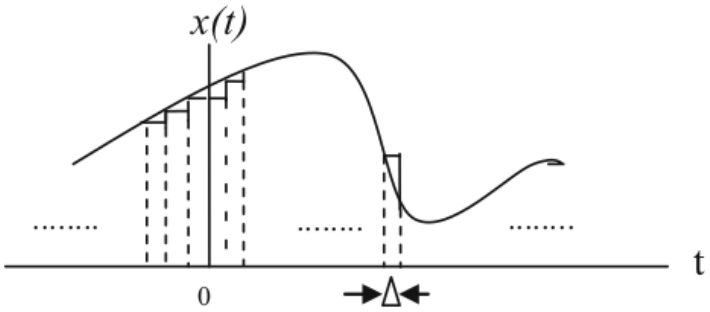


Figura 5 – Este sinal é representado em termos de impulsos.

Fonte: Adaptado (RAO, 2018).

Conforme $\Delta \to 0$, a aproximação $\hat{x}(t)$ pode ser escrita como

$$\hat{x}(t) = lim_{\Delta \to 0} \sum_{k=-\infty}^{+\infty} x(k\Delta)\delta(t - k\Delta)\Delta \tag{11}$$

Como $\Delta \to 0$, a somatória se aproxima de uma integral, e o pulso se aproxima de uma impulso unitário. Com isto, a Equação 11 pode ser reescrita como

$$\hat{x}(t) = \int_{-\infty}^{\infty} x(\tau)\delta(t - \tau)d\tau \tag{12}$$

A interpretação para a Equação 12 é de que o sinal contínuo pode ser representado como superposição ponderada de impulsos deslocados. Deste modo, $x(\tau)d\tau$ é o peso no impulso $\delta(t - \tau)$ que é determinado do valor do sinal de entrada $x(t)$ no tempo da ocorrência de cada impulso.

f) **Representação do Sinal Discreto em Termos de Impulsos**

Um sinal de tempo discreto é composto por uma sequência de impulsos individuais. A Figura 6 mostra uma sequência de impulsos e a soma destes sinais é descrita de forma mais compacta como

$$x[n] = \sum_{k=-\infty}^{+\infty} x[k]\delta[n-k] \tag{13}$$

A Equação 13 é chamada de *propriedade seletiva* que representa uma sequência arbitrária como combinação linear dos impulsos unitários deslocados em $\delta[n-k]$, onde os pesos estão contidos em $x[k]$ (OPPENHEIM; WILLSKY; HAMID, 2010).

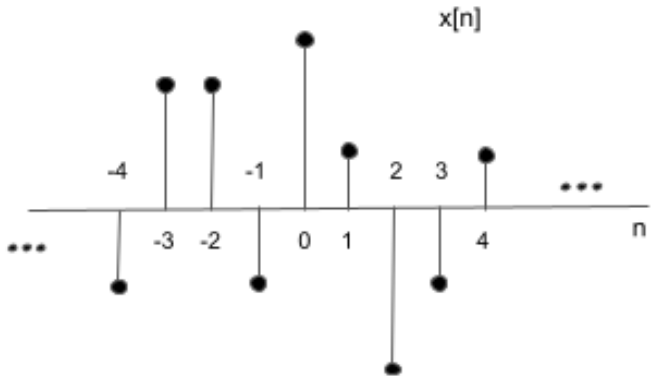


Figura 6 – Este sinal é representado em termos de impulsos discretos que pode ser descrito como:
$x[n] = ... + x[-3]\delta[n+3] + x[-2]\delta[n+2] + x[-1]\delta[n+1] + x[0]\delta[n] + x[1]\delta[n-1] + x[2]\delta[n-2] + x[3]\delta[n-3] + ...$

Fonte: Adaptado (HAYKIN, 1999).

g) **Sistema LIT de Tempo Contínuo: Integral de Convolução**

Um sistema com uma saída gerada por uma entrada descrita como uma superposição ponderada é dada por

$$y(t) = \Re[\hat{x}(t)] = \Re[\int_{-\infty}^{+\infty} x(\tau)\delta(t-\tau)d\tau \tag{14}$$

Para um sistema LIT a variação de sinal de entrada deve ser a mesma no sinal de saída, então $\Re[\delta(t-\tau)] = h(t-\tau)$ e obtém

$$y(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau \tag{15}$$

Portanto a saída $y(t)$ de uma sistema LIT para uma entrada arbitrária $x(t)$ é obtida em termos da entrada do impulso unitário $\delta(t)$, a Equação 15 é definida como Integral Convolucional, denotada pelo símbolo $\star$ (RAO, 2018).

$$y(t) = x(t) \star h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t-\tau)d\tau \tag{16}$$

h) **Sistema LIT de Tempo Discreto: Soma de Convolução**

A representação para a soma de convolução tanto para sistemas lineares quanto para sistemas invariantes no tempo, pois essa soma é resultado da junção desses dois sistemas. Considerando $h_k[n]$ como a resposta linear para um impulso unitário deslocado no tempo $\delta(t-\tau)$, considerando $y[n]$ como a resposta para a entrada $x[n]$ da Equação 13 e que $h_k[n]$ é uma versão deslocada no tempo, a saída pode ser expressa como

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] \tag{17}$$

O resultado da Equação 17 é conhecida como *soma de convolução* ou *soma de superposição*, representada pelo símbolo $\star$ (OPPENHEIM; WILLSKY; HAMID, 2010).

$$y[n] = x[n] \star h[n] \tag{18}$$

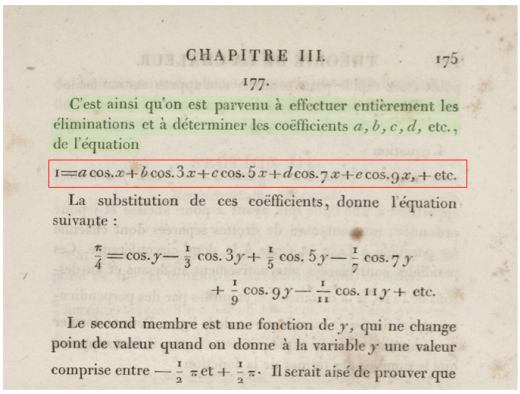
3.2 Transformada de Fourier

A maioria dos sinais são representados no domínio do tempo e a Transformada de Fourier é uma importante ferramenta para este tipo de análise, ou seja, a conversão do sinal do domínio do tempo t para o domínio da frequência ω, facilita a interpretação de sinais complexos, por exemplo, processamento de sinais digitais – revisado na seção 3.1.

O fundamento teórico para a Transformada de Fourier se baseia fundamentalmente nos estudos iniciais em propagação de calor feitos por Jean Baptiste Joseph Fourier[1].

O artigo [2] foi submetido ao *Institut de France* e dentre os revisores estavam Joseph Louis Lagrange (1736-1813) – seu orientador, e Pierre Simon *de* Laplace (1749-1827). Este trabalho sofreu duras críticas de Lagrange e o trabalho de Fourier foi rejeitado. O artigo só foi publicado após a morte de Lagrange (SMITH, 1999).

A afirmação de Fourier foi que a análise da propagação de calor poderia ser simplificada pela soma de funções trigonométricas (Figura 7 e Figura 8), o que foi posteriormente chamada de Série de Fourier (HECK; KAMEN, 2014). A sua representação de uma função periódica $x(t)$ e período T, sem a sua componente complexa, é descrita na Equação 19.



CHAPITRE III. 175

177.

C'est ainsi qu'on est parvenu à effectuer entièrement les éliminations et à déterminer les coëfficients $a, b, c, d,$ etc., de l'équation

$$1 = a\cos.x + b\cos.3x + c\cos.5x + d\cos.7x + e\cos.9x, + \text{etc.}$$

La substitution de ces coëfficients, donne l'équation suivante :

$$\frac{\pi}{4} = \cos.y - \frac{1}{3}\cos.3y + \frac{1}{5}\cos.5y - \frac{1}{7}\cos.7y + \frac{1}{9}\cos.9y - \frac{1}{11}\cos.11y + \text{etc.}$$

Le second membre est une fonction de y, qui ne change point de valeur quand on donne à la variable y une valeur comprise entre $-\frac{1}{2}\pi$ et $+\frac{1}{2}\pi$. Il serait aisé de prouver que

Figura 7 – Acima da equação lê-se: "Assim conseguimos fazer todas as eliminações e determinar os coeficientes a, b, c, d, etc., da equação"
Fonte: Adaptado "The Analytical Theory of Heat".

$$x(t) = \frac{a_0}{2} + a_1 cos(2\pi f_0 t) + a_2 cos(2\pi 2 f_0 t) + ... + b_1 sen(2\pi f_0 t) + b_2 sen(2\pi 2 f_0 t) + ... \quad (19)$$

[1] Obra original de 1878 com o título "The Analytical Theory of Heat"
[2] Mathematics Subject Classification: 33—Special functions em https://zbmath.org/static/msc2020.pdf

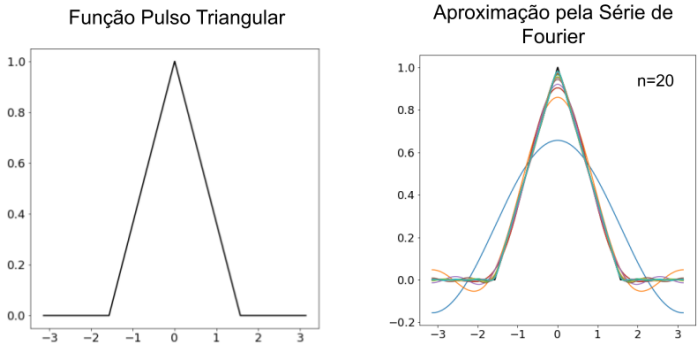


Figura 8 – A Série de Fourier se aproximou da função triangular na vigésima iteração.
Fonte: Elaborado pelo autor, 2022.

Onde, $\frac{a_0}{2}$ = valor médio da função

a_1, b_1 = amplitudes do primeiro harmônico ou a componente fundamental

a_2, b_2 = amplitude do segundo harmônico

a_n, b_n = amplitude do e-nésimo harmônico

$f_0 = \frac{1}{T}$ = frequência fundamental do sinal

A Equação 19 pode ser reescrita do seguinte modo

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}[a_n cos(2\pi n f_0 t) + b_n sen(2\pi n f_0 t)] \tag{20}$$

Além da condição da função ser periódica e caso haja descontinuidade, deve ser finito, finitos números de máximos e mínimos dentro do período e ser integrável em um período, tal que $\int_t^{t+T} |x(t')|dt' < \infty$. Simplificando a série de Fourier, ela pode ser descrita como

$$x(t) = E_0 + \sum_{n=1}^{\infty}[E_n cos(2\pi n f_0 t + \theta_n)] \tag{21}$$

Onde,

$E_0 = \frac{a_0}{2}$

E_n = amplitude do n-ésimo harmônico ; θ_n = fases do n-ésimo harmônico

Utilizando a série de Fourier para transformar o sinal no domínio do tempo para o domínio da frequência é necessário as funções senos e cossenos e a fase da onda informando

o seu deslocamento. Os termos $cos(2\pi n f_0 t)$ e $sen(2\pi n f_0 t)$ formam uma base ortogonal (FIGUEIREDO, 1977), isto é:

$$\int_{-T/2}^{T/2} cos(2\pi n f_0 t) sen(2\pi n f_0 t) dt = 0 \tag{22}$$

Para $n = 0, 1, 2, 3, 4, 5, ...$, os coeficientes a_n da série, são obtidos como segue:

$$a_n = \frac{2}{T} \int_{-T/2}^{T/2} cos(2\pi n f_0 t) dt = 0 \tag{23}$$

Para $n = 0, 1, 2, 3, 4, 5, ..$, os coeficientes b_n da série são obtidos como segue:

$$b_n = \frac{2}{T} \int_{-T/2}^{T/2} sen(2\pi n f_0 t) dt = 0 \tag{24}$$

Os coeficientes E_n e θ_n são obtidos da seguinte forma:

$$E_n = \sqrt{a_n^2 + b_n^2} \tag{25}$$

$$\theta_n = \arctan \frac{b_n}{a_n} \tag{26}$$

A definição do instante inicial, indica a fase do sinal e a consideração do valor da fase, se negativo ou positivo. Quando acontecer o pico positivo do sinal depois do instante zero, a fase será negativa, mas se o pico positivo do sinal estiver mais próximo de aparecer mais próximo de zero, a fase será positiva. A ilustração da Figura 9 exemplifica a transformação do sinal no domínio do tempo para o domínio da frequência com a sua amplitude e fase.

Considerando sinais elétricos, onde o sinal é composto por determinados componentes senoidais com amplitudes e fases, a Transformada de Fourier facilita a interpretação desses sinais. A Figura 10 ilustra o comportamento de uma onda não senoidal no domínio do tempo $eixo - t$ e os valores transformados para o domínio da frequência em função da fase (HAYKIN; VEEN, 2002).

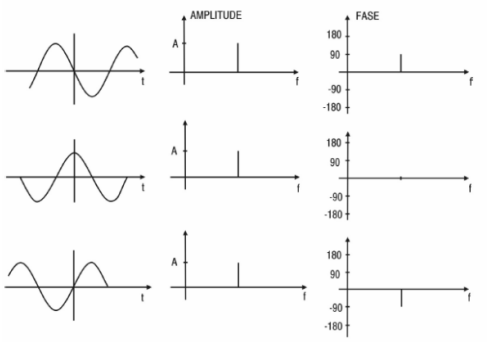


Figura 9 – Exemplo de Amplitude com Fases de sinais cos-senoidal com Fourier (JOAQUIM; SARTORI, 2003).

Fonte: Adaptado (HAYKIN; VEEN, 2002)

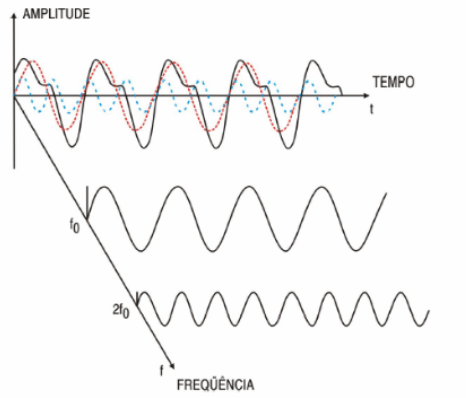


Figura 10 – Sinal transformado do domínio do tempo para o domínio da frequência com o posicionamento das senóides geradas para calcular a fase e a sua amplitude (JOAQUIM; SARTORI, 2003).

Fonte: Adaptado (HAYKIN; VEEN, 2002)

Embora a Transformada de Fourier apresenta uma eficiente forma de tratar a informação, por exemplo, calcular a intensidade de oscilações em frequência [Hz] do sinal, as informações são calculadas pela média de todo o domínio do tempo e com mudanças repentinas no início e no fim do sinal.

Alguns eventos podem não ser detectados pela Transformada de Fourier e isto limita esta transformada em algumas situações, onde os valores médios não são suficientes, conforme ilustrado na Figura 11.

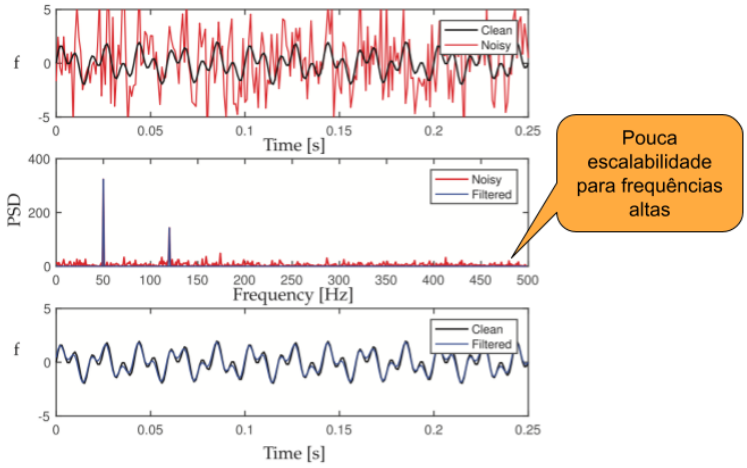


Figura 11 – Pouca informação para altas frequências.

Fonte: Eleborado pelo autor, 2022.

Dennis Gabor, em 1946 fez uma nova abordagem para resolver este problema, ele fez com que a manipulação do sinal fosse feito por funções conhecidas como janelas. Esta técnica ficou conhecida como a Transformada de Fourier de Curta Duração ($STFT$) (MULLER, 2015).

3.3 Transformada de Fourier de Curta Duração (STFT)

Uma introdução prévia ao Produto Interno Hermitiano é necessária antes da introdução na STFT, que é definido como

$$< f(x), g(x) >= \int_a^b f(x)\bar{g}(x)dx = 0 \tag{27}$$

$\bar{g}(x)$ denota o conjugado complexo de g(x).

A relevância da consideração desse produto interno para discretizar as funções $f(x)$ e $g(x)$, destina-se para a criação de vetores de dados sob o ponto de vista de aplicabilidade. (BRUNTON; KUTZ, 2019). O produto interno dos vetores $\mathbf{f} = [f_1 f_2 ... f_n]^T$ e $\mathbf{g} = [g_1 g_2 ... g_n]^T$ é definido por

$$
\begin{aligned}
< \mathbf{f}, \mathbf{g} > &= \mathbf{g}^{*}\mathbf{f} && (28)\\
&= \sum_{k=1}^{n} f_k \bar{g}_k && (29)\\
&= \sum_{k=1}^{n} f(x_k)\bar{g}(x_k) && (30)
\end{aligned}
$$

Onde, $\mathbf{g}^{*}$ é a transposta do conjugado complexo de $\mathbf{g}$.

A magnitude desse produto interno crescerá quanto mais pontos de dados forem adicionados, ou seja, aumentando "n".

Se as funções $f(t)$ e $g(t)$ estão no intervalo $[a, b]$, a normalização de $\Delta x = \frac{(b-a)}{(n-1)}$ é dada por

$$
\frac{(b-a)}{(n-1)} < \mathbf{f}, \mathbf{g} >= \sum_{k=1}^{n} f(x_k)\bar{g}(x_k)\Delta x \tag{31}
$$

A Equação 31 é uma aproximação de Riemann para o produto interno de funções contínuas, isto por causa do limite de $n \to \infty$. O produto interno do vetor converge para o produto interno das funções da Equação 27.

Este produto interno induz uma norma nas funções dada por

$$
||f||_2 = (< f, f >)^{\frac{1}{2}} = \sqrt{< f, f >} = \left(\int_a^b f(x)\bar{f}(x)dx \right)^{\frac{1}{2}} \tag{32}
$$

O conjunto de todas as funções com norma limitada define o conjunto de funções quadradas integráveis, denotadas por $L^2([a, b])$. Isto é conhecido como o conjunto de funções integráveis de Lebesgue, com intervalos $(-\infty, +\infty), [a, +\infty)$ ou periódico $[-\pi, \pi)$ (BRUNTON; KUTZ, 2019).

A STFT tem por objetivo encontrar uma pequena seção do sinal contínuo no tempo, ao redor de um ponto t através de uma única função de janelamento em torno do ponto $t = 0$. Esta função é deslocada através do tempo e se $f \in L^2(\Re)$ é um sinal e $g : \Re \to \Re$ seja uma função de janelamento (MULLER, 2015), então a função $f_{g,t}$ localizada no ponto t é definida como

$$
f_{g,t}(u) := f(u)g(u - t) \tag{33}
$$

Desse modo, dado um sinal $f \in L^2(\Re)$ bem como uma função de janelamento $g \in L^2(\Re)$ (BRUNTON; KUTZ, 2019), a STFT (tempo contínuo) é definida como

$$f[stft]_g(t,\omega) := f[fourier]_{g,s}(\omega) = \int_{u\in\Re} f(u)\overline{g}(u-t)exp(-2\pi i\omega)du \tag{34}$$

O espaço $L^2(\Re)$ é definido como o conjunto de todos os sinais de energia finita contemplando sinais contínuos ou discretos no tempo para a aplicação da transformada de Fourier e STFT.

Os tipos de janelamento mais utilizados são o retangular, triangular e o de Hann (Hanning), em especial este último que é mais utilizado em processamento de sinais de aúdio por suavizar os limites para zero (MULLER, 2015). A Figura 12 ilustra a aplicação da STFT em áudio gerado por um inseto com alguns tipos de janelamentos aplicados. A frequência deste sinal no tempo t é $\omega = 20Hz$ e a STFT localizou frequências próximo de $\omega = 20Hz$.

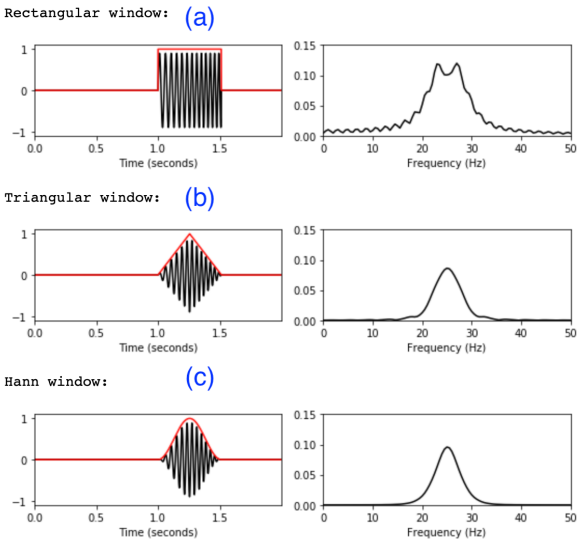


Figura 12 – Sinal de de som de inseto janelado e suas tranformados de Fourier usando diferentes funções de janelamento. (a) Janela retangular. (b) Janela triangular. (c) Janela Hanning.

Fonte: Adaptado (MULLER, 2015)

A STFT de um sinal consegue dispor as informações de cada ponto no tempo t na frequência ω e estas informações são visualizadas pela média de um espectrograma em duas dimensões, sendo o eixo horizontal para o tempo e o eixo vertical para a frequência. A ilustração da Figura 13 representa um espectrograma do som gerado por um inseto, onde $f(t) = sin(400\pi t^2)$ para $t = 0$ para $t \in [0, 1]$ e é possível observar que a frequência aumenta linearmente de $\omega = 0$ em $t = 0$ até $\omega = 400Hz$ em $t = 1$ e as barras à direita representam

os coeficientes de magnitude d_ω que expressam a intensidade em uma específica frequência ω num específico tempo t, segundo Muller (2015).

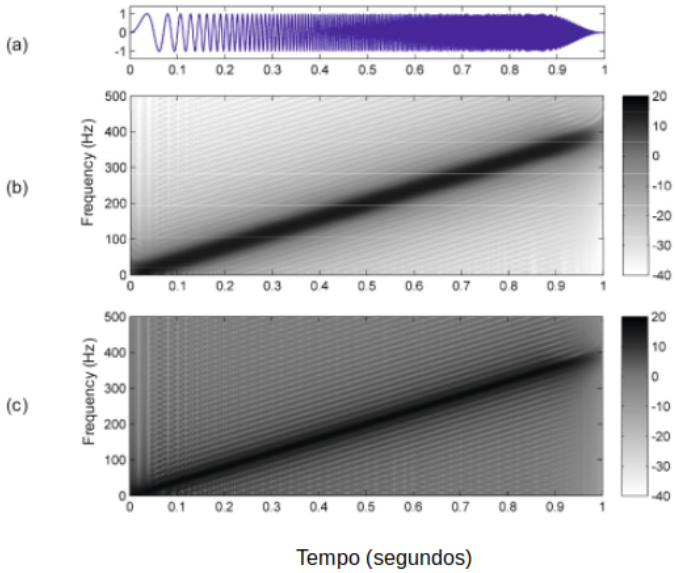


Figura 13 – Espectrograma de som de inseto com dois diferentes funções de janelamento. (a) Sinal. (b) Janela Hanning com tamanho de 62.5ms. (c) Janela retangular com tamanho de 62.5ms.

Fonte: Adaptado (MULLER, 2015).

Nesta seção foram apresentadas a Série de Fourier, Transformada de Fourier(Figura 14)que muda o domínio de um sinal do tempo para domínio da frequência, o conjunto integrável de Lebesgue, as funções de janelamento através da STFT que, em instantes definidos no domínio do tempo, possuem a capacidade de criar espectrogramas com aplicações na pesquisa em tratamento de áudio.

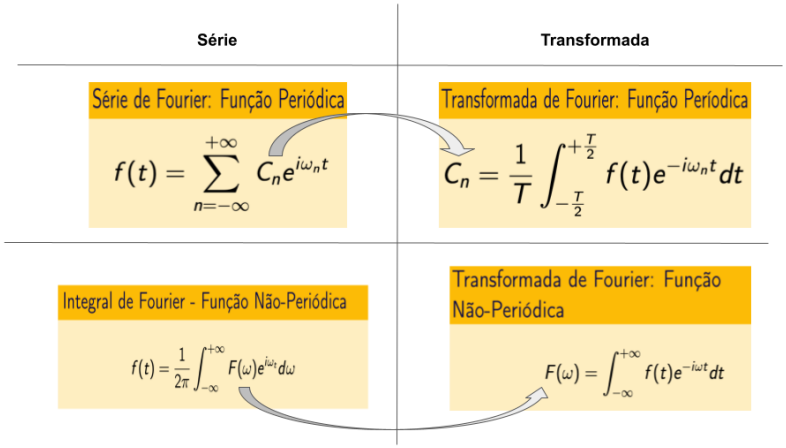


Figura 14 – Os coeficientes do lado das séries tornam-se as tranformadas.

Fonte: Elaborado pelo autor, 2022.

3.4 Transformada Wavelet

Embora a transformada de Fourier permite analisar todo o sinal no domínio da frequência, os eventos que ocorrem em intervalos t distintos afetam globalmente a transformada. Por outro lado a STFT permite uma análise do sinal localmente no tempo, fazendo com que uma função de janelamento desloca-se pelo domínio do tempo, mas a sua função de janelamento permanece constante para todas as frequências (HAYKIN; VEEN, 2002).

O princípio detrás das transformadas de Wavelet é muito similar ao das Transformadas de Fourier, ou seja, aaquele de fazer-se uma projeção linear do sinal numa função base para assim obter informação relevante (GEARMIN, 2009). Aqui começam as diferenças, pois enquanto em Fourier utiliza-se funções cossenoidais, que têm domínio infinito, as Wavelets têm domínio compacto: isso permite localizar e estudar características do sinal que sejam localizados no tempo com escalabilidade em ambos os domínios: tempo e frequência, como ilustra a Figura 15. O espectrograma, isto é, os coeficientes das expansões do sinal nas funções da base, não são localizados na transformada de Fourier (esquerda), são localizados na transformada de Fourier janelada (centro) e são localizados e em multiescala nas transformadas de wavelets (direita)

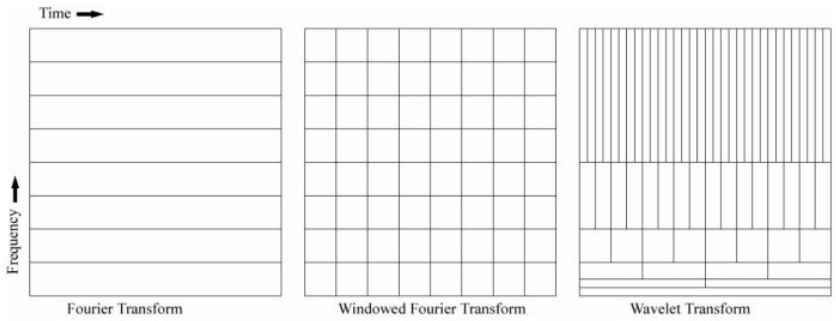


Figura 15 – Multiresolução de escalas através da Transformada Wavelet.
Fonte: Adaptado (GEARMIN, 2009).

O termo *wavelet* foi introduzido originalmente por Jean Morlet e sua base matemática foi desenvolvida pelo físico teórico Alex Grossmann. A sismologia foi a área na qual a transformada de Wavelet surgiu e através de dados sísmicos, Morlet percebeu que a transformada de Fourier não era tecnicamente adequada para este tipo de análise (GROSSMAN; MORLET, 1984). Para a investigação de sinais, as funções Wavelets foram

catalogadas dentro de dicionário de Wavelet, que foi construído a partir da Wavelet Mãe ψ de média zero Equação 35 (MALLAT, 2008):

$$\int_{-\infty}^{+\infty} \psi(t)dt = 0, \tag{35}$$

Esta mesma Wavelet Mãe é dilatada pelos parâmetros de escala **s** e o de posição **u**.

$$\Psi_{s,u}(t) = \frac{1}{\sqrt{s}}\Psi(\frac{t-u}{s}), u \in \Re, s > 0 \tag{36}$$

A função de dilatação da Wavelet Mãe deve satisfazer a Condição de Admissibilidade, que preserva a característica ondulatória da função Wavelet de forma prática (DOMINGUES *et al.*, 2016).

$$C_\psi = 2\pi \int_{-\infty}^{+\infty} \frac{|\hat{\psi(u)}|^2}{|u|} du < \infty, \tag{37}$$

onde, $\hat{\psi(u)}$ é a transformada de Fourier de $\psi(t)$ e com isto se $\hat{\psi}$ é uma função contínua, então C_ψ pode ser finito somente se $\hat{\psi(0)} = 0$, ou seja, $\int_{-\infty}^{+\infty} \psi(t)dt = 0$ para preservar a característica ondulatória da função Wavelet.

$$\mathcal{W}_(s,u) = \int_{-\infty}^{\infty} f(t)\psi_{s,u}(t)dt \tag{38}$$

A Equação 38 depende de dois parâmetros, s e u, que correspondem respectivamente às informações de escala e de tempo ou translação e esta equação pode ser também representada como o produto interno de f com uma função Wavelet $\psi_{s,u}$: $F(u,s) =< f, \psi_{s,u} >$. A generalização de um sinal dado $f(t)$ em relação a uma Wavelet Mãe da Equação 39, também pode considerar que uma Wavelet é a dilatação de s com a translação em u de uma Wavelet Mãe ψ:

$$\mathcal{W}_{\psi(f(t))(s,u)} = \langle f(t)\ ,\ \psi_(s,u) \rangle \tag{39}$$

A Transformada Wavelet projeta um sinal $x(t)$ dentro de uma janela de tempo chamada "Wavelet", ela é transformada para permitir um redimensionamento da escala s onde a mudança da localização da frequência acontece e a Wavelet pode ser deslocada no tempo para qualquer localização da translação u e o termo da direita Ψ é o tipo de Wavelet escolhida fazendo u variar no tempo t e a escala s variar na frequência ω e podemos

visualizar a Wavelet atuando em função do tempo e da frequência conforme a ilustração da Figura 16 que representa a variação da escala "s" e da translação "u" (WEIHS *et al.*, 2019).

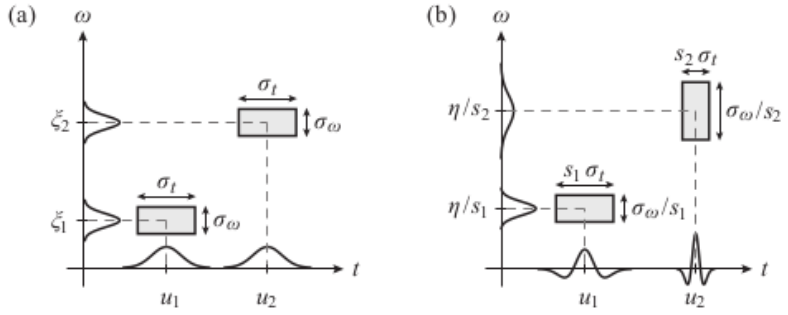


Figura 16 – (a) STFT em instantes u_1 e u_2 e suas frequências ξ_1 e ξ_2. (b) transformada Wavelet variando ambos os domínios em janelas diferentes.

Fonte: Adaptado (WEIHS *et al.*, 2019).

A característica fundamental da Transformada Wavelet é a manipulação da escala da frequência com a translação no tempo com a vantagem do tamanho de janela não ser constante, isto faz com que a manipulação dos parâmetros de escala "s" e o de translação "u" exerçam um papel importante no emprego dessa técnica. A **??** ilustra como esses parâmetros se deslocam de forma mais geral.

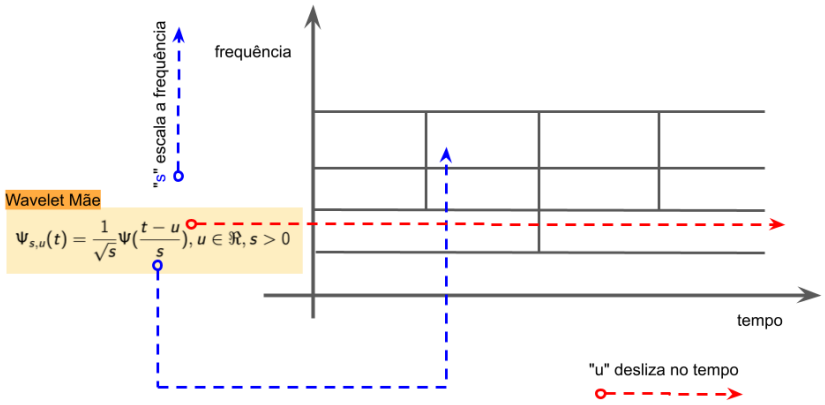


Figura 17 – Parâmetro **s** se desloca pela dimensão da frequência, assim como o parâmetro **u** se desloca na dimensão do tempo.

Fonte: Eleborado pelo autor, 2022.

O parâmetro de escala "s" de uma Wavelet tem o significado de poder dilatar ou contrair a Wavelet Mãe, de tal modo que, quanto menor o valor de escala 's", mais contraída será a Wavelet e considerando esta característica, a escala está relacionada com

a frequência do sinal que baixa escala "s" faz com que a Wavelet seja comprimida e os detalhes da transformada mudem rapidamente em uma alta frequência ω.

Por outro lado, uma alta escala "s" implica em Wavelet dilatada onde os detalhes da transformada mudam lentamente em uma baixa frequência ω (MALLAT, 2008) e a Figura 18 ilustra a forma como os domínios variam de formas diferentes.

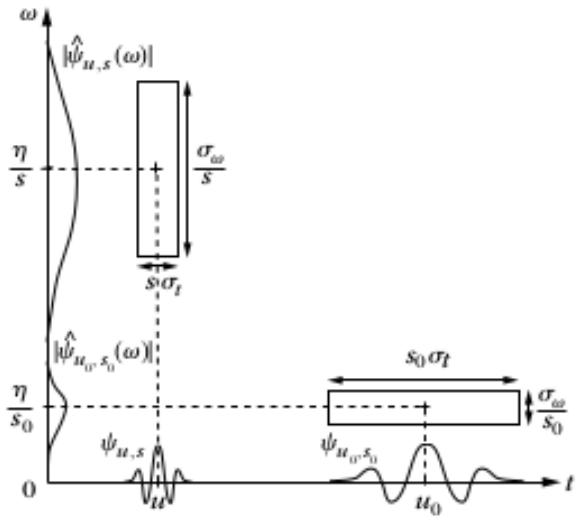


Figura 18 – A baixa escala de "s" deixa a Wavelet comprimida e com alta frequência ω (janela esquerda). A alta escala "s" faz com que a Wavelet fique dilatada e com baixa frequência ω (janela direita).

Fonte: Adaptado (MALLAT, 2008).

3.5 Tipos de Wavelet

1. **Wavelet de Haar**

   O exemplo de Wavelet mais simples é a Wavelet *Haar*, desenvolvida em 1910 (HAAR, 1910) como

$$\Psi(t)_{Haar} = \begin{cases} 1, & 0 \leq t < \frac{1}{2} \\ -1, & \frac{1}{2} \leq t < 1 \\ 0, & otherwise \end{cases}$$

   As três Wavelets de Haar, $\Psi_{1,0}$, $\Psi_{1/2,0}$ e $\Psi_{1/2,1/2}$ estão ilustrados na Figura 19, na Figura 20, na Figura 21 e sintetizada na Figura 22.

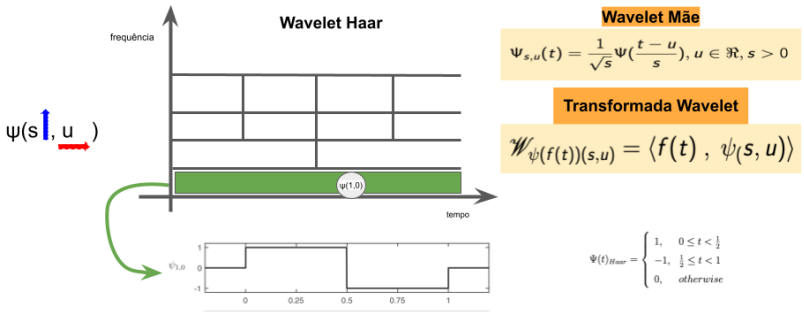


Figura 19 – Primeira camada de de multiresolução ilustrada na Figura 15.

Fonte: Adaptado (BRUNTON; KUTZ, 2019).

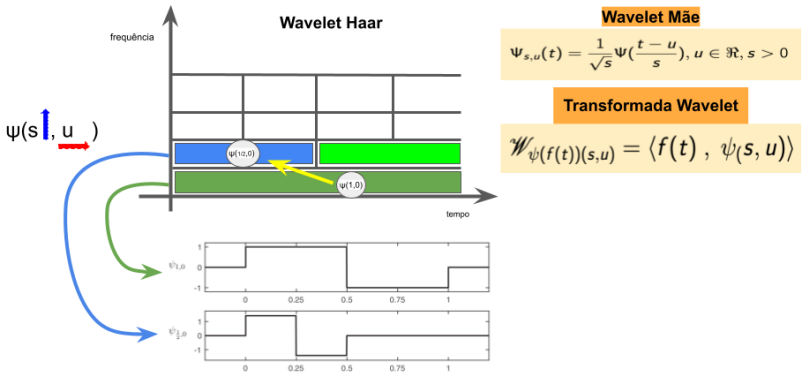


Figura 20 – Segunda camada de de multiresolução ilustrada na Figura 15.

Fonte: Adaptado (BRUNTON; KUTZ, 2019).

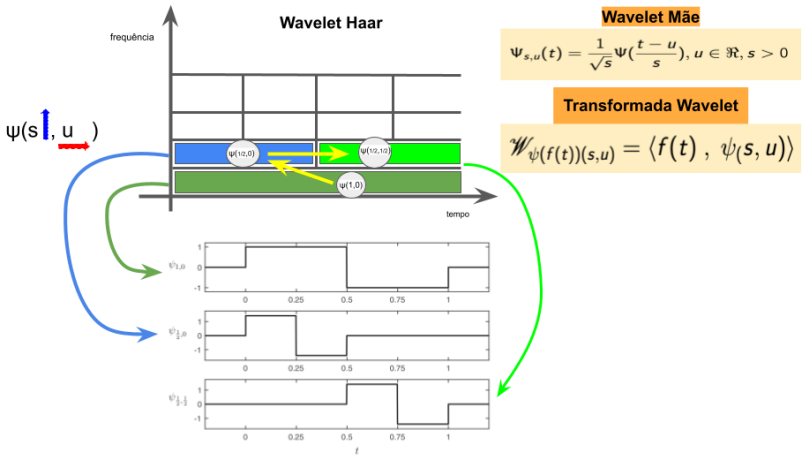


Figura 21 – Terceira camada de multiresolução ilustrada na Figura 15.

Fonte: Adaptado (BRUNTON; KUTZ, 2019).

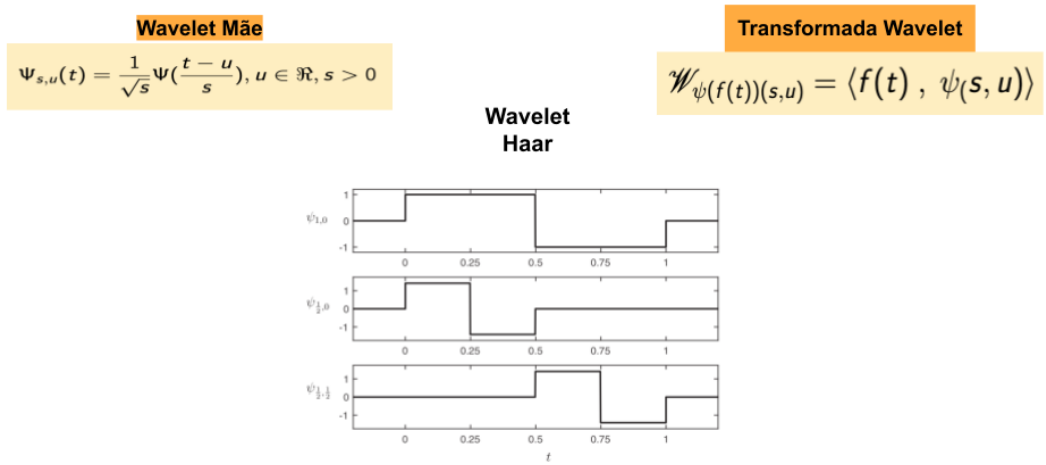


Figura 22 – Terceira camadaa de multiresolução ilustrada na .

Fonte: Adaptado (BRUNTON; KUTZ, 2019).

Dada uma série $T = t_1, ..., t_n$, a Wavelet Haar Discreta transforma a saída em duas séries: as aproximações A_i e os detalhes D_i, onde $1 \leq i \leq \frac{n}{2}$ (LI *et al.*, 2016) descritos como

$$A_i = \frac{t_{2i-1} + t_{2i}}{\sqrt{2}} \tag{40}$$

$$D_i = \frac{t_{2i-1} - t_{2i}}{\sqrt{2}} \tag{41}$$

2. **Wavelet Mexican Hat**

A Wavelet Chapéu Mexicano (*Mexican Hat*, este nome deve-se a curva parecer um chapéu mexicano) tem sido indicada para o tratamento de áudios musicais e ela é a segunda derivada da Gaussiana e normalizada para $[0, 1]$, definida pela Equação 42 e a Figura 23 ilustra esse tipo de Wavelet com σ=1 (MALLAT, 2008).

$$\psi(t) = \frac{2}{\pi^{1/4}\sqrt{3\sigma}} \left(\frac{t^2}{\sigma^2} - 1\right) \exp\left(\frac{-t^2}{2\sigma^2}\right) \tag{42}$$

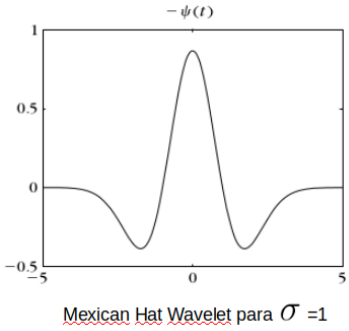


Figura 23 – Wavelet com σ=1.

Fonte: Adaptado (MALLAT, 2008).

A transformada de Wavelet apresenta um espectrograma diferente da STFT quando aplicado em um sinal, a ilustração da Figura 24 apresenta uma aplicação da Wavelet Mexican Hat em uma dada $f(t)$. As cores representam as variações dos coeficientes da Wavelet, neste caso as cores preta, cinza e branca correspondem, respectivamente, aos coeficientes positivo, zero, e negativo desta Wavelet. (MALLAT, 2008).

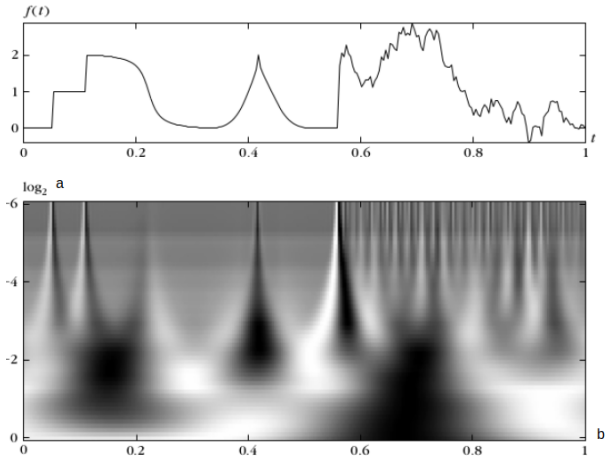


Figura 24 – Transformada Wavelet aplicada em um sinal

Fonte: Elaborado pelo autor, 2020.

3. **Wavelet Morse Generalizada**

A Wavelet conhecida como Morse Generalizada é um tipo de Wavelet utilizada em sinais modulados, definida no domínio da frequência pela Equação 43. A frequência de pico é $\left(\frac{P}{\gamma^2}\right)^{\frac{1}{\gamma}}$.

$$\psi_{P,\gamma}(\omega) = U(\omega)\alpha_{P,\gamma}\omega^{\frac{P^2}{\gamma}}e^{-\omega^{\gamma}} \tag{43}$$

Onde,

Uω é a função de etapa de Heaviside. O termo $\alpha_{\beta,\gamma}$ é a constante de normalização da Wavelet. β e γ são os parâmetros respectivamente reponsáveis pelo decaimento e simetria da função Wavelet, mas estão representadas pela variável P^2 que é o produto $\beta\gamma$, sendo a largura de banda pelo tempo (LILLY; OLHEDE, 2012). A Figura 25 ilustra as variações dos parâmetros P e γ.

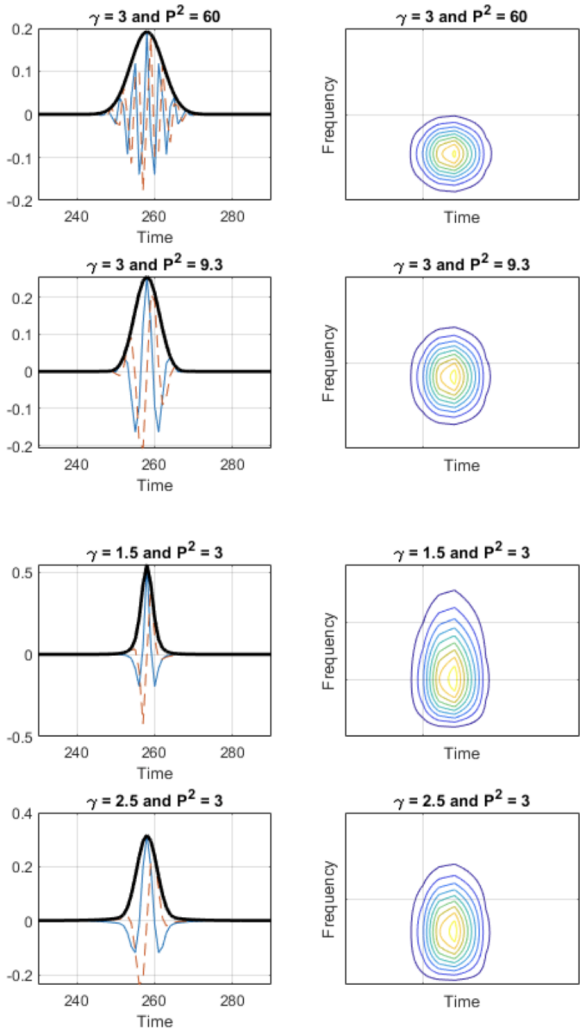


Figura 25 – Variando os parâmetros P e γ

Fonte: Adaptado (LILLY; OLHEDE, 2012).

3.6 Conceitos de Redes Neurais

Nesta seção foi apresentado um tipo de aprendizagem baseada em um modelo abstrato do funcionamento dos neurônios do ser humano. A rede neural computacional foi desenvolvida por McCulloch e Pitts (1943) e a inspiração biológica deste modelo consiste no processamento da informação através dos neurônios, onde cada um recebe sinais de diversos outros neurônios através de conexões sinápticas. As unidades de processamento (neurônios) executam determinadas funções matemáticas e são dispostas em uma ou mais camadas que se interligam por conexões, a informação recebida do processo de entrada é associado a pesos que ponderam a informação para o neurônio de saída (RUSSELL, 2015).

A proposta do modelo baseado no comportamento do neurônio é uma simplificação, pois há diversos outros tipos de neurônios[3], os dendritos são representados como n terminais de entrada $\{x_1, x_2, ..., x_n\}$ e o axônio é representado pela saída y, a dinâmica da sinapse é feito pelos pesos $\{w_1, w_2, ..., w_n\}$ associado a cada terminal de entrada, implicando que a sinapse particular é dada por $w_i x_i$ (MCCULLOCH; PITTS, 1943). As ilustrações da Figura 26 apresenta um tipo de neurônio biológico e um modelo matemático para fazer com que a condição de ativação seja dada pela seguinte função linear (HAYKIN, 1999):

$$\sum_{i=1}^{n} w_i x_i \geq \theta \tag{44}$$

onde, n é o número de entrada do neurônio, w_i é o peso associado à entrada x_i e θ é o limiar do neurônio.

[3] https://sites.icmc.usp.br/andre/research/neural/image/neuro5.jpg, acessado em 3/1/2020

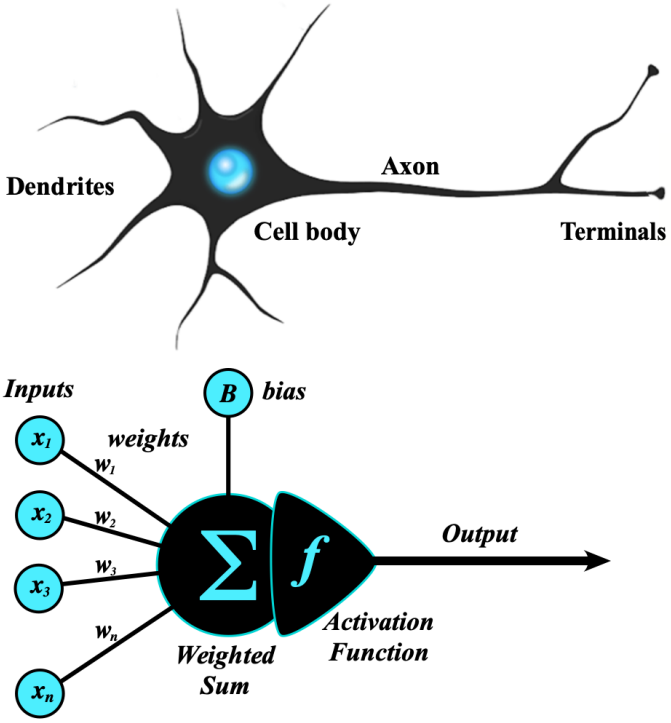


Figura 26 – (a) neurônio biológico (b) modelo criado por McCulloch e Pitt (RUSSELL, 2015).

Fonte: Elaborado por (BRUNEL; HAKIM; RICHARDSON, 2014)

Para a ativação do neurônio da Figura 26 há algumas funções de ativação, assim chamadas, para serem utilizadas nas redes neurais.

- **Função Sigmoide (Logística)**

Esta função compreende os valores resultantes do conjunto dos $\Re$ e é definida por

$$y(x) = \frac{1}{1 + exp(-x)} \tag{45}$$

O parâmetro a refere-se a inclinação da curva, a ativação para gerar o valor de saída é efetuada com a derivada da Equação 45 que resulta em

$$y'(x) = ay(1 - y) \tag{46}$$

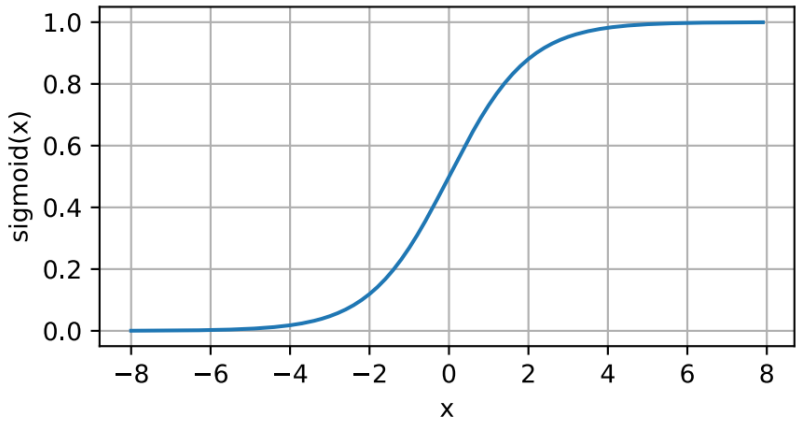


Figura 27 – Esta função compreende todos os valores dos $\Re$ de $-\infty, +\infty$.

Fonte: Adaptado (ZHANG *et al.*, 2021).

- **Função Hiperbólica**

Outra função utilizada é a *tanh* definida por

$$y(x) = \frac{1 - exp(-2x)}{1 + exp(-2x)} \tag{47}$$

Sendo a função de ativação a sua derivada como descrita abaixo

$$y'(x) = (1 + y)(1 - y) \qquad -1 \leq y \leq 1 \tag{48}$$

A função linear também é utilizada para ter a sua derivada como função de ativação:

$$y(x) = x, \qquad com \quad a \quad derivada \quad y' = 1 \tag{49}$$

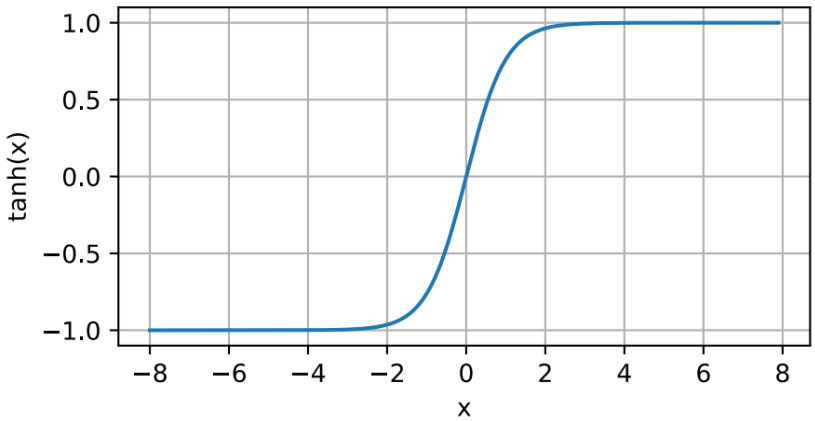


Figura 28 – Esta função tem quase o mesmo comportamento da Sigmoide.

Fonte: Adaptado (ZHANG *et al.*, 2021).

- **Função ReLu**

A função ReLu (*Rectified Linear Unit*) considera somente os valores positivos da saída de cada neurônio. Dado um elemento x, a função de ativação é definida com o máximo daquele valor e 0.

$$ReLu(x) = max(x, 0) \tag{50}$$

Um exemplo de aplicação desta função é utilizá-la em cada saída de cada camada de convolução de uma Rede Neural Convolucional. Isto será visto na seção 3.8.

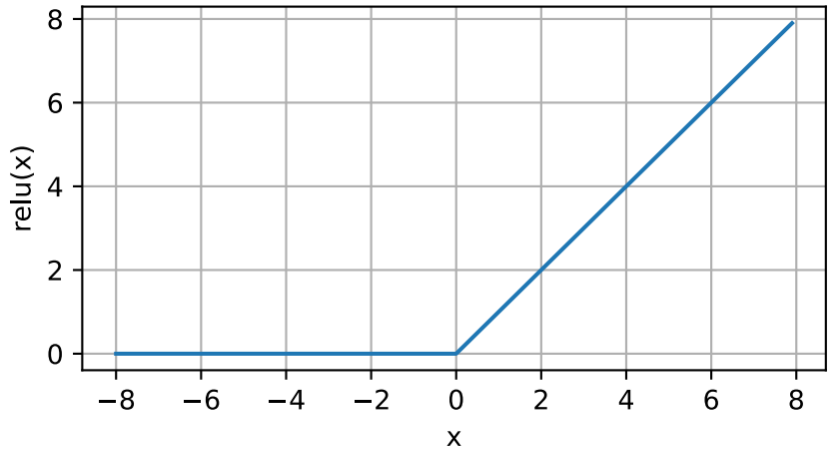


Figura 29 – Preserva os valores positivos.

Fonte: Adaptado (ZHANG *et al.*, 2021).

A arquitetura de uma rede neural é importante para saber qual o tipo de problema poderá ser abordado por ela. As redes neurais com uma camada conseguem resolver apenas problemas linearmente separáveis, por exemplo, classificar os bits de uma porta lógica OR e AND, mas não conseguem o mesmo feito em uma porta lógica XOR. Com o número de camadas, tipo de camadas e o tipo de conexão entre os neurônios a arquitetura da rede define o algoritmo mais adequado para ser usado (HAYKIN, 1999).

A arquitetura generalizada de uma rede neural com múltiplas camadas é conhecida como ***Feedforward***, onde a informação é tratada a partir da camada de entrada até a camada de saída sem retroalimentação (GOODFELLOW; BENGIO; COURVILLE, 2016).

A Figura 30 ilustra a arquitetura de uma Rede Neural Feedforward. Cada valor de entrada das variáveis do sistema, o fluxo é contínuo sem retroalimentação entre as camadas.

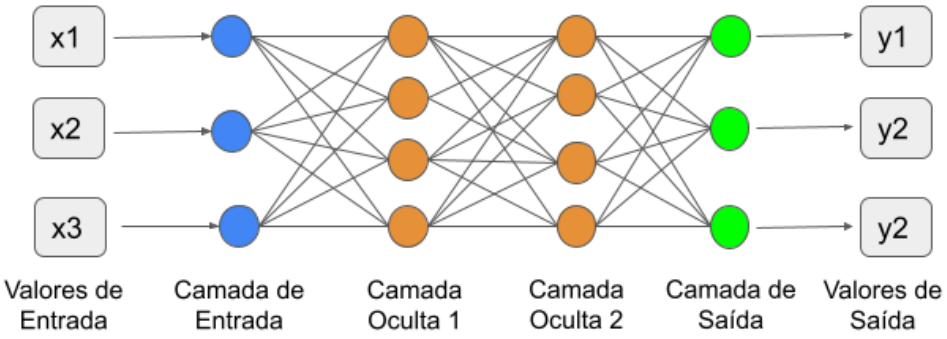


Figura 30 – Exemplo de uma Rede Neural de Aprendizagem Profunda com camadas ocultas

Fonte: Elaborado pelo autor, 2022.

Neste contexto, há redes neurais com camadas ocultas, ou seja, arquiteturas complexas exercendo um papel fundamental para a solução dos mais diversos tipos de problemas que serão tratados mais adiante.

3.7 Redes Neurais Profundas

Para trabalhar com redes de aprendizagem profunda é importante ter os conjuntos de dados separados para treinamento, teste e validação, este último faz o ajuste de hiperparâmetros do modelo e também previne de super-ajuste (*overfitting*) porque faz uma parada precoce (*early stopping*) e tem sido um método padrão de utilização.

Há um risco iminente do aparecimento de *overfitting* e assim como os pesos, os ajustes destes hiper-parâmetros do algoritmo de aprendizagem podem ser feitos durante as tentativas manuais apresentadas mais à frente desta seção e usando os conjuntos de validação como o guia ilustrado na Figura 31 (WITTEN *et al.*, 2016).

Esta monitorização é feita pela medição dos valores da curva que plota a perda média dos conjuntos de treinamento e de validação. A ilustra o ponto no qual a perda média do conjunto de validação começa a se deteriorar (WITTEN *et al.*, 2016).

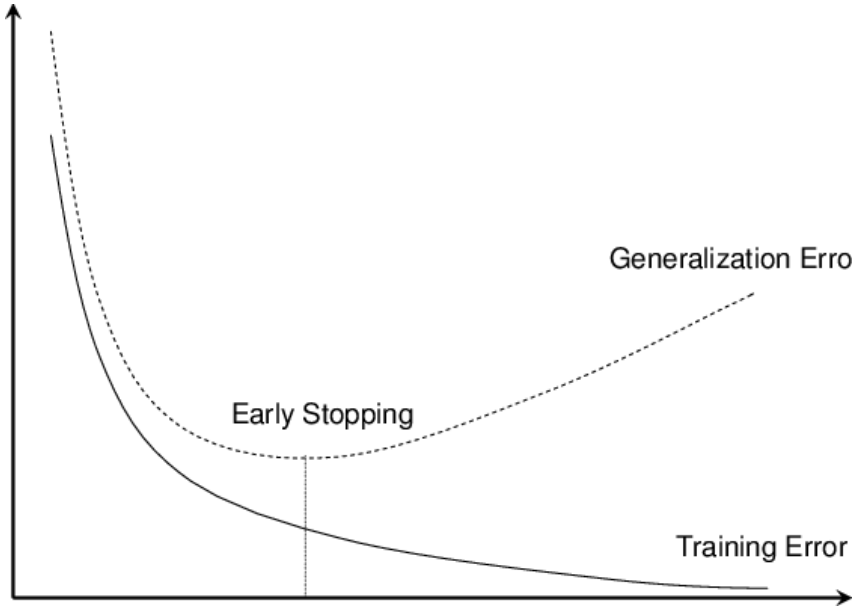


Figura 31 – Curva de aprendizagem para os conjuntos de treinamento e de teste.

Fonte: Adaptado (WITTEN *et al.*, 2016).

- **Backpropagation**

  Este método é referente ao cálculo dos parâmetros do gradiente da Rede Neural, diferentemente do fluxo do *feedforward*, este método atua de forma reversa, ou seja, da camada de saída para a camada de entrada que é definido como

$$\frac{\partial Z}{\partial X} = prod\left(\frac{\partial Z}{\partial y}, \frac{\partial Y}{\partial X}\right) \tag{51}$$

  onde,

  X, Y, Z = tensores

  $prod$ = operador para multiplicar os argumentos depois que as operações originais tenham sido efetuadas, tais como transposição e mudanças de posicionamento de valores da entrada (ZHANG *et al.*, 2021).

- **Gradiente Descendente Estocástico Baseado em Mini-Lote**

  Esta técnica (*Mini-Batch-Based Stochastic Gradient Descent*) apresenta bom resultado para as redes neurais com problema de um mínimo local, naquele caso, para encontrar o melhor peso para a "backpropagation". O Gradiente Estocástico Descendente é utilizado para problemas com mais de um mínimo local, na técnica deste item, ele atualiza os parâmetros do modelo de acordo com o gradiente computado de um exemplo. A variável de mini-lote usa um pequeno subconjunto dos dados e atualiza

os parâmetros na média do gradiente nos exemplos dos lotes. Operacionalmente segue o fluxo abaixo ilustrado na Figura 32.

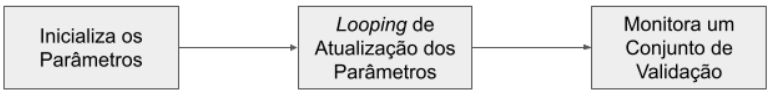


Figura 32 – Fluxo do Gradiente Estocástico Descendente de Mini-Lote
Fonte: Elaborado pelo autor, 2022.

Cada passagem por um conjunto de mini-lotes representa um conjunto de treinamento completo, que é uma *época*. Depois do processamento, um mini-lote é atualizado pelo termo de regularização com a função objetiva dada por

$$\theta^{new} \leftarrow \theta - \eta_t \left[\frac{1}{B_k} \sum_{i \in I} \left[\frac{\partial}{\partial \theta} L(f(x_i; \theta), y_i) \right] + \frac{B_k}{N} \lambda \frac{\partial}{\partial \theta} R(\theta) \right] \tag{52}$$

onde,

θ e $R(\theta)$ = reguladores

η_i = é taxa de aprendizagem (pode depender na época t)

kth = lote

B_k = exemplos representados pelo conjunto $I = I(t, k)$

$I = I(t, k)$ = índices dentro do dado original

N = tamanho do conjunto de treinamento

$L(f(x_i; \theta), y_i)$ = perda por exemplo

x_i= rótulo

y_i = parâmetro

λ = peso

Quando um único mini-lote contém o conjunto inteiro de treinamento, o padrão é atualizado pelo gradiente descendente, enquanto que no outro, com tamanho de lote 1, é o padrão a atualização do gradiente descendente estocástico do exemplo único. Esta técnica de classificação pode ser comprometida em função da capacidade computacional e o tamanho do lote pode influenciar a estabilidade e velocidade da aprendizagem.

- ***Dropout***

  O *Dropout* é uma forma de deletar aleatoriamente unidades e suas conexões durante o treinamento, reduzir o grau de coadaptação dessas unidades e então combater o *overfitting*. Se a unidade é retida com uma probabilidade p durante o treinamento, seus pesos de saída são escalados ou multiplicados por um fator de p na hora do teste (WITTEN *et al.*, 2016). Sugerimos o valor de 0,25 para esta variável.

- **Normalização de Lote (*Batch Normalization*)**

  Batch normalization é uma forma de aceleração do treinamento e tem sido utilizado para problemas de *benchmark* e cada elemento de uma camada na rede neural, é normalizada com média 0 e variância 1 baseada na estatística de cada mini-lote, porém isto pode mudar o poder representacional da rede neural, então para cada ativação é dada uma escala de aprendizagem e parâmetro de mudança. O gradiente estocástico descendente de mini-lote é modificado pelo cálculo da média μ_j e variância $\sigma^2{}_j$ sobre o lote de cada unidade h_j em cada camada oculta e assim normalizando as unidades, dimensionando-as usando o parâmetro de escala aprendido γ_j e deslocando-os pelo parâmetro de deslocamento aprendido β_j (WITTEN *et al.*, 2016), assim

$$\hat{h}_j \leftarrow \gamma_j \frac{h_j - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}} + \beta_j \tag{53}$$

- **Função de Perda L**

  A função de perda L atua numa rede para diminuir os erros. Considere uma simples instância de treinamento

$$E = \tfrac{1}{2}(y - f(x))^2,$$

  onde, $f(x)$ é a predição da rede obtida de uma unidade de saída e y identificação da classe da instância (neste caso é binário (0 ou 1), o fator $\frac{1}{2}$ sairá quando começar a aplicação de derivadas. Um procedimento de otimização é necessário, chamado de Gradiente Descendente (*Gradient Descent*) que utiliza as informações dos parâmetros da função de ajuste e para usá-lo para encontrar os pesos do perceptron de multicamada, é necessário a derivada do erro quadrado para cada parâmetro de E. Diferenciando a função do erro com respeito a um particular peso w_i temos

$$\frac{dE}{dw_i} = \frac{1}{2}(-1)(y-1)\frac{f(x)}{dw_i} \tag{54}$$

$$\frac{dE}{dw_i} = (f(x) - y)\frac{f(x)}{dw_i} \tag{55}$$

onde, $f(x)$ é a saída do perceptron e x é a soma dos pesos das entradas. A derivada do segundo fator da direita $\frac{f(x)}{dw_i}$, a derivada da *sigmoid* será necessária porque ela está em termos de $f(x)$ então

$$\frac{df(x)}{dx} = f(x)(1 - y) \tag{56}$$

Embora a derivada da função logística esteja em função de x, o termo da diferencial da função de erro está em termos de w_i, pelo motivo de

$$x = \sum_i (w_i a_i) \tag{57}$$

a derivada de $f(x)$ em relação a w_i é

$$\frac{d(fx)}{dw_i} = f'(x)a_i \tag{58}$$

assim a derivada da função de erro fica

$$\frac{dE}{dw_i} = (f(x) - y)f'(x)a_i \tag{59}$$

A Equação 59 dá condições para calcular a mudança do peso w_i causado por um particular vetor **a** (WITTEN *et al.*, 2016). Essa demonstração feita foi para um *perceptron* sem camada oculta, porém para um *perceptron* que possua camada oculta (Figura 33), basta substituir a_i pela saída do i-ésimo unidade escondida gerando

$$\frac{dE}{dw_i} = (f(x) - y)f'(x)f(x_i) \tag{60}$$

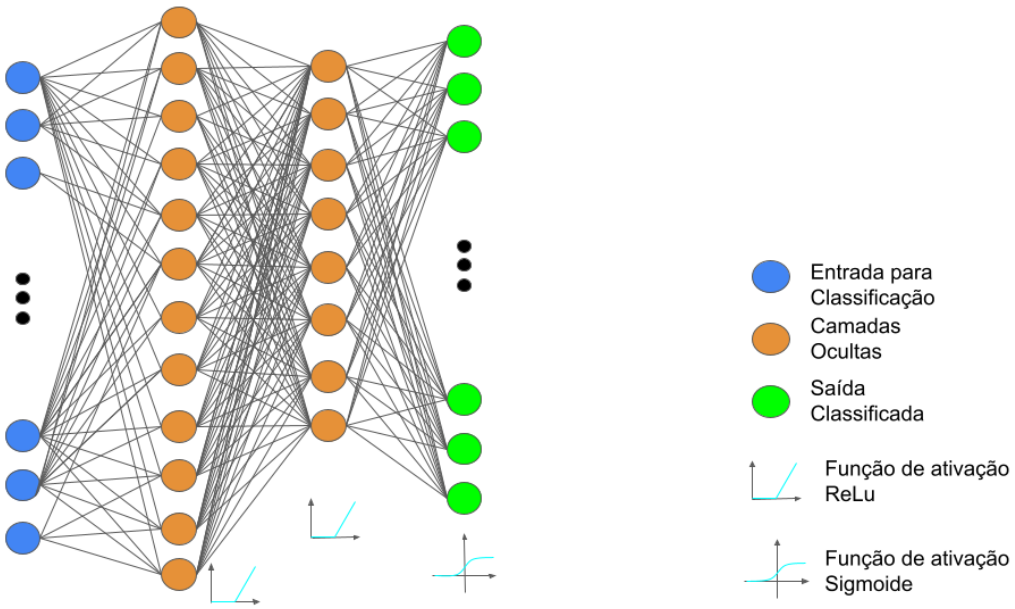


Figura 33 – A função de ativação ReLu é utilizada nas camadas ocultas, neste caso trata-se de uma Rede Feedforward.

Fonte: Elaborado pelo autor, 2022.

Os itens supracitados sugerem atenção com a hiperparametrização de uma rede neural profunda, uma dessas redes que se destaca para a classificação de imagens é a rede neural convolucional que é apresentada na próxima subseção.

As redes neurais de múltiplas camadas possuem um fluxo do sinal que se propaga pela rede para frente e por cada camada, com aplicação de pesos, o *perceptron* tem a função de encontrar os pesos corretos para classificação dos dados, a esse tipo de rede é dado o nome de *perceptron* de múltiplas camadas (Multi-Layer Preceptron-MLP). Este tipo de rede é treinada com o algoritmo de retropropagação de erro (Backpropagation) para ajustar os pesos dos valores transmitidos para toda a rede.

Esse fluxo é sistematizado da seguinte forma, *sentido para frente*: fase na qual os sinais de entrada são recebidos pelos neurônios e são propagados para toda a rede com os pesos fixos; *sentido para trás*: os pesos são ajustados de acordo com a regra de correção de erros que é a resposta atual comparada com a resposta desejada, essa diferença é o sinal de erro quadrado da saída da rede, esse sinal é propagado para trás pela rede e os pesos sinápticos são ajustados para que a resposta esteja cada vez mais próxima da resposta correta (HAYKIN, 1999). A Figura 34 ilustra a complexidade de uma rede de neurônios é maior do que a de uma Rede Neural Artificial, enquanto depende de elementos fisiológicos

a outra depende de recursos computacionais para que os modelos matemáticos possam ser aplicados.

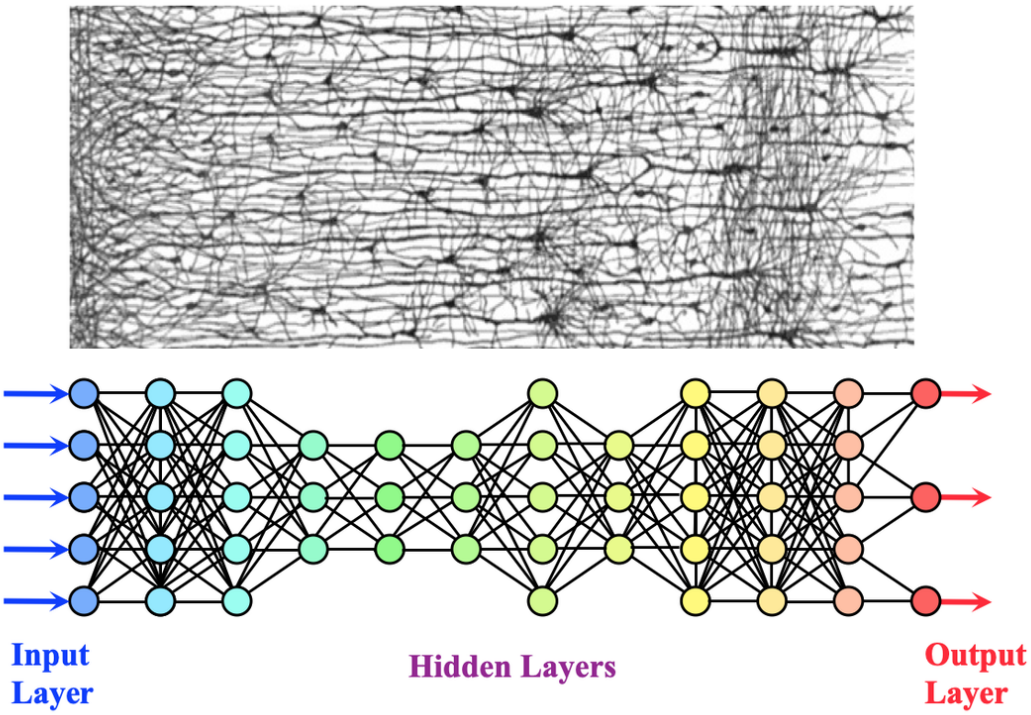


Figura 34 – Comparação entre uma rede neural de neurônios naturais com uma rede neural artificial.

Fonte: Adaptado (BRUNEL; HAKIM; RICHARDSON, 2014).

3.8 Rede Neural Convolucional

A Rede Neural Convolucional (*Convolutional Neural Network-CNN*) foi criada para abordar problemas de classificação de imagens e tem provado grande sucesso para classificar imagens. A imagem atua como o sinal de entrada na rede que a identifica, cria uma matriz quadrada de pixels, mapeia os valores mais altos, seleciona-os e os aplica em uma camada de rede neural profunda onde a saída é o resultado para a classificação da imagem conforme a probabilidade calculada (LECUN *et al.*, 1989).

As etapas do processo de uma rede neural convolucional é apresentada nos itens abaixo de forma detalhada para deixar menos abstrato o processo empregado por este algoritmo na imagem, a Figura 35 ilustra as etapas.

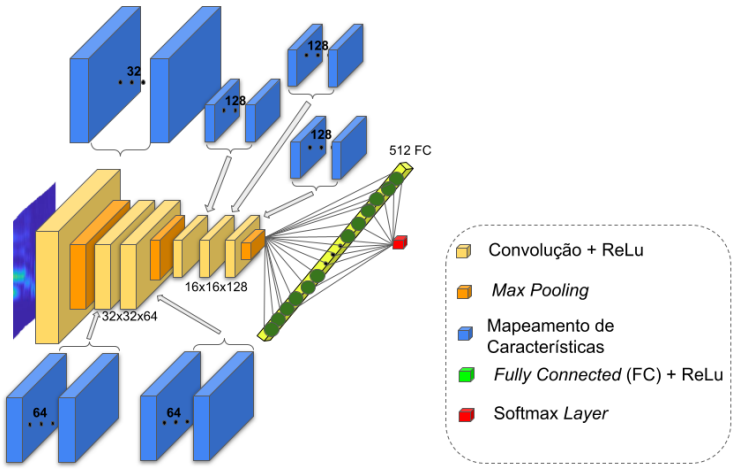


Figura 35 – Exemplo de de convolução, *pooling* e processo de diminuição usado em uma rede neural convolucional (WITTEN *et al.*, 2016).

- **Convolução**

  Uma operação de *convolução* é denotada com o asterisco $\star$ entre duas funções, se uma função inicial $x(t)$, onde t é o valor do tempo medido, e a seja valor de medição t por exemplo, $a = \frac{1}{n}\sum_i^n t_i$, a função $w(a)$ será necessária para os valores de a e aplicando esses valores de $x(t)$, a função $s(t)$ é obtida como segue (GOODFELLOW; BENGIO; COURVILLE, 2016):

$$s(t) = \int x(a)w(t-a)da \tag{61}$$

  que pode ser reescrita como uma operação de convolução contínua por

$$s(t) = (x \star w)(t) \tag{62}$$

  A terminologia em redes convolucionais refere-se ao primeiro argumento (x) da Equação 62 como entrada *input* (imagem) e o segundo argumento (w) como *kernel* (filtro). A saída é referida algumas vezes como mapa de característica (*feature map*). A Equação 62 apresenta uma equação de tempo contínuo, porém para aplicações reais, é necessário uma convolução discretizada (Equação 63):

$$s(t) = (x \star w)(t) = \sum_{\infty}^{a=-\infty} x(a)w(t-a) \tag{63}$$

  Se a convolução usar mais que um eixo no tempo, por exemplo uma imagem I de duas dimensões (2D), a utilização do *kernel* K é dada por

$$S(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{64}$$

A operação é comutativa e como resultado da Eq. 2.46 há uma variação em K, com isto a *Cross-Correlation* é obtida por

$$S(i,j) = (I \star K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{65}$$

- **Filtro de imagem para as camadas convolucionais**

  Não diferentemente, há a necessidade de computar o gradiente para otimizar a rede convolucional. Em uma camada que haja $i = 1, ..., N^l$ filtros de características e os mapas de característica correspondentes, às matrizes de *kernel* convolução $\mathbf{K}_i$ contém pesos movimentados em relação a matriz de peso $\mathbf{W}_i$. Uma função de ativação $act()$ e para cada característica i um fator de dimensionamento g_i e uma matriz de *bias*[4] $\mathbf{B}_i$, o mapa de características são as matrizes $\mathbf{H}_i(\mathbf{A}_i(\mathbf{X}))$ podendo ser visualizado com um conjunto de imagens de mapas de características dada por

$$\mathbf{H}_i = g_i act[\mathbf{K}_i \star \mathbf{X} + \mathbf{B}_i] = g_i act[\mathbf{A}_i(\mathbf{X})] \tag{66}$$

  As derivadas parciais de saída da camada oculta em relação a $\mathbf{X}$ de unidades convolucionais são

$$\frac{\partial L}{\partial \mathbf{X}} = \sum_i \sum_j \sum_k \frac{\partial a_{ijk}}{\partial \mathbf{X}} \frac{\partial \mathbf{H_i}}{\partial a_{ijk}} \frac{\partial L}{\partial \mathbf{H_i}} = \sum_i \frac{\partial a_i}{\partial \mathbf{x}} \frac{\partial \mathbf{h_i}}{\partial a_i} \frac{\partial L}{\partial \mathbf{h_i}} \tag{67}$$

  onde, $\mathbf{D}_i = dL/\partial \mathbf{A}$ é a matriz contendo a derivada parcial da entrada da função de ativação do *element-wise*[5] em relação ao seu valor de pré-ativação para o ith tipo de característica, organizado de acordo com as posições espaciais conforme linha j e coluna k (WITTEN *et al.*, 2016).

- **Agrupamento (Pooling)**

  A função do *pooling* é ajudar a fazer uma representação aproximadamente invariante para pequenas translações da entrada dos dados, isto significa que se houver translação na entrada dos dados e em pequena quantidade, os valores agrupados da saída não mudam (GOODFELLOW; BENGIO; COURVILLE, 2016), mas esses valores de entrada consistem das matrizes $\mathbf{H_i}$ para cada mapa de característica com os elementos h_ijk. Os mapas de características médio *pooling* e máximo *pooling* constituem a matriz $\mathbf{P}_i$ com os elementos $pijk$. Quando o gradiente de "backpropagation" encontrar esses

[4] $bias(\hat{\theta}_m) = \mathbf{E}(\hat{\theta}_m) - \theta)$, aplicada em distribuição de probabilidade

[5] produto de Hadamard

valores, o máximo *polling* será o escolhido e nessas zonas sem sobreposição o gradiente é propagado de volta da camada de *polling* P_i para a camada de características $\mathbf{H_i}$ Witten2016. A perda L pode ser escrita da seguinte forma

$$\frac{\partial L}{\partial h_{i,r_j,c_k}} = \begin{cases} 0, & r_j \neq r_j^*, c_k \neq r_j^*, \\ \frac{\partial L}{\partial p_{ijk}}, & r_j = r_j^*, c_k = r_j^*. \end{cases} \tag{68}$$

onde, rj e c_k são os conjuntos dos índices que codificam as regiões de *pooling* (WITTEN *et al.*, 2016).

3.9 Avaliação de Modelos de Redes Neurais

Avaliar um modelo de Redes Neurais é uma fase importante após o treinamento de uma rede. Dentre os modelos de avaliação, o modelo escolhido foi a Matriz de Confusão.

3.9.1 Matriz de confusão

Um método comum para descrever a performance de um modelo de classificação é a *matriz de confusão*. Para um problema de duas classes, a diagonal dessa matriz apresenta os casos em que foram corretamente previstas, enquanto que, a diagonal inversa representa os casos de erro na previsão. Ela está representada na Tabela 1 e relaciona as contagens das previsões verdadeiras positivas (VP), verdadeiras negativas (VN), falsas positivas (FP) e falsas negativas (FN) de um classificador, conforme descritas as seguir (KUHN; JOHNSON, 2018):

- **VP**: verdadeiro positivo, referem-se às instâncias positivas que foram corretamente previstas pelo classificador;
- **FN**: falso negativo - representam às instâncias positivas que foram incorretamente rotuladas como negativas;
- **FP**: falso positivo - representam às instâncias negativas que foram incorretamente rotuladas como positivas;
- **VN**: verdadeiro negativo, referem-se às instâncias negativas que foram corretamente previstas pelo classificador.

De forma prática VP e VN indicam o quanto o classificador está acertando nas predições, enquanto FP e FN indicam quanto o classificador está errando nas predições.

Tabela 1 – Matriz de confusão para problema de duas classes (KUHN; JOHNSON, 2018).

		Classe Prevista	
		Evento	Não-Evento
Classe Real	Evento (P)	*VP*	*FN*
	Não-Evento (N)	*FP*	*VN*

Essa matriz fornece duas informações estatísticas importantes, uma definida como *Sensibilidade* sendo a Taxa de Valores Positivo (TVP) do modelo que é taxa do evento de interesse que foi previsto corretamente para todas as amostras que o tiveram:

$$\text{Sensibilidade} = TVP = \frac{VP}{P} = \left(\frac{VP}{FN + VP}\right) \tag{69}$$

A *Especificidade* sendo o Total de Verdadeiro Negativo (TVN), referem-se às instâncias negativas que foram corretamente previstas pelo classificador.

$$\text{Especificidade} = TVN = \left(\frac{VN}{FP + VN}\right) \tag{70}$$

Outros valores também são gerados a partir da matriz de confusão. O Valor de Predição Positiva (VPP) é o percentual de instâncias rotuladas como positivas e que realmente são positivas.

$$\text{Valor de Predição Positiva} = VPP = \left(\frac{VP}{VP + FP}\right) \tag{71}$$

O Valor de Predição Negativa (VPN) que é o percentual de instâncias rotuladas como Negativas e que realmente foram classificadas erroneamente como positivas.

$$\text{Valor de Predição Negativa} = VPN = \left(\frac{VN}{VN + FN}\right) \tag{72}$$

O Total de Falso Positivo (TFP) representa a proporção de instâncias negativas que foram incorretamente classificadas como positivas, em relação ao total de negativos (N).

$$\text{Taxa de Falso Positivo} = TFP = \left(\frac{FP}{FP + VN}\right) \tag{73}$$

A Acurácia (ACC) do classificador apresenta a porcentagem de instâncias previstas corretamente pelo classificador como sendo positivas ou negativas.

$$\text{Acurácia} = ACC = \left(\frac{VP + VN}{VP + VN + FP + FN} \right) \tag{74}$$

O Erro de Predição (EP) pode ser entendido como a soma de todas as previsões falsas divididas pelo número total de predições.

$$\text{Erro de Pedição} = EP = (1 - ACC) \tag{75}$$

3.10 Breve Revisão de Trabalhos de Classificação de Estilos Musicais

Dentre os trabalhos realizados para a classificação musical, a de prever os estilos aparece com maior frequência.

Um método utilizado foi a extrair características musicais através de histogramas dos coeficientes Wavelet de Daubechies (*Daubechies Wavelet Coefficient Histograms-DWCH*) e também pela STFT. Os métodos de Machine Learning como SVM, LDA e KNN foram empregados nas bases de dados com 756 músicas com 5 gêneros. A melhor acurácia foi obtida com dois modelos usando os dados do DWCH: 74.9 (SVM), 71.3 (LDA) (LI; OGIHARA; LI, 2003).

Já foi utilizado um método conhecido como "Discrete Wavelet Packet Transform" (DWPT) para extração de características musicais. Os autores aplicaram a DWPT em uma base de dados com 200 músicas, com 5 gêneros musicais. O algoritmo utilizado para a classificação foi o KNN, a acurácia não foi a melhor que a de Li (2003) (GRIMALDI; CUNNINGHAM; KOKARAM, 2003).

Dada a versatilidade da aplicação da STFT esta técnica tem sido utilizada para a geração de espectrogramas para utilizar as imagens em uma rede neural convolucional juntamente com o método SVM para classificar estilos musicais. A sua base de dados continha 2348 músicas e obteve um boa performance de classificação usando Rede Neural Convolucional com SVM, alcançou uma acurácia de 92%, melhor resultado quando comparado com outros supracitados. Entretanto, quando foi utilizado somente a Rede Neural Convolucional classificar individualmente os estilos musicais, não apresentou o melhor resultado (COSTA; OLIVEIRA; SILLA, 2017).

Neste capítulo foram apresentados todos os conceitos baseados na revisão bibliográfica deste trabalho, inicialmente os conceitos básicos sobre música, os fundamentos da

transformada de Fourier, a ideia da transformada de Wavelet e de seus janelamentos,a avaliação do modelo pela acurácia através da matriz de confusão.

Foi apresentada uma subárea da aprendizagem de máquina conhecida como redes neurais, foram vistos conceitos, o funcionamento de uma rede neural profunda e como esse tipo de rede efetua a classificação com as suas técnicas que melhoram o modelo via *backpropagation*, gradiente descendente, gradiente estocástico descendente, gradiente estocástico descendente de mini-lotes, funções de perda. Foi revisado o conceito e funcionamento de uma Rede Neural Convolucional e uma breve revisão de trabalhos de classificação de estilos musicais.

4 Metodologia

Este capítulo a seção 4.1 é dedicada para descrever a base dados utilizada neste trabalho, por exemplo, a sua origem, tamanho dessa base de dados e a sua qualidade. Na seção 4.2 explicamos o processo de criação das imagens a partir dos arquivos de áudio com a Transformada Wavelet. A subseção 4.2.3 é apresentado o modo de como esses dados foram organizados antes do treinamento da Rede Neural Convolucional (CNN). Nesta seção 4.3 é detalhatado a configuração a Rede, suas camadas convolucionais e hiperparâmetros. Na seção 3.9 é apresentado as formas de avaliação de uma Rede Neural de Aprendizagem Profunda e, finalizando, na seção 4.4 é mostrado as métricas avaliadas com a acurácia necessária que a Rede fosse capaz de efetuar o maior percentual de previsões de estilos musicais possível. E, por fim, na seção 4.5 comparamos esse método com o outro apresentou valores menos eficientes.

4.1 Base de Dados Utilizada

A base de dados utilizada neste trabalho é uma coletânea de áudios disponibilizada na Internet[1] que foi inicialmente manipulada por Zanetakis *et al.* para o estudo de Recuperação de Informação Musical (TZANETAKIS; COOK, 2002), desde então essa base de dados continua sendo utilizada para estudos acadêmicos (GHILDIYAL; SINGH; SHARMA, 2020).

O seu conteúdo contém 1000 arquivos de áudio, separados por quantidade igualmente distribuída pelos 10 estilos: *Blues, Classical, Country, Disco, Hip Hop, Jazz, Metal, Pop, Reggae e Rock*, conforme ilustrado na Figura 36. Cada diretório contém 100 músicas diferentes, porém há um outra versão para a mesma música, por exemplo, no estilo Clássico a música “Rhapsody in Blue” de George Gershwin está no arquivo número 44 e possui uma outra versão da mesma música no arquivo 48. No total, há 13 eventos semelhantes a este (STURM, 2013).

[1] https://github.com/marsyas/marsyas

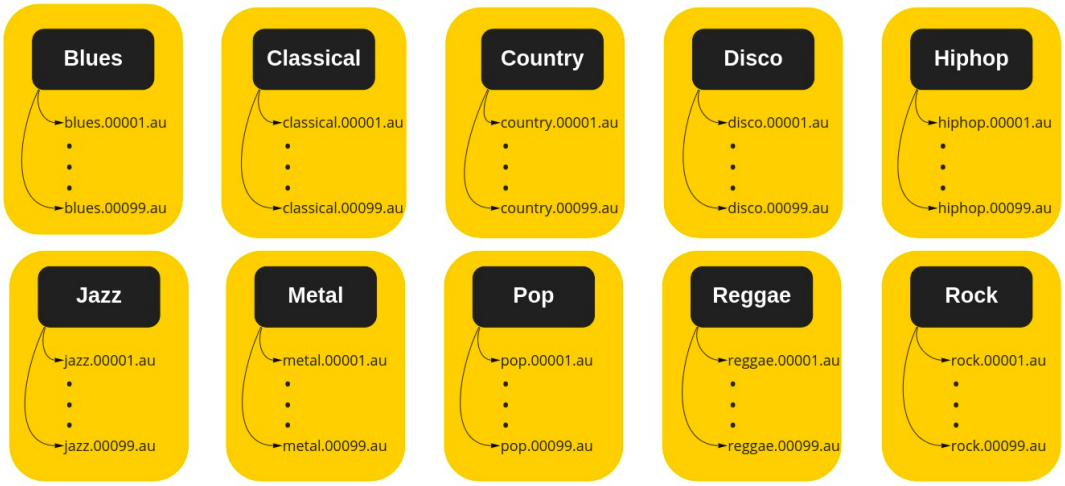


Figura 36 – Estrutura dos diretórios com os arquivos de áudio originais.

Fonte: Elaborado pelo autor, 2022.

O formato de compressão dos arquivos disponibilizados para este trabalho é o AU[2] *Audio File Format*, os detalhes dessa compressão nos arquivos estão descritos na Tabela 2. Esses arquivos foram convertidos para o tipo de compressão WAV sem perda de qualidade ou propriedades para se adequar aos códigos criados neste trabalho e ao tratamento dos arquivos à geração das imagens que está descrito na seção 4.2.

Tabela 2 – Detalhes do tipo de compressão *AU* dos arquivos da base de dados utilizada no trabalho.

Parâmetro	**Valor**
Canais	1
Taxa de Amostragem	22050 Hz
Codificação	16 bits
Modulação	PCM
Duração	30.01 s
Quantidade de Amostras	661794

Fonte: Elaborado pelo autor, 2022.

4.2 Organização dos Dados

O sistema operacional, versão da distribuição, versão de *kernel*, linguagens de programação utilizadas neste trabalho estão detalhados na Tabela 3. A importância disto é garantir que haja a reprodução do presente trabalho com os dados utilizados, através destas informações.

[2] https://docs.oracle.com/cd/E36784$_0$1/*html*/*E*36882/*au* − 4.*html*

Tabela 3 – Especificações do computador e linguagens de programação.

Configurações PC	**Processador**
16 GB de memória RAM	Intel Core i7 oitava geração
Sistema Operacional	**Distribuição/Versão/Kernel**
Linux	Ubuntu/18.04/4.15.0-193-generic
Graphics Processing Unit (Versão)	**Linguagens de programação (Versão)**
NVIDIA GEFORCE GTX	Matlab (r2021.a)
(NVIDIA-SMI 470.141.03) (Driver Version: 470.141.03) (CUDA Version: 11.4)	Python (3.7.3) GCC (7.3.0)

Fonte: Elaborado pelo autor, 2022.

Algumas etapas antecederam a geração das imagens para o treinamento da Rede Neural Convolucional a partir de um sinal de áudio. Este sinal é uma série temporal e dela é extraído os coeficientes da Transformada Wavelet; Estes coeficientes serão necessários para gerar os Espectros de Potência Wavelet, que são as imagens geradas para, por fim, organizá-las para o treinamento e testes da Rede Neural.

O processo de organização dos dados seguiu três etapas, descritas em subseções e ilustrada da figura Figura 37: a primeira etapa da subseção 4.2.1 extrai os coeficientes aproximados da Transformada Wavelet da série temporal dos arquivos de áudio, para que a partir destes coeficientes fossem criadas novas séries temporais; a segunda etapa na subseção 4.2.2 utilizou das novas séries temporais para gerar os espectros de potência, ou escalogramas, que foram as imagens que treinaram e testaram a Rede Neural Convolucional; e na terceira etapa na subseção 4.2.3 separamos os dados para treinamento e para testes.

Figura 37 – Fases para extração, geração e organização para treinar a Rede Neural Convolucional.

Fonte: Elaborado pelo autor, 2022.

4.2.1 Geração dos Coeficientes

Antes da geração dos coeficientes da Transformada Wavelet, os arquivos tiveram a sua compressão convertida de *au* para *wav* pela razão do código de programação deste trabalho estava estruturado neste formato. Vale ressaltar que as propriedades dos arquivos foram mantidas exatamente as mesmas após a conversão (Figura 38).

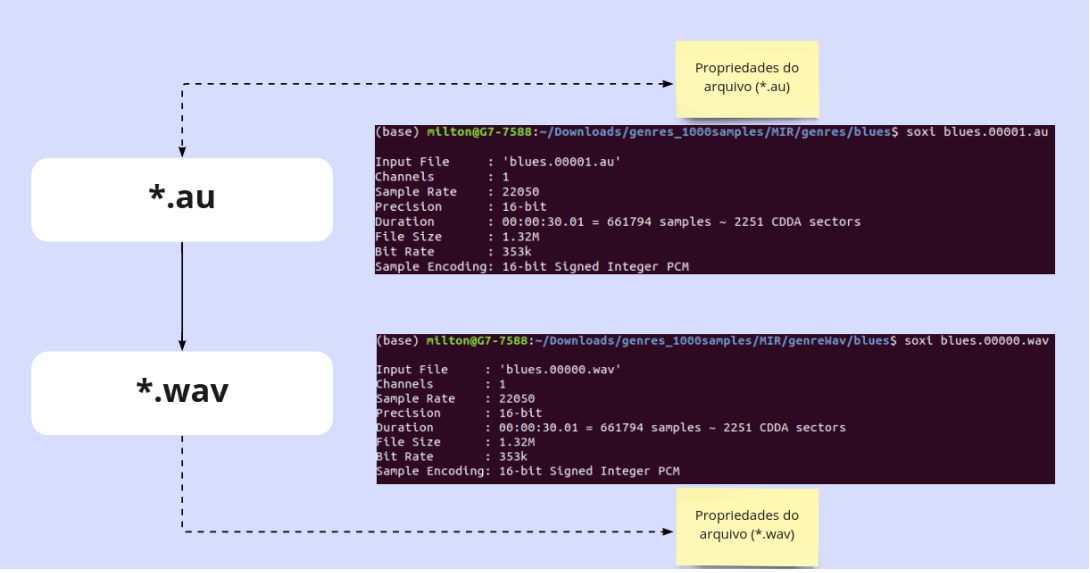


Figura 38 – Esta conversão para *wav* garantiu os mesmos valores do tipo de compressão *wav*.

Fonte: Elaborado pelo autor, 2022.

Os coeficientes aproximados foram providos a partir da série temporal de cada aúdio e esta escolha deveu-se a duas constatações: 1) foi utilizado a Wavelet D1, da família Daubechies, equivalente a Wavelet Haar por causa de sua simplicidade (ADDISON, 2002); 2) esse tipo de Wavelet permitiu capturar a maior parte do sinal com o coeficiente aproximado se comparado com a do coeficiente detalhado, confome ilustrado na Figura 39.

A base teórica dos coeficientes da Transformada Wavelet foi revisada na seção 3.5.

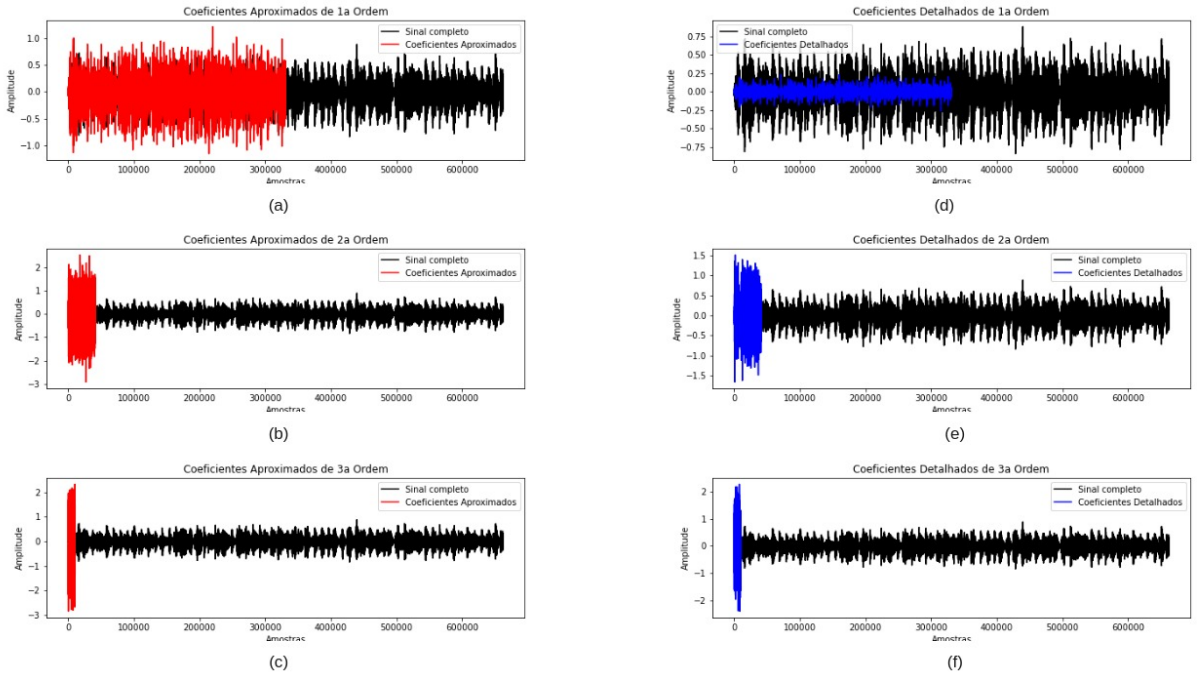


Figura 39 – (a) Coeficientes aproximados de ordem 1. (b) Coeficientes aproximados de ordem 2. (c) Coeficientes aproximados de ordem 3. (d) Coeficientes detalhados de ordem 1. (e) Coeficientes detalhados de ordem 2. (f) Coeficientes detalhados de ordem 3.

Fonte: Elaborado pelo autor, 2022.

Há disponível uma base de dados na Internet conhecida como MINIST[3] - dígitos escritos a mão – que possui 60.0000 imagens para treinamento e 10.000 para testes e tem sido utilizada para servir de referência para testar as Redes Neurais Convolucionais e ela tem mostrado que a quantidade é importante para evitar o *Overfitting* (WITTEN *et al.*, 2016).

Com isto, encontramos uma limitação com o tamanho da base de dados, somente 1.000 arquivos e sabendo que uma porção seria retirada para a previsão antes do treinamento, o restante dividida entre treinamento e validação e, com este cenário, coletamos os coeficientes aproximados de ordem 1 (cA1), 2 (cA2) e 3 (cA3) (Figura 40).

[3] http://yann.lecun.com/exdb/mnist/

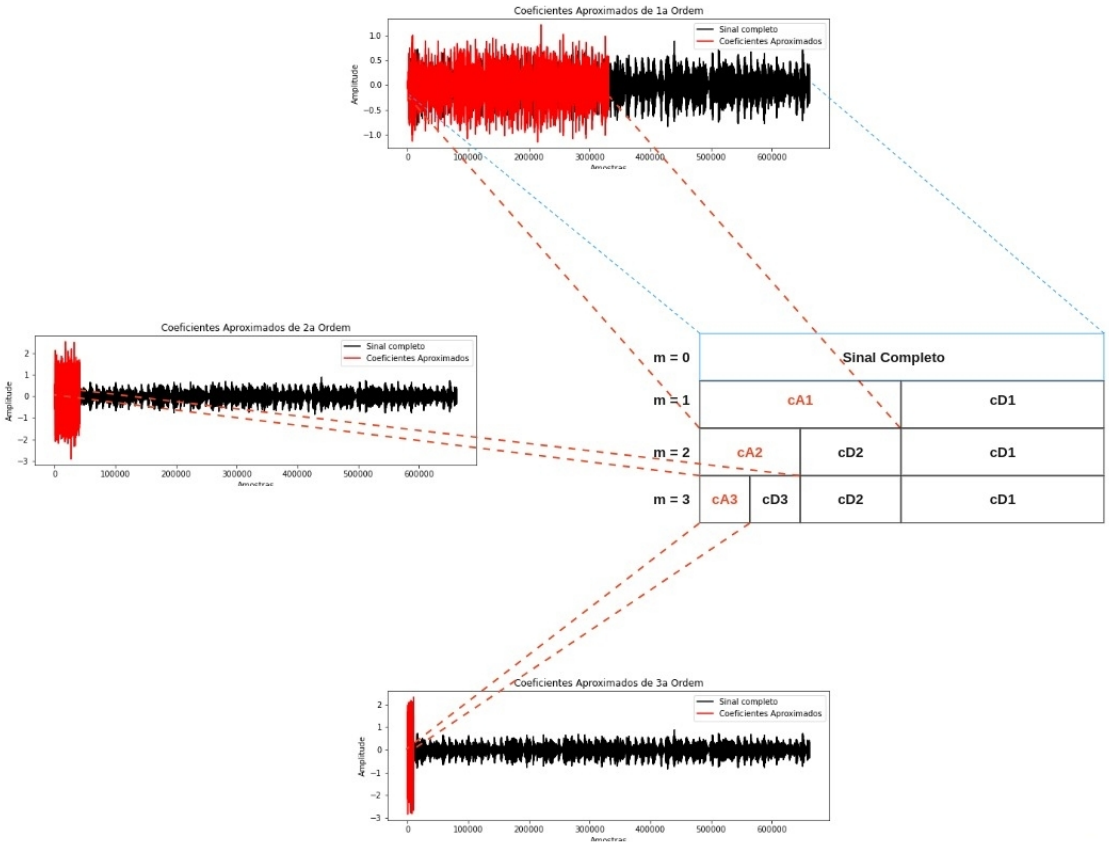


Figura 40 – No sinal de áudio aplicamos a Transformada Wavelet Daubechies *db*1 (Wavelet Haar) para gerar os coeficientes aproximados, ou seja, mais séries temporais.

Fonte: Elaborado pelo autor, 2022.

Aplicando a Wavelet *db*1 (Daubechies) para Sinais Discretos, os coeficientes aproximados cA1, cA2 e cA3 geraram 1.000 novos arquivos por coeficiente, totalizando 3.000 novas séries temporais. Deste modo, triplicamos a quantidade de séries temporais. A próxima seção contém os detalhes de como foi aumentado a quantidade de imagens de 3.000 para 9.000 e como foi o processo para transformar essas séries temporais em imagens dos espectrogramas para treinar a Rede Neural Convolucional.

4.2.2 Geração dos Espectrogramas

Nesta subseção apresentamos o método utilizado para gerar a quantidade de séries temporais para que, a partir delas, fosse possível aumentarmos a nossa base de dados para aplicar a Transformada Wavelet de Sinal Contínuo e, assim, foi possível gerar uma base de dados com 9.000 imagens

A série temporal (ST) de cada coeficiente foi dividida em três partes iguais para que o código da geração das imagens (espectros de potência) fossem gerados. O processo do algoritmo seguia esse fluxo: 1) acessar a ST do coeficiente aproximado cA_n; 2) ler o comprimento da ST; 3) criar uma variável com o valor de 1/3 da ST; 4) aplicar a Transformada Wavelet Contínua (CWT); 5) Repetir isto 3.000 vezes.

O algoritmo 1 demonstra o pseudo-código para a extração dos coeficientes. Este processo gerou 9.000 séries temporais para que fossem utilizadas para a criação das imagens contendo dos espectros de potência, ilustrado na Figura 41.

Algoritmo 1 Extração dos Coeficientes Aproximados do Sinal de Áudio

Entrada:
Fs: Frequência de amostragem do sinal de áudio
genres: Estilos musicais dos áudios
g: Repetidor para identificar os estilos musicais
filename: Repetidor para identificar os áudios de cada estilo musical
songname: Nome de cada arquivo de áudio
Saída:
coeffs: Vetor com os valores dos coeficientes aproximado e detalhado da Wavelet db1
level: grau da geração dos coeficientes variando de 1 a 3
cA: Vetor com os valores dos coeficientes aproximados

```
1 início
2     para g in genres faça
          /* Listar diretórios de cada estilo                    */
3         para filename in lista de estilos faça
              /* Listar arquvos de cada estilo                   */
4             coeffs ← level ∈ 1, 2, 3
              cA ←coeffs(Coeficiente Aproximados)
              arquivo←cA;
              fim
5         fim
6     fim
```

Seguem as principais variáveis do algoritmo 1:

- $coeffs$: vetor que armazena os dois tipos de coeficientes da Transformada Wavelet Discreta.
- cA: vetor com os coeficientes aproximados de cada iteração com o *level*.
- *arquivo*: arquivo texto com os cAs, que é uma Série Temporal, que será utilizado pelo código para gerar as imagens.

Os espectrogramas foram gerados através da CWT Morse Generalizada, revisada na seção 3.4. O Apêndice B apresenta o código desta finalidade e abaixo segue seu pseudo-código.

Algoritmo 2 Geração de Espectrogramas

Entrada:
Fs: Frequência de amostragem do sinal de áudio
genres: Estilos musicais dos áudios
fb: Conjuntos de parâmetros para gerar o espectrograma
frq: Frequência do sinal
Saída:
cfs: Coeficientes da Transformada Wavelet Morse Generalizada
songname: Nome de cada arquivo de áudio
wt: Transformada Wavelet Morse Generalizada

```
2 início
3     para fb in genres faça
          /* Percorrer todas as séries temporais no formato TXT          */
4         para wt in fb faça
              /* Gerar os coeficientes                                   */
5             cfs, frq ← fb
              figura←(intervalor de pontos/Fs), frq, abs(cfs);
              fim
6         fim
7     fim
```

Este processo criou 9.000 imagens com espectrogramas, o fluxo desse processo está ilustrado na Figura 41 e a Tabela 4 contém os intervaloes para cada ordem dos coeficientes aproximados.

É possível verificar que a série temporal **cA1** foi dividida em três partes iguais e nela foi aplicada a CWT para gerar os espectrogramas. Neste exemplo, são os espectros do áudio blues.00000.wav

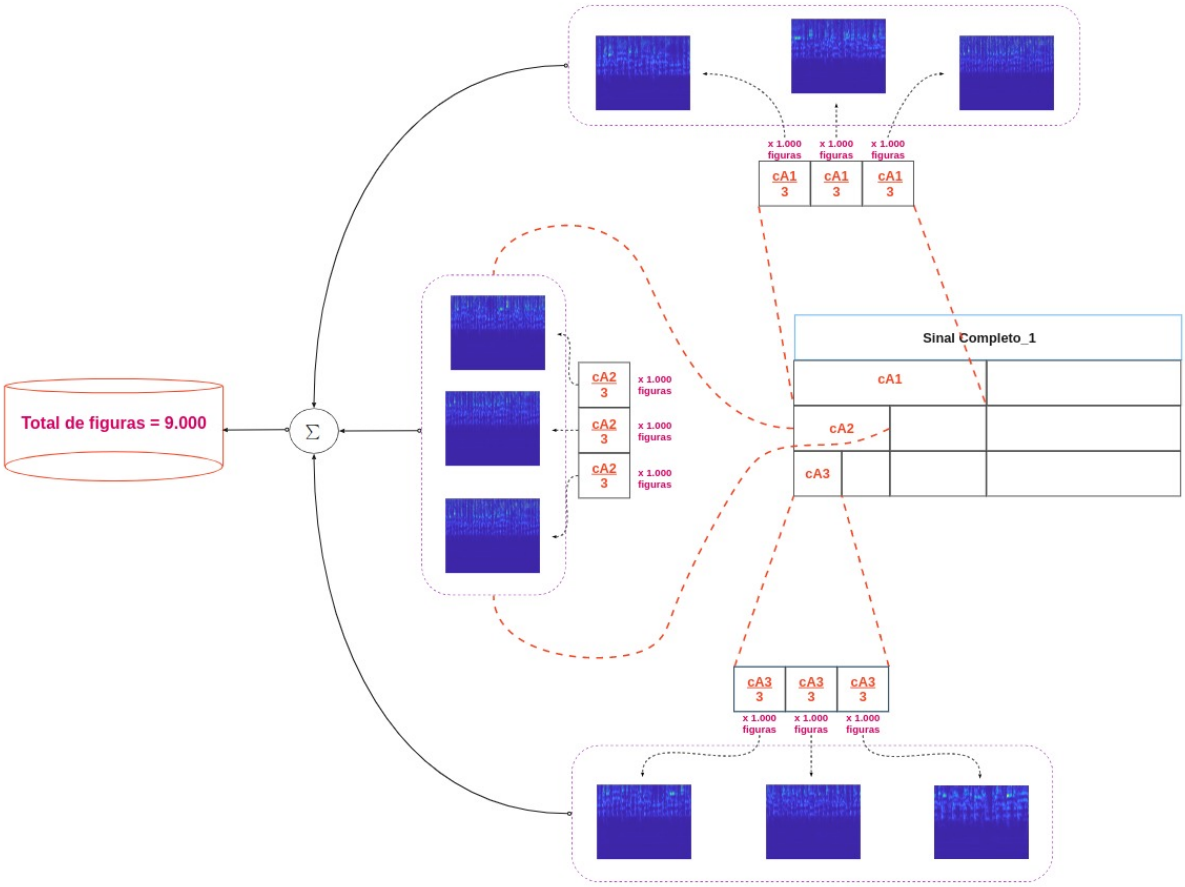


Figura 41 – Exemplo da divisão da série temporal cA1 em três partes iguais.

Fonte: Elaborado pelo autor, 2022.

Tabela 4 – Intervalos para a geração dos Espectros de Potência em função de cada Coeficiente Aproximado gerado. Em cada ordem o total de pontos foi dividido em três partes iguais, onde cada ordem posterior possui a metade de pontos da ordem anterior, exceto os Coeficientes Aproximados cA3 que ficou muito próximo da metade.

	1º Intervalo de Pontos	**2º Intervalo de Pontos**	**3º Intervalo de Pontos**	**Total de Imagens**
cA1	1 a 82600	82601 a 165200	165200 a 330400	3.000
cA2	1 a 55066	55067 a 110134	110135 a 165200	3.000
cA3	1 a 27562	27563 a 55125	55126 a 82688	3.000

Fonte: Elaborado pelo autor, 2022.

Uma vez que a base de dados foi aumentada em nove vezes, o próximo passo foi organizar os dados para treinamento e teste da Rede Neural e, uma outra, para serem previstos os seus estilos musicais por esta Rede.

A subseção 4.2.3 traz o método utilizado para escolher as figuras de modo que não houvesse viés, por exemplo, escolher sempre as 10 primeiras imagens sequenciais de cada estilo para serem testadas.

4.2.3 Dados para Treinamento e Testes

A Rede Neural Convolucional utilizada neste trabalho foi baseada na arquitetura VGG16[4] (SIMONYAN; ZISSERMAN, 2014), este tipo de arquitetura possuir uma grande quantidade de parâmetros e, por ser uma arquitetura conhecida e testada na literatura acadêmica para tratamento de imagens, ela foi utilizada como referência (HAYKIN; VEEN, 2002).

A arquitetura VGG16 já foi adaptada para uma arquitetura menor, chamada de "SmallVGGNet", porém esta foi dimensionada para uma arquitetura que atendesse os recursos computacionais utilizados neste trabalho. Os seguintes itens abaixo foram alterados da arquitetura "SmallVGGNet" para compor a Rede utilizada neste trabalho: a) o dimensionamento das imagens de entrada na Rede foram alterados de 256x256 para 64x64 pixels; b) as camadas convolucionais foram aumentadas de 5 para 6; c) as camadas de *Max Pooling* foram diminuídas de 5 para 3 .

A Figura 42 ilustra essas adaptações. As figuras tiveram a quantidade de pixels diminuída e a quantidade de camadas convolucionais segue uma progressão aritmética de razão 1 a cada camada de *Max Pooling*. Há somente uma camada *Full Connected* conectando todos os seus 512 neurônios com a última camada de *Max Pooling* e com a última camada de ativação de probabilidades (*Softmax*) para finalizar a classificação

[4] https://www.robots.ox.ac.uk/ vgg/research/very$_d$*eep*/

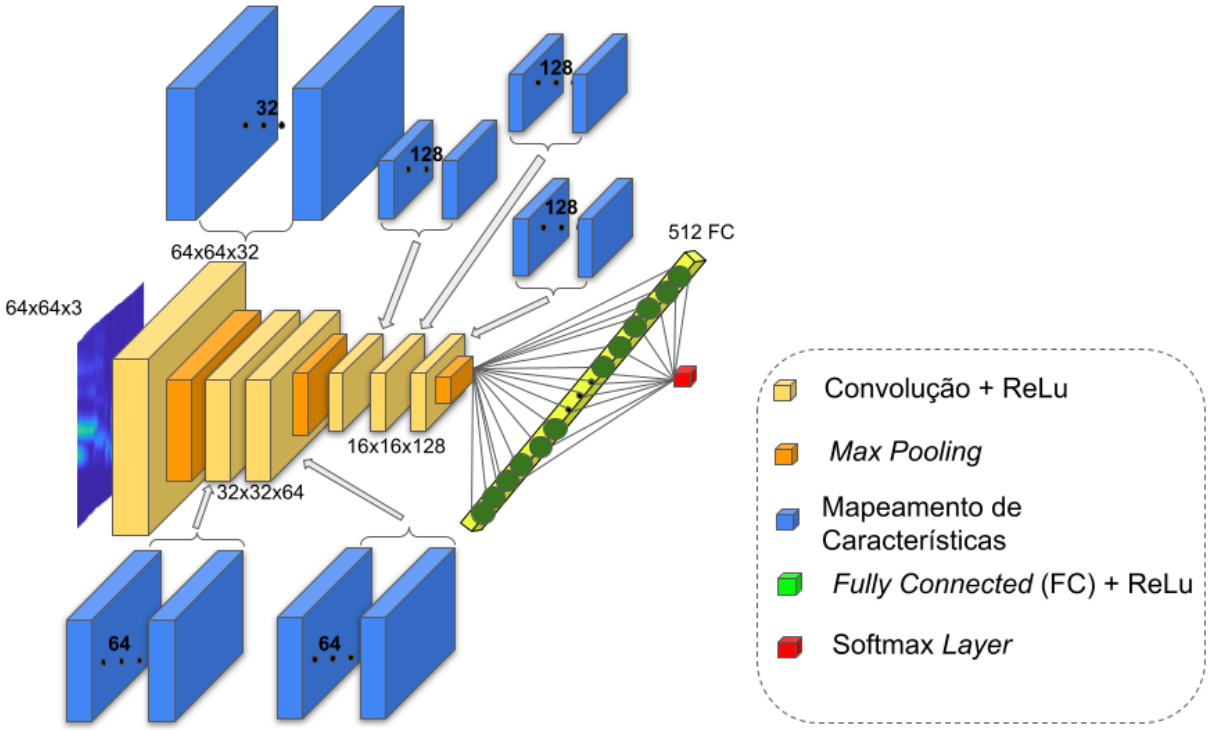


Figura 42 – Rede Neural Convolucional "SmallVGGNet"adaptada para este trabalho.
Fonte: Adaptado (KHADEMI; SIMON, 2019).

Toda Rede Neural precisa ter uma base de dados para treinamento e testes. Criamos grupos de figuras de treinamento para compararmos as performances da Rede Neural.

Primeiramente, 10 % (900) das figuras da base dados foram escolhidas aleatoriamente, sem repetição, antes da escolha dos grupos de testes. Essas figuras escolhidas não se repetiram na seleção aleatória para teste.

O 1° grupo teve 2700 (30%) de figuras escolhidas aleatoriamente, sem repetição, para treinamento.

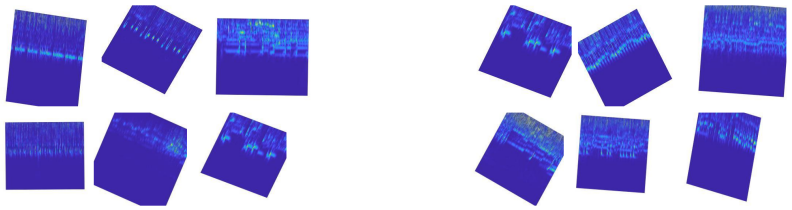
O 2° grupo teve 6300 (70%) de figuras escolhidas aleatoriamente, sem repetição, para treinamento.

O 3° grupo teve 8100 (90%) de figuras escolhidas aleatoriamente, sem repetição, para treinamento.

Foi utilizado a técnica de aumento de imagens pelo recurso conhecido como *geração sintética de imagens* ((HAYKIN; VEEN, 2002), p. 437) que é um recurso atuante com o *Sthocastic Gradient Descendent* (SGD) e, por ser um hiperparâmetro, o seu resultado implica na parametrização de treinamento ((SIETSMA; DOW, 1991). Este recurso faz rotações, ampliação, redução, recorte e inversão da imagem, conforme a ilustração da

Figura 43. Utilizamos os valores padrão da biblioteca de Tensorflow[5] – exceto para o valor de rotação que foi utilizado 30 ao invés de 20.

A documentação da biblioteca informa que cada característica da imagem, por exemplo, a rotação da imagem, é feita 9 vezes a cada loop (fim de cada época). Um exemplo da consequência disto é que treinamos a Rede com 8100 imagens e cada uma delas sofreu esse tratamento sintético.



(a) Alterado para 30 rotações por imagem do estilo Blues

(b) Alterado para 30 rotações por imagem do estilo Clássico

Figura 43 – Rotações das imagens de dois estilos musicais diferentes.

Fonte: Elaborado pelo autor, 2022.

[5] *https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator*

4.3 Treinamento da Rede

O treinamento dessa Rede Neural Convolucional para cada grupo teve 75% das figuras utilizadas para treinamento e 25% para validação. A Tabela 5 apresenta essas quantidades.

Tabela 5 – Quantificação de Imagens para Treinamento e Testes da Rede Neural Convolucional.

Quantidade de figuras	**Quantidade para Treinamento**	**Quantidade para Validação**
2.700	2.025	675
6.300	4.725	1.575
8.100	6.075	2.025

Fonte: Elaborado pelo autor, 2022.

Todos os parâmetros e hiperparâmetros foram deixados constantes, exceto o número de épocas que limitado devido a capacidade computacional de nosso ambiente de trabalho, os valores estão detalhados na Tabela 6.

Tabela 6 – Hiperparametrização de treinamento da Rede Neural Convolucional com 1.000 épocas. A coluna de Gerador de Imagens apresenta as variáveis escolhidas para tratar as imagens com um tratamento dado por um módulo da biblioteca Tensor Flow. Este hiperparâmetro gera mais imagens, por exemplo, invertendo a imagem modo.

Quantidade de Figuras	**Taxa de Aprendizagem (learning rate)**	**Redimensionamento de Imagens (Pixels)**	**Gerador de Imagens**	**Épocas**
2700	0,01	64 x 64 x 3	Rotação, Deslocamento em Largura, em Altura, Corte da Imagem, Ampliação, Preenchimento de pontos	1.000
6300	0,01	64 x 64 x 3	Rotação, Deslocamento em Largura, em Altura, Corte da Imagem, Ampliação, Preenchimento de pontos	1.000
8100	0,01	64 x 64 x 3	Rotação, Deslocamento em Largura, em Altura, Corte da Imagem, Ampliação, Preenchimento de pontos	1.000

Fonte: Elaborado pelo autor, 2022.

Segue abaixo o pseudocódigo do treinamento da Rede Neural Convolucional.

Algoritmo 3 Treinamento da Rede Neural Convolucional SmallVGGNet

Entrada:
pacotes: Pacotes necessários para rodar o Tensor Flow por trás da plataforma Keras
genres: Imagens com Espectrogramas
imagePaths: Escolha de imagens
EPOCHS: Épocas de treinamento
CONV: Camada de Convolução
ReLu: Função de Ativação na saída de cada CONV
Saída:
FC: Rede *Full Connected*
Softmax: Função de Ativação saída da Rede
model.save: Salvar o modelo e serializá-los para o disco

```
3 início
    /* Carregar pacotes                                          */
4   para imagePaths in genres faça
      /* Carregar e redimensionar as imagens                     */
5     para CONV in imagePaths faça
6       CONV ← RelU
        FC ← Softmax
        /* EPOCHS = 1000 repete 10 vezes */
7       model.save ←Softmax_Final;
        fim
8     fim
9   fim
```

4.4 Métricas de Desempenho

O desempenho da Rede Neural Convolucional foi medido pela Matriz de Confusão avaliando o melhor resultado de acurácia obtido após dez treinamentos executados para cada Rede de 1.000 épocas cada, por exemplo, o 1° grupo com 2.700 figuras teve a sua Rede treinada e validada dez vezes com os valores listados na Tabela 6. A Rede com a melhor acurácia previu os estilos de 900 figuras (10 %) que não participou dos grupos de treinamento.

Os valores de cada treinamento foram armazenados em arquivo para facilitar a sua utilização sem precisar carregar toda a Rede Neural Convolucional novamente. Provemos isto salvando os resultados utilizando o módulo "model.save" da biblioteca Keras[6] e serializamos esses Bits através da biblioteca *Pickle*[7] para executar os resultados com maior rapidez.

[6] https://keras.io/api/models/model_saving_apis/
[7] https://docs.python.org/2/library/pickle.html

4.5 Método Comparativo

A Figura 44 ilustra o fluxo cada Rede Neural Convolucional treinada e validada com cada base de dados e com o número fixo de épocas. E a geração de arquivos em texto para análise dos resultados. A cada treinamento da rede, feita dez vezes, os resultados obtidos foram usados para previsão dez vezes, com cada estilo musical.

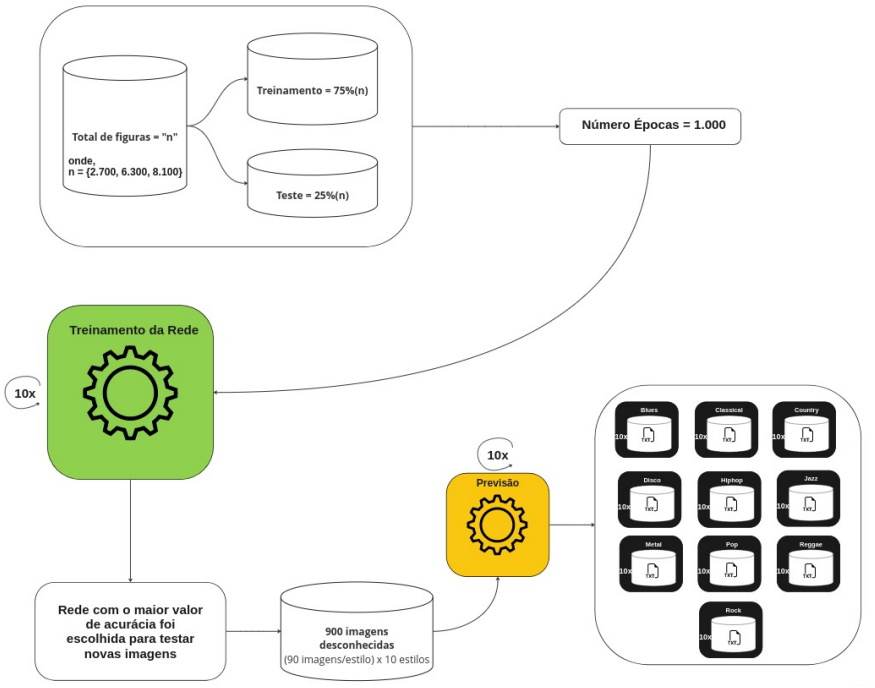


Figura 44 – Fluxo para o treinamento da Rede Neural Convolucional.

Fonte: Elaborado pelo autor, 2022.

Após as métricas de desempenho serem avaliadas, as Redes com o melhor valor de acurácia foram selecionadas para serem testadas. Comparamos o percentual de acerto de previsibilidade em cada grupo de imagens a fim de identificar em qual deles houve o melhor resultado.

Criamos um novo conjunto de testes para comparar os resultados com os obtidos até aqui, nós repetimos o mesmo método com a metade do número da camada de mapeamento de características, conforme ilustrado na Figura 45.

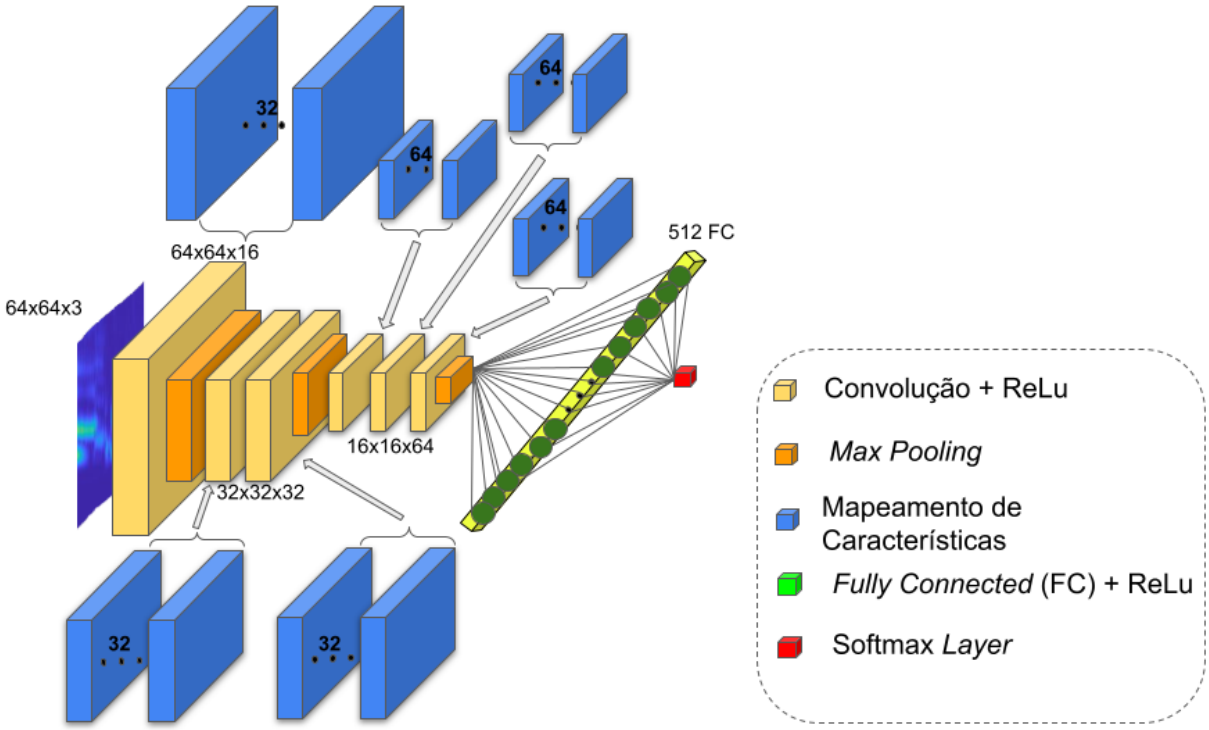


Figura 45 – Essa Rede foi treinada com metade das camadas de mapeamento de características se comparada com a Rede que é objeto principal deste trabalho.

Fonte: Adaptada de (KHADEMI; SIMON, 2019).

Portanto, no Capítulo 5 será apresentado os resultados obtidos através do método descrito até aqui, comparando o nosso método contra o método contendo a série temporal inteira do sinal e também comparar o resultado com uma rede neural tradicional com camadas ocultas.

5 Resultados

O presente capítulo tem por objetivo apresentar os resultados obtidos pelo treinamento da Rede Neural Convolucional, baseada na rede VGG16, com as imagens com os espectros de potência geradas pelos coeficientes da Transformada Wavelet. A teoria desta ferramenta foi revisada na seção seção 3.4 e na seção 3.5.

5.1 Resultados Gerais

Esses resultados iniciais foram obtidos após testes com mudanças em parâmetros como, por exemplo, o período da série temporal do áudio que seria utilizada para gerar as imagens e o dimensionamento das imagens.

A Tabela 7 apresenta os valores que foram essenciais para o avanço e conclusão deste trabalho.

Tabela 7 – A coluna "Conjunto de Ações" de 1 a 3 teve uma valor de acurácia de treinamento da Rede menor que 1% e a previsão do estilo musical valor menor igual a 10 %, considerando que as imagens foram geradas a partir da série temporal inteira do sinal e também sem o tratamento sintético das imagens (Figura 43) as tentativas ficaram em torno desses valores. A partir dos Conjuntos de Ações 4 e 5, foi possível notar o maior valor de acurácia e com melhor previsão de novas imagens usando a segmentação do sinal pelos coeficientes aproximados.

Conjunto de Ações	**Período [s]**	**Freq. Amostr. [Hz]**	**Pixels**	**Canais**	**Quant. Imagens para Treinamento**	**Camadas Convol.**	**Learning Rate**	**Épocas**	**Acurácia (F1-score)**	**Previsão de Acerto 20 Imagens**
1	30	22500	128x128	3	1000	6	0,02	700	0,08	1/20 = 5/%
2	30	22500	64x64	3	1000	6	0,01	300	0,38	2/20 = 10/%
3	30	22500	64x64	3	3000	6	0,01	600	0,50	2/20 = 10/%
4*	"3x 10"	22500	64x64	3	3000	6	0,01	1.500	0,73	14/20 = 70/%
5*	"3x 10"	22500	64x64	3	8100	6	0,01	4.000	0,85	14/20 = 70/%

Fonte: Elaborado pelo autor, 2022.

Neste gráfico da Figura 46 que contém o acerto de previsões para cada grupo de dados (30%, 70% e 90%), não ficou agrupado com uma tendência organizada.

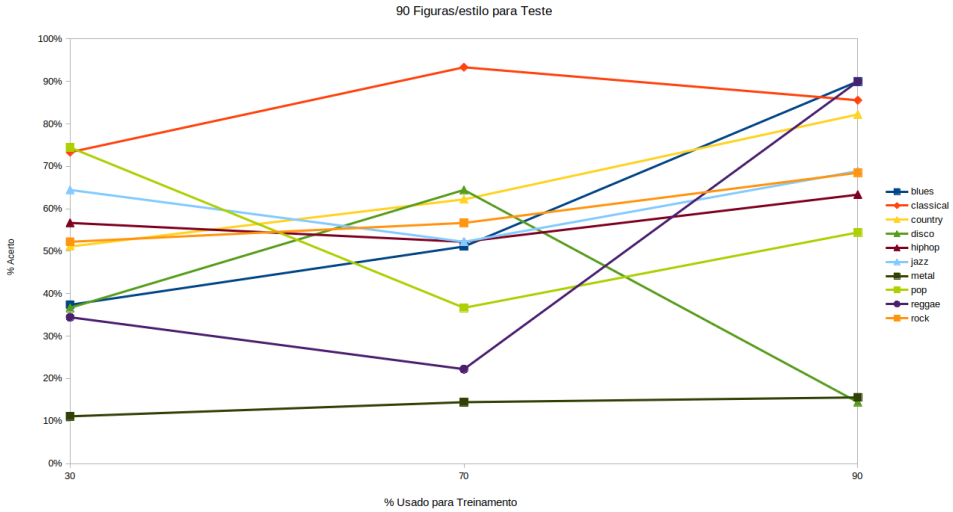


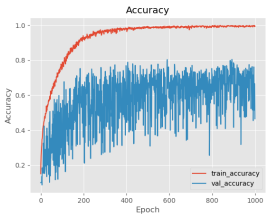
Figura 46 – Resultados sem aleatoriedade para a escolha dos dados para treinamento e validação.

Fonte: Elaborado pelo autor, 2022.

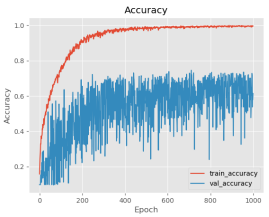
5.2 Principais Resultados

Os resultados abaixo apresentam os valores das acurácias de treinamento e da validação. Para cada conjunto de imagens, 30% (2.700), 70% (6.300) e 90% (8.100), as Redes foram treinadas 10 vezes com 1.000 épocas cada uma com uma taxa de aprendizagem de 0,01.

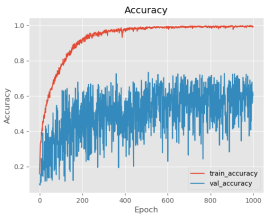
Segue a sequência de treinamento com 2.700 imagens:



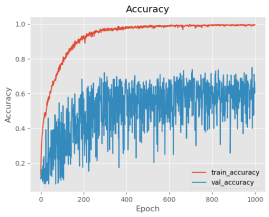
(a) Treinamento 1 com 1.000 épocas



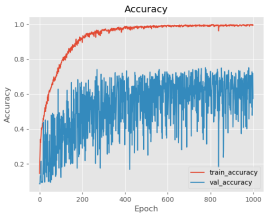
(b) Treinamento 2 com 1.000 épocas



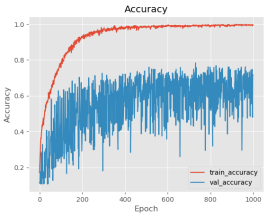
(c) Treinamento 3 com 1.000 épocas



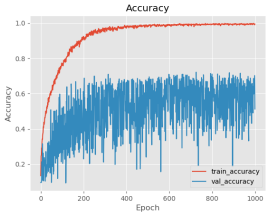
(d) Treinamento 4 com 1.000 épocas



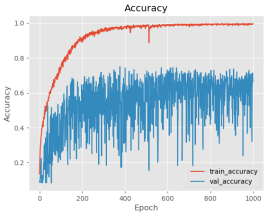
(e) Treinamento 5 com 1.000 épocas



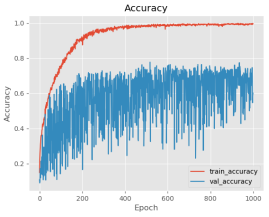
(f) Treinamento 6 com 1.000 épocas



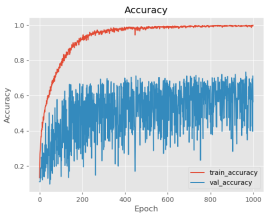
(g) Treinamento 7 com 1.000 épocas



(h) Treinamento 8 com 1.000 épocas



(i) Treinamento 9 com 1.000 épocas

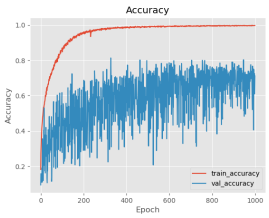


(j) Treinamento 10 com 1.000 épocas

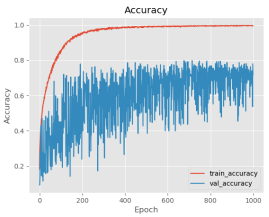
Figura 47 – Todos os treinamentos da Rede aparesentaram a mesma tendência da Acurácia com 30% das imagens.

Fonte: Elaborado pelo autor, 2022.

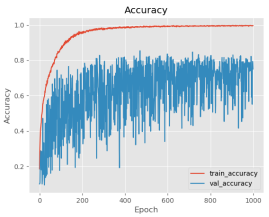
Segue a sequência de treinamento com 6.300 imagens:



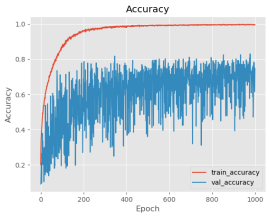
(a) Treinamento 1 com 1.000 épocas



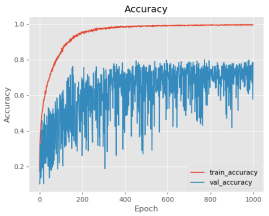
(b) Treinamento 2 com 1.000 épocas



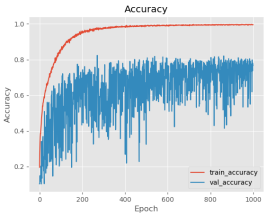
(c) Treinamento 3 com 1.000 épocas



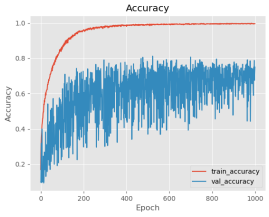
(d) Treinamento 4 com 1.000 épocas



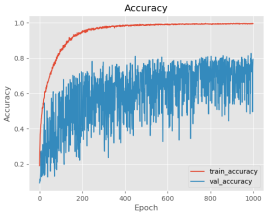
(e) Treinamento 5 com 1.000 épocas



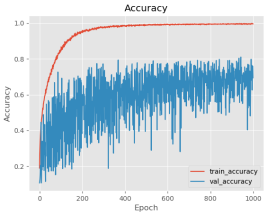
(f) Treinamento 6 com 1.000 épocas



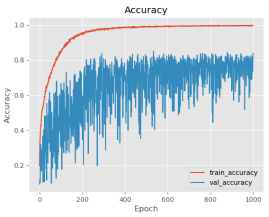
(g) Treinamento 7 com 1.000 épocas



(h) Treinamento 8 com 1.000 épocas



(i) Treinamento 9 com 1.000 épocas

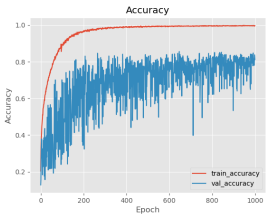


(j) Treinamento 10 com 1.000 épocas

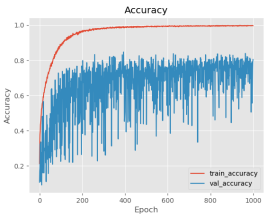
Figura 48 – Todos os treinamentos da Rede apresentaram a mesma tendência de Acurácia com 70% das imagens.

Fonte: Elaborado pelo autor, 2022.

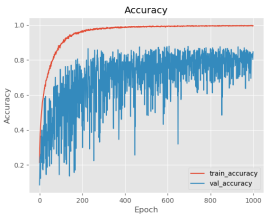
Segue a sequência de treinamento com 8.100 imagens:



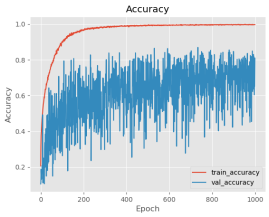
(a) Treinamento 1 com 1.000 épocas



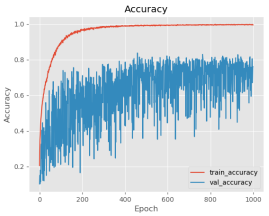
(b) Treinamento 2 com 1.000 épocas



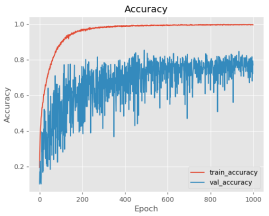
(c) Treinamento 3 com 1.000 épocas



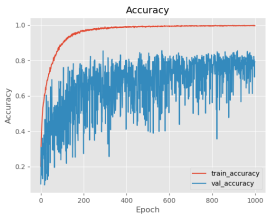
(d) Treinamento 4 com 1.000 épocas



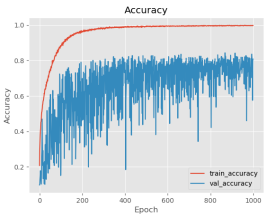
(e) Treinamento 5 com 1.000 épocas



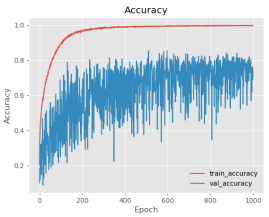
(f) Treinamento 6 com 1.000 épocas



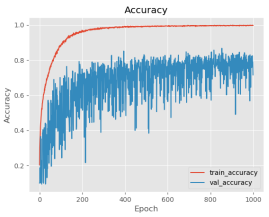
(g) Treinamento 7 com 1.000 épocas



(h) Treinamento 8 com 1.000 épocas



(i) Treinamento 9 com 1.000 épocas

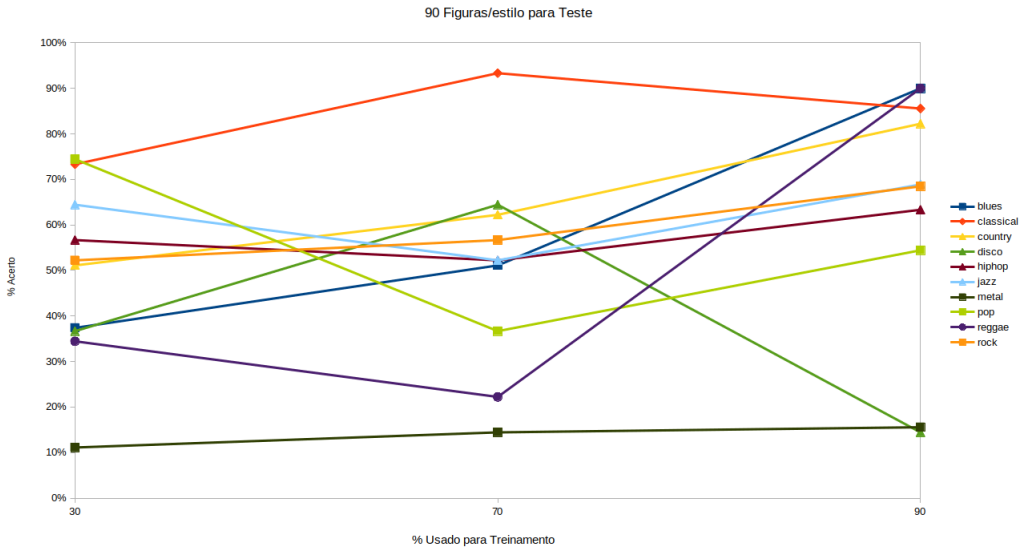


(j) Treinamento 10 com 1.000 épocas

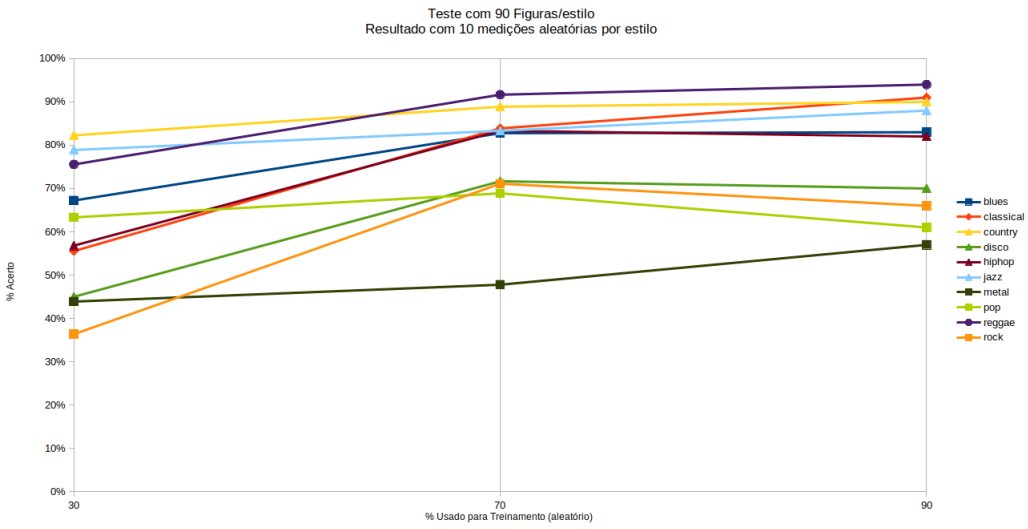
Figura 49 – Todos os treinamentos da Rede apresentaram a mesma tendência de Acurácia com 90% da imagens.

Fonte: Elaborado pelo autor, 2022.

Figura 50 apresenta o resutlado com: (a) a falta de aleatoriedade na escolha das figuras tanto para treinamento quanto para a previsão, fez com que os percentuais de acerto ficassem não uniformes. (b) a aleatoriedade na escolha das imagens para treinamento e previsão, fez com que houvesse a uniformidade sem valores discrepantes.



(a)



(b)

Figura 50 – a) Resultado sem aleatoriedade. b) Resultado com aleatoriedade.

Fonte: Elaborado pelo autor, 2022.

Tabela 8 – Exemplo do percentual de acerto na previsão do estilo musical em 90 imagens pela Rede treinada com 90% da base de dados. O percentual final é a média aritmética dos acertos. Neste exemplo é possível identificar que houve casos com 100% de acerto, conforme as linhas 1, 2, 6, 85 e 87.

Classificação Corretas do estilo Blues	
Estilo Previsto	**Percentual Acerto**
Blues	1.0000
Blues	1.0000
Blues	0.8916
Blues	0.9997
Blues	1.0000
...	...
Blues	0.9450
Blues	1.0000
Blues	0.4229
Blues	1.0000
Blues	0.9996
Média Percentual Acerto Blues	**78.89%**

Fonte: Elaborado pelo autor, 2022.

O estilo musical Country conseguiu mais que 80% de acerto com apenas 30% das imagens da base de dados e, embora o estilo Reggae tenha tido o maior percentual de acerto com 70% de imagens, os estilos Blues, Classical, Country, Hip Hop ficaram com os percentuais parecidos. Os estilos Clássical e Country tiveram acerto acima dos 90%, conforme ilustrado na Figura 53.

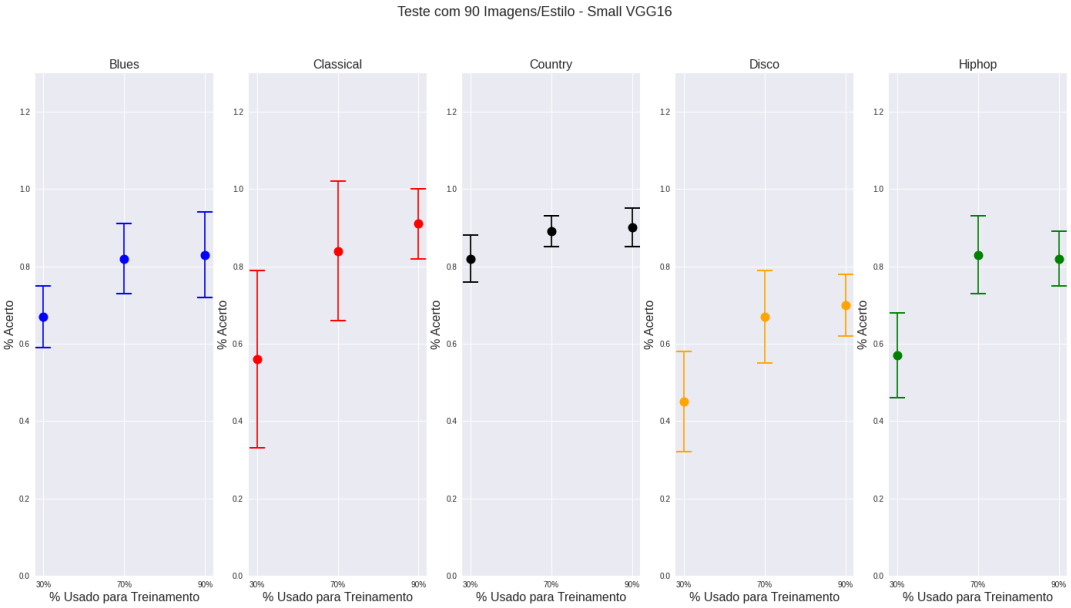


Figura 51 – Melhores resultados dos cinco primeiros estilos.

Fonte: Elaborado pelo autor, 2022.

Em segundo lugar ficou o estilo Jazz com 79% de acerto com apenas 30% da base de dados, Jazz ficou com o percentual de acerto próximo aos da figura Figura 53. O estilo Reggae teve acerto de 94% das previsões com 90% da base de dados, percentual próximo dos estilos Clássical e Country, conforme ilustrado na Figura 55.

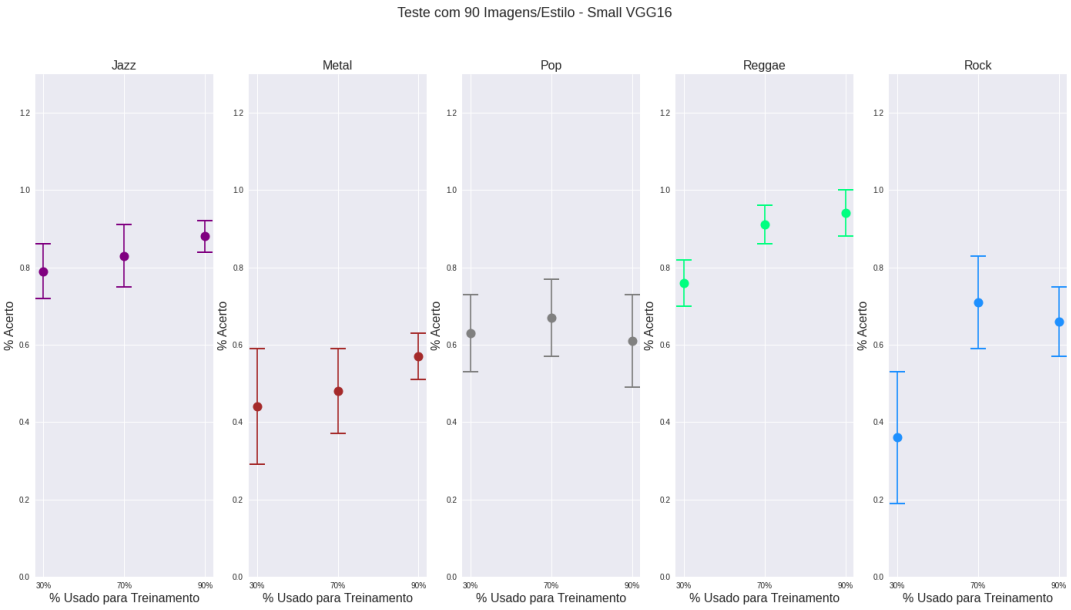


Figura 52 – Melhores resultados dos cinco últimos estilos.

Fonte: Elaborado pelo autor, 2022.

Tabela 9 – 30% das imagens geradas pelos Coeficientes Aproximados previram o estilo musical Country com maior valor médio de acerto de 82% e com desvio padrão de 0,06.

30%	**blues**	**classical**	**country**	**disco**	**hiphop**	**jazz**	**metal**	**pop**	**reggae**	**rock**
média	0,67	0,56	**0,82**	0,45	0,57	0,79	0,44	0,63	0,76	0,36
desvio padrão	0,08	0,23	**0,06**	0,13	0,11	0,07	0,15	0,10	0,06	0,16

Fonte: Elaborado pelo autor, 2022.

Tabela 10 – 70% das imagens geradas pelos Coeficiente Aproximados previram o estilo musical Reggae com maior valor médio de acerto de 91% e com desvio padrão de 0,05.

70%	**blues**	**classical**	**country**	**disco**	**hiphop**	**jazz**	**metal**	**pop**	**reggae**	**rock**
média	0,82	0,84	0,89	0,67	0,83	0,83	0,48	0,67	**0,91**	0,71
desvio padrão	0,09	0,18	0,04	0,12	0,10	0,08	0,11	0,10	**0,05**	0,12

Fonte: Elaborado pelo autor, 2022.

Tabela 11 – 90% das imagens geradas pelos Coeficiente Aproximados previram o estilo musical Reggae com maior valor médio de acerto de 94% e com desvio padrão de 0,06.

90%	**blues**	**classical**	**country**	**disco**	**hiphop**	**jazz**	**metal**	**pop**	**reggae**	**rock**
média	0,83	0,91	0,90	0,70	0,82	0,88	0,57	0,61	**0,94**	0,66
desvio padrão	0,11	0,09	0,05	0,08	0,07	0,04	0,06	0,12	**0,06**	0,09

Fonte: Elaborado pelo autor, 2022.

Segue abaixo o pseudocódigo algoritmo 4 da previsão de acerto da Rede Neural Convolucional..

Algoritmo 4 Previsão de Acerto da Rede

Entrada:
genres: Estilo musical
EstiloPrevisto: Vetor com a quantidade de valores do estilo
PercentualAcerto: Vetor com o percentual de acerto
Saída:
PercentualAcertoEstilo: Percentual de acerto por estilo musical

```
4  início
5      para estilo in genres faça
           /* Carregar os valores previstos de cada estilo                    */
6          para EstiloPrevisto in estilo faça
               /* Listar resultado de cada estilo                             */
7              PercentualAcerto_Estilo ← PercentualAcerto
               /* Repete 10 vezes para cada estilo */
8              arquivo ←PercentualAcerto_EstiloFinal;
9          fim
10     fim
11 fim
```

5.3 Comparação de Treinamento somente com cA e metade das Camadas de Mapeamento de Características

Nestes cinco primeiros estilos, a Rede "SmallVGG" obteve melhor previsão em todos os valores do que a Rede com menos camadas de mapeamento de características, exceto o estilo Clássical, conforme ilustrado na Figura 54.

Com apenas 30% das imagens, o percentual de acerto nas previsões do estilo Clássical alcançou 60%. Este resultado foi maior do que os 30% da Rede com maior número de camadas (Figura 54).

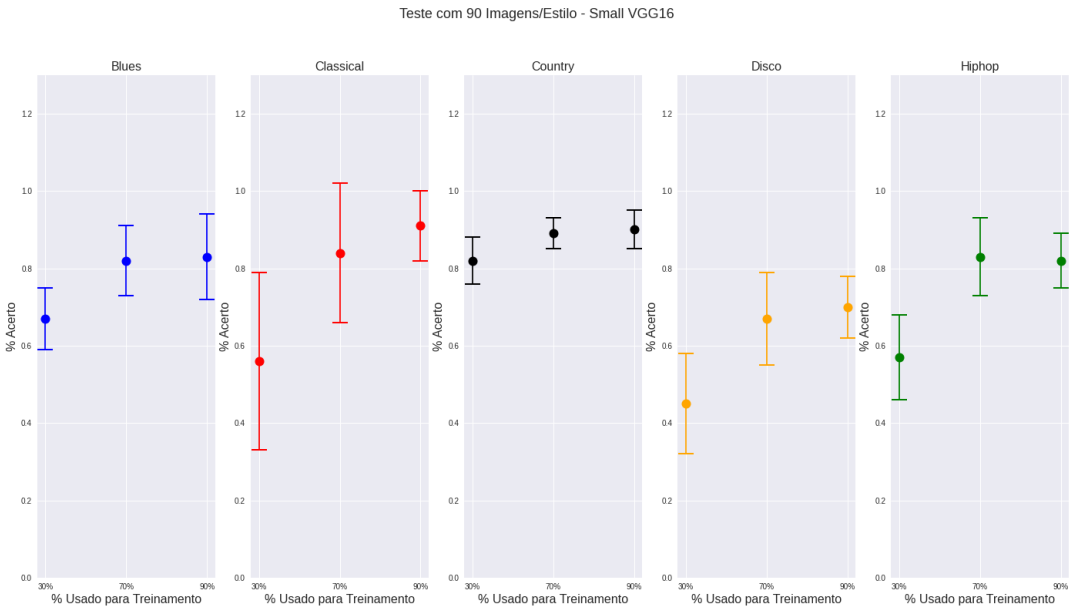


Figura 53 – Resultados da Rede "SmallVGG16" dos cinco primeiros estilos.

Fonte: Elaborado pelo autor, 2022.

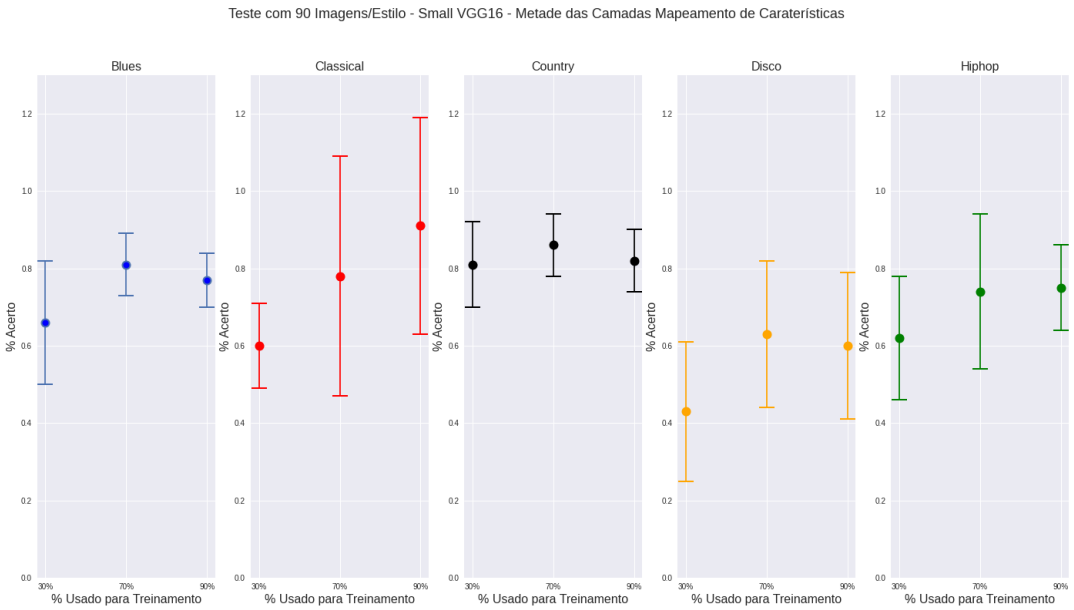


Figura 54 – Estilo Clássical acertou 60% das previsões com apenas 30% imagens.

Fonte: Elaborado pelo autor, 2022.

Nestes cinco últimos estilos, a Rede "SmallVGG" obteve os valores com previsão melhores do que as da Rede com menos Camadas de Mapeamento de Características (Figura 56). O estilo Metal apresentou uma curva quase linear, mas com alto desvio padrão.

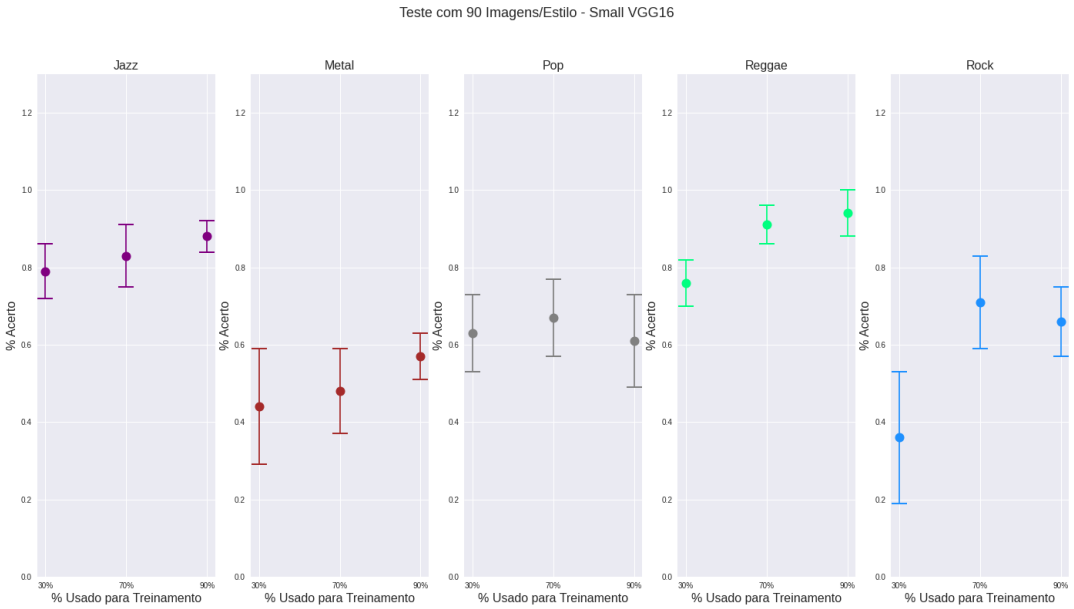


Figura 55 – Resultados da Rede "SmallVGG16" dos cinco últimos estilos

Fonte: Elaborado pelo autor, 2022.

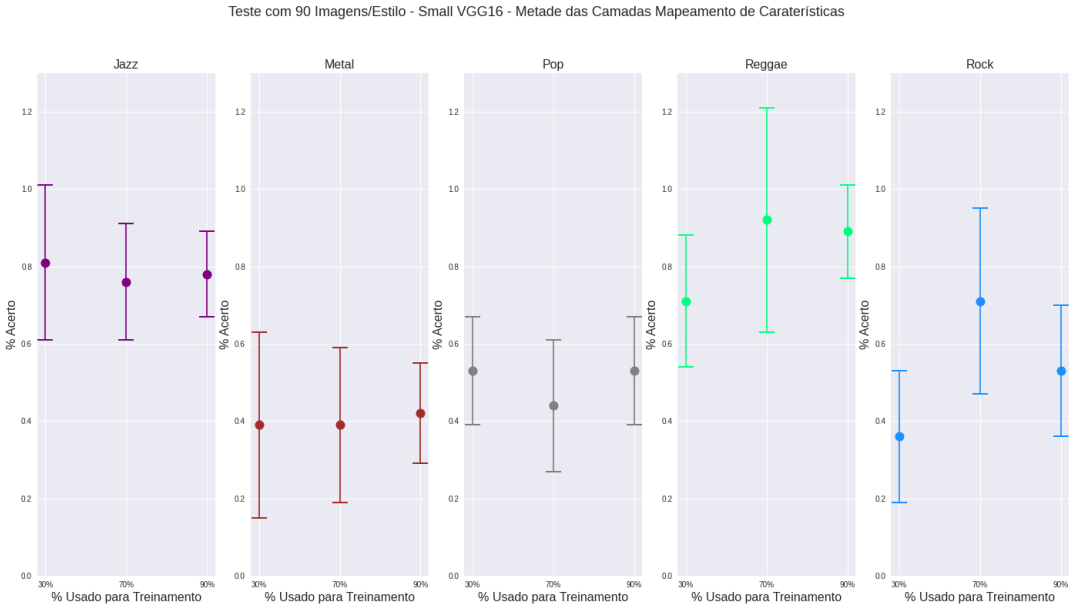


Figura 56 – Rede "SmallVGG obteve melhor valores que metade das Camadas de Mapeamento de Características dos cinco últimos estilos.

Fonte: Elaborado pelo autor, 2022.

5.4 Comparação entre a Rede Neural Convolucional com uma com Sinal Completo

Tabela 12 – Os mesmos parâmetros foram mantidos na comparação, a Rede treinada com todos os Coeficiente Contínuos da Transformada Wavelet teve o percentual 8 vezes menor do que a Rede treinada com o Coeficientes Aproximados da Transformada Wavelet Discreta.

Período [s]	**Freq. Amostragem [Hz]**	**Pixels**	**Canais**	**Coeficientes Transformada Wavelet**	**Quant. Cam. Convol.**	***Learning Rate***	**Quant. Imagens Treinam. (sem aleat.)**	**Épocas**	**Acurácia**	**Quant. Imagens Previsão**	**Acerto do Estilo %**
30	22500	128x128	3	Contínuo	6	0,01	1.000	700	0,129	20	5%
30	22500	64x64	3	Discreto (cA1)	6	0,01	1.000	600	0,50	20	**40%**

Fonte: Elaborado pelo autor, 2022.

5.5 Comparação entre a Rede Neural Convolucional com uma Rede Neural com Camadas Ocultas

Tabela 13 – Nesta comparação foi utilizado o mínimo da Rede Neural Convolucional com 64x64 Pixels, a rede neural que foi comparada possui 24x24 Pixels utilizando os 30 segundos de cada arquivo de áudio e, mesmo assim, conseguiu apenas 10% de acerto nas previsões.

Período [s]	**Frequência Amostragem [Hz]**	**Pixels**	**Canais**	**Tipo de Rede Neural**	**Quant. Cam. Convol.**	***Learning Rate***	**Quant. Imagens Treinam.**	**Épocas**	**Acurácia**	**Quant. Imagens Previsão**	**Acerto do Estilo %**
30	22500	28x28	3	Rede Neural Camada Oculta	6	0,01	1.000	15	0,10	5	10%
30	22500	64x64	3	Rede Neural Convolucional	6	0,01	1.000	600	0,50	20	**40%**

Fonte: Elaborado pelo autor, 2022.

Na seção 5.1 foram apresentados os primeiros resultados que fizeram parte do desenvolvimento da análise.

Esses resultados iniciais foram alcançados pela utilização de todo o sinal de áudio, com período igual a 30 segundos, porém este método se demonstrou ineficiente pelos resultados. Inferimos os dois coeficientes, Aproximados e Detalhados.

Isto gerou espectrogramas muito parecidos em todos os estilos, o que também dificultou a definição da quantidade de pontos amostrados do sinal para criar essas imagens.

Embora os resultados obtidos inicialmente foram diferentes do esperado vale ressaltar a importância desses valores naquela fase do trabalho, pois isto ajudou na mudança de estratégia para alcançar os principais resultados deste trabalho.

A Tabela 7 apresentou melhores resultados utilizando os Coeficientes Aproximados do sinal, que fez alterar a abordagem, porque observando a coluna "Conjunto de Ações" 4 e 5, houve mais de 60% de acerto na previsão se comparada com as ações 1 a 3. Esses melhores valores nesses conjuntos de ações ocorreram devido a extração somente dos Coeficientes Aproximados de cada série temporal.

Na coluna "Período", o valor "3x 10" significa que foi retirada a porção do sinal com os Coeficientes Aproximados, essa quantidade foi dividida em 3 partes iguais e, de cada parte, foram retirados os pontos amostrados. A melhora da acurácia e da previsão foi devido a esta segmentação que permitiu à Rede aumentar os acertos na previsão dos estilos.

As imagens geradas através dos Coeficientes Aproximados da Transformada Wavelet Discreta, se mostraram eficientes para a obtenção desses melhores resultados. Entretanto, esses valores não apresentou uma uniformidade na tendência de acertos das previsões dos estilos. A Figura 46 ilustrou bem essa disparidade.

A separação ao acaso das 10 últimas imagens para serem previstas e o restante para serem treinadas, fizeram com que os resultados do gráfico mostrados na Figura 46 apresentassem o comportamento sem padrão.

Outro ponto importante, a base de dados utilizada para serem previstas pela Rede treinada, não foi a mesma utilizada por Tzanetakis (2002). Os áudios foram escolhidos na Internet[1], porém com os 10 estilos de treinamento. Esta base de dados interferiu nos resultados da previsão.

[1] https://archive.org/

Na seção seção 5.2 foi apresentado os principais resultados com o efeito da aleatoriedade, tanto na escolha das imagens para treinamento e validação quanto às imagens para serem previstas. Os resultados mostraram uma tendência maior de acerto na previsão conforme aumentado o número de imagens para treinamento. Este método foi ilustrado na Figura 44 do Capítulo 4.

A homogeneidade dessa tendência pode ser observada na comparação da Figura 46 (sem aleatoriedade)com a Figura 50 (com aleatoriedade).

Após esses ajustes, o comportamento da curva de validação do treinamento e da acurácia da Rede ficaram estáveis em torno de um dado valor, isto pode ser verificado nos resultados da validação de treinamento e acurácia de 1.000 épocas executadas 10 vezes em cada grupo contendo 30% (Figura 47), 70% (Figura 48) e 90% (Figura 49) da base de dados de imagens.

Na Tabela 8 foi apresentado um exemplo com o resultado de percentual de acerto de previsão feita pela Rede treinada com 8.100 imagens para o estilo musical Blues.

Os gráficos ilustrados na Figura 53 e Figura 55 apresentaram os melhores resultados para cada grupo de imagens e, os valores específicos de cada medição estão na Tabela 9, Tabela 10 e Tabela 11.

Na seção 5.3 foi comparada a vantagem de utilizar os Coeficientes Aproximados ao contrário do uso de todo o sinal para treinamento da Rede. Houve a comparação desses resultados principais com uma Rede treinada somente com a metade das camadas de mapeamento de características (Figura 42) e com o mesmo conjunto de dados e foram apresentados na seção 5.4.

E, por fim, na seção 5.5 comparamos a Rede treinada pelo nosso método com uma Rede Neural, somente com camadas ocultas, treinada com todo o sinal.

6 Conclusão

Neste trabalho, explorou-se o problema de classificação de sinais de áudio em seus estilos musicais com um método eficiente de classificação utilizando os Coeficientes Aproximados da Transformada Wavelet Discreta.

Com apenas metade do sinal de áudio, ou seja, 15 segundos, foram geradas mais imagens para o treinamento da Rede Neural Convolucional. Isto foi possível devido às novas séries temporais geradas pelos coeficientes.

Esta Rede foi comparada com outra rede utilizando todo o sinal de 30 segundos, os resultados mostraram que a Rede treinada com 15 segundos do sinal foi mais eficiente no acerto das previsões dos estilos musicais.

Comparamos a Rede treinada com 15 segundos com outra rede, com os mesmos 15 segundos, porém com menor número de camadas de mapeamento de características. A Rede com o maior número de camadas também apresentou melhores percentuais de acertos nas previsões dos estilos.

E por fim, esta mesma Rede treinada com 15 segundos foi comparada com a uma Rede Neural de Camadas Ocultas e também apresentou maiores percentuais de acertos nas previsões dos estilos.

6.1 Considerações Finais e Trabalhos Futuros

Método deste trabalho teve como premissa a utilização de uma base de dados já validada por diversos trabalhos acadêmicos, deste modo, a qualidade dos dados já foi validada. Embora essa base tenha poucas regravações, ou seja, mais de uma versão para a mesma música, isto não gera demérito para a utilização desses arquivos.

Os resultados foram promissores pelo fato dessa Rede Neural Convolucional ser menor que a VGG16, isto utilizou menos recursos computacionais, porém vale ressaltar que é inevitável a utilização de uma placa de vídeo com GPU para as simulações.

O tempo que este modelo precisou para ser criado, passou um fluxo de quatro estágios: um que **captura dos coeficientes aproximados** do sinal de áudio para obter as séries temporais; depois a **geração das imagens** utilizando a Transformada Wavelet;

o **treinamento da Rede**; e por fim, permitir que a Rede treinada faça a **previsão de novas imagens** para classificá-las nos estilos musicais corretos.

A análise dos dados antes do treinamento tem que ser através de escolha com aleatoriedade, tanto para treinamento ou para as previsões.

O processo da criação das espectrogramas pode ser melhorado utilizando a mesma linguagem de programação, ao invés das duas utilizadas (Python e Matlab).

Conforme os resultados, as imagens dos espectrogramas geradas pelos coeficientes da Transformada Wavelet, demonstram eficiência para treinar uma Rede Neural Convolucional e com este método pode ser um outro meio para obter bons resultados, assim como a Transformada de Fourier tem sido utilizada.

A sugestão para trabalhos futuros seriam duas, a primeira seria para treinar a mesma Rede com outros tipos de Wavelets e com a mesma quantidade de Coeficientes Aproximados para encontrar as melhores Wavelets. A segunda sugestão seria descobrir qual seria a melhor quantidade de níveis de Coeficientes Aproximados para cada tipo de Wavelet.

Referências

ADDISON, N. *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine, and finance*. 1st. ed. [S.l.]: Taylor Francis, 2002. ISBN 0750306920; 9780750306928.

BRUNEL, N.; HAKIM, V.; RICHARDSON, M. J. Single neuron dynamics and computation. *Current Opinion in Neurobiology*, v. 25, p. 149–155, 2014. ISSN 0959-4388. Theoretical and computational neuroscience. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S0959438814000130⟩.

BRUNTON, S. L.; KUTZ, J. N. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. [S.l.]: Cambridge University Press, 2019. ISBN 9781108422093; 1108422098.

COSTA, Y. M.; OLIVEIRA, L. S.; SILLA, C. N. An evaluation of convolutional neural networks for music classification using spectrograms. *Applied Soft Computing*, v. 52, p. 28 – 38, 2017. ISSN 1568-4946.

DODGE, S.; KARAM, L. Understanding how image quality affects deep neural networks. *Eighth International Conference on Quality of Multimedia Experience (QoMEX),pp. 1-6*, 2016.

DOMINGUES, M.; MENDES, O.; KAIBARA, M.; MENCONI, V.; BERNARDES, E. Explorando a transformada wavelet contínua. *Revista Brasileira de Ensino de Física*, Sociedade Brasileira de Física, v. 38, n. Rev. Bras. Ensino Fís., 2016 38(3), 2016. ISSN 1806-1117. Disponível em: ⟨https://doi.org/10.1590/1806-9126-RBEF-2016-0019⟩.

DOWNEY, A. B. *Think DSP: Digital Signal Processing in Python*. [S.l.]: O'Reilly Media, 2016. ISBN 9781491938454; 1491938455.

FIGUEIREDO, D. G. *Análise de Fourier e equações diferenciais parciais*. [S.l.]: Edgard Blücher, 1977.

GEARMIN, F. The wavelet transform applications in music information retrieval. In: *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval*. New York, NY, USA: Association for Computing Machinery, 2009. (MIR '03), p. 102–108. ISBN 1581137788. Disponível em: ⟨https://doi.org/10.1145/973264.973281⟩.

GHILDIYAL, A.; SINGH, K.; SHARMA, S. Music genre classification using machine learning. *4th International Conference on Electronics, Communication and Aerospace Technology (ICECA). IEEE,pp. 1368–1372*, 2020.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. Deep learning. MIT Press, 2016.

GRIMALDI, M.; CUNNINGHAM, P.; KOKARAM, A. A wavelet packet representation of audio signals for music genre classification using different ensemble and feature selection techniques. Association for Computing Machinery, New York, NY, USA, p. 102–108, 2003.

GROSSMAN, A.; MORLET, J. Decomposition of hardy functions into squared integrable wavelets of constant shape. In: . [S.l.: s.n.], 1984. v. 15, p. 777–781.

HAAR, A. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen, 69(3):331–371*, 1910.

HAYKIN, S. Neural networks – a comprehensive foundation. Prentice Hall,, 1999.

HAYKIN, S.; VEEN, B. V. *Signals and Systems, 2nd Edition*. [S.l.]: Wiley, 2002. ISBN 0471164747.

HECK, B. S.; KAMEN, E. W. *Fundamentals of Signals and Systems Using the Web and MATLAB: Pearson New International Edition*. 3rd ed. ed. [S.l.]: Pearson Education Limited, 2014. ISBN 9781292025988; 1292025980.

JOAQUIM, M. B.; SARTORI, J. *Análise de Fourier*. [S.l.]: Dep. Engenharia Elétrica de São Carlos, 2003.

KHADEMI, G.; SIMON, D. Convolutional neural networks for environmentally aware locomotion mode recognition of lower-limb amputees. In: AMERICAN SOCIETY OF MECHANICAL ENGINEERS. *Dynamic Systems and Control Conference*. [S.l.], 2019. v. 59148, p. V001T07A005.

KUHN, M.; JOHNSON, K. *Applied Predictive Modeling*. [S.l.]: Springer, 2018. ISBN 1461468485.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, v. 1, n. 4, p. 541–551, 1989.

LERCH, A. *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. [S.l.]: Wiley-IEEE Press, 2012.

LI, D.; BISSYANDE, T. F. D. A.; KLEIN, J.; TRAON, Y. L. Time series classification with discrete wavelet transformed data: Insights from an empirical study. In: *The 28th international conference on software engineering and knowledge engineering (SEKE 2016)*. [S.l.: s.n.], 2016.

LI, T.; OGIHARA, M.; LI, Q. A comparative study on content-based music genre classification. In: . New York, NY, USA: Association for Computing Machinery, 2003. ISBN 1581136463. Disponível em: ⟨https://doi.org/10.1145/860435.860487⟩.

LILLY, J.; OLHEDE, S. Generalized morse wavelets as a superfamily of analytic wavelets. *IEEE Transactions on Signal Processing*, v. 60, p. pp. 2661–2670, 03 2012.

MALLAT, S. *A Wavelet Tour of Signal Processing: The Sparse Way*. [S.l.]: Academic Press, 2008. ISBN 9780123743701.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MULLER, M. *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. [S.l.]: Springer, 2015. ISBN 3319219448.

OPPENHEIM, A. V.; WILLSKY, A. S.; HAMID, S. *Sinais e Sistemas*. 2. ed. [S.l.]: Pearson, 2010. ISBN 857605504X; 9788576055044.

RAO, K. D. *Signals and Systems*. [S.l.]: Birkhäuser, 2018. ISBN 9783319686745; 3319686747; 9783319686752; 3319686755.

RUSSELL, S. *Artificial Intelligence: A Modern Approach*. [S.l.]: Pearson Education India, 2015. ISBN 9789332543515.

SCALVENZI, R. F.; GUIDO, R. C.; MARRANGHELLO, N. Wavelet-packets associated with support vector machine are efective for monophone sorting in music signals. International Journal of Semantic Computing, Vol. 13, No. 3 (2019) 415–425, 2019.

SHARMA, G.; KARTIKEYAN, U.; KRISHNAN, S. Trends in audio signal feature extraction methods. Elsevier, 2019.

SIETSMA, J.; DOW, R. J. F. Creating artificial neural networks that generalize. *Neural Networks*, v. 4, p. 67–79, 1991.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

SMITH, S. W. *The Scientist and Engineer's Guide to Digital Signal Processing*. 2. ed. [S.l.]: California technical Publishing, 1999. ISBN 0966017641; 9780966017649; 0966017668; 9780966017663; 0966017676; 9780966017670.

SPANIAS, A.; PAINTER, T.; ATTI, V. *Audio Signal Processing and Coding*. [S.l.]: Wiley-Interscience, 2007. ISBN 9780471791478; 0471791474.

STURM, B. L. The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use. *CoRR*, abs/1306.1461, 2013. Disponível em: ⟨http://arxiv.org/abs/1306.1461⟩.

TZANETAKIS, G.; COOK, P. Musical genre classification of audio signals. *IEEE Transactions on Audio and Speech Processing*, 2002.

WEIHS, C.; JANNACH, D.; VATOLKINA, I.; RUDOLPH, G. *Music Data Analysis: Foundations and Applications*. [S.l.]: Chapman and Hall/CRC, 2019.

WITTEN, I. H.; FRANK, E.; HALL, M. A.; PAL, C. J. *Data Mining: Practical Machine Learning Tools and Techniques*. 4. ed. [S.l.]: Morgan Kaufmann, 2016. (Morgan Kaufmann Series in Data Management Systems). ISBN 0128042915; 9780128042915.

ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

Apêndice A – Código 1 Python - Extração de Coeficientes

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun May 23 12:48:37 2021

@author: milton
"""
"""
Gerar ts em arquivo txt

dos Coeficientes Aproximados
"""

%matplotlib inline
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import wave
import scipy.io.wavfile
import argparse
import librosa
from pydub.playback import play
from pydub import AudioSegment
from scipy.fftpack import dct
from scipy.io.wavfile import read
import pathlib
import csv
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from PIL import Image
import warnings
warnings.filterwarnings('ignore')

#Wavelet
import pywt
from scipy import signal
import scaleogram as scg

# wavedec function
from pywt import wavedec
```

```
# ==========================================================================
# Criação de TXT dos cAs de 1a, 2a e 3a ordem
# ==========================================================================




# ==========================================================================
# TXT_FS_WAVSong_cA1
# ==========================================================================
# Fs = 22050
# ==========================================================================
# wt = db1
# ==========================================================================
# cA - 1st order
# ==========================================================================

Fs = 22050
plt.figure(figsize=(10,10))
genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
for g in genres:
    pathlib.Path(f'6.TXT_FS_TestWAVSong_cA1/{g}').mkdir(parents=True, exist_ok=True)
    for filename in os.listdir(f'./MIR/genres3_2songs_wav/{g}'):
        songname = f'./MIR/genres3_2songs_wav/{g}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=30, sr=Fs)
        x=y
        coeffs = wavedec(x, 'db1', level=1)
        cA, cD = coeffs
        np.savetxt(f'6.TXT_FS_TestWAVSong_cA1/{g}/{filename[:-3].replace(".","")}.txt',
                   cA, fmt="%4f", newline='\n')

# ==========================================================================
# TXT_FS_WAVSong_cA2
# ==========================================================================
# Fs = 22050
# ==========================================================================
# wt = db1
# ==========================================================================
# cA2 - 2nd order
# ==========================================================================

Fs = 22050
plt.figure(figsize=(10,10))
genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
for g in genres:
    pathlib.Path(f'7.TXT_FS_TestWAVSong_cA2/{g}').mkdir(parents=True, exist_ok=True)
    for filename in os.listdir(f'./MIR/genres3_2songs_wav/{g}'):
```

```
        songname = f'./MIR/genres3_2songs_wav/{g}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=30, sr=Fs)
        x=y
        # cA - 1st order
        coeffs = wavedec(x, 'db1', level=1)
        # cA - 2nd order
        coeffs2 = wavedec(x, 'db1', level=2)
        cA2, cD2, cD1 = coeffs2
        np.savetxt(f'7.TXT_FS_TestWAVSong_cA2/{g}/{filename[:-3].replace(".","")}.txt',
                   cA2, fmt="%4f", newline='\n')

# ==========================================================================
# TXT_FS_WAVSong_cA3
# ==========================================================================
# Fs = 22050
# ==========================================================================
# wt = db1
# ==========================================================================
# cA3 - 3rd order
# ==========================================================================

Fs = 22050
plt.figure(figsize=(10,10))
genres = 'blues classical country disco hiphop jazz metal pop reggae rock'.split()
for g in genres:
    pathlib.Path(f'8.TXT_FS_TestWAVSong_cA3/{g}').mkdir(parents=True, exist_ok=True)
    for filename in os.listdir(f'./MIR/genres3_2songs_wav/{g}'):
        songname = f'./MIR/genres3_2songs_wav/{g}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=30, sr=Fs)
        x=y
        # # cA - 1st order
        # coeffs = wavedec(x, 'db1', level=1)
        # # cA - 2nd order
        # coeffs2 = wavedec(x, 'db1', level=2)
        # cA2, cD2, cD1 = coeffs2
        # cA - 3rd order
        coeffs3 = wavedec(x, 'db1', level=3)
        cA3, cD3, cD2, cD1 = coeffs3
        np.savetxt(f'8.TXT_FS_TestWAVSong_cA3/{g}/{filename[:-3].replace(".","")}.txt',
                   cA3, fmt="%4f", newline='\n')


# ==========================================================================
```

Apêndice B – Código 2 Matlab - Geração de Espectrogramas

```
% ========================================================================
% Exemplo de uma parte codigo que gerou os
% espectrogramas dos Coeficientes Aproximados
% com pontos amostrados de 1 a 82600
% ========================================================================

fid = fopen('/home/milton/Downloads/genres_1000samples/3.TXT_FS_WAVSong_cA1/blues
/blues00000.txt', 'rt');
output = textscan(fid, '%f');
fclose(fid);
(output{1});

Fs = 22500;
fb = cwtfilterbank('SignalLength',length(1:82600),...
     'SamplingFrequency',Fs,...
     'VoicesPerOctave',12);
sig = output{1}(1:82600);
[cfs,frq] = wt(fb,sig);
t = (1:82600)/Fs;figure;pcolor(t,frq,abs(cfs))
%set(gca,'yscale','log');shading interp;axis tight;
set(gca,'yscale','log','XTick',[], 'YTick', []);shading interp; %remover valores dos
eixos-x e y
%title('Scalogram');xlabel('Time (s)');ylabel('Frequency (Hz)')
saveas(gcf,'blues00000.jpg');
close gcf

% ========================================================================
```

Apêndice C – Código 3 Python - Treinamento de CNN SmallVGGNet

```
# Configurar o matplotlib em backend para as figuras sejam salvas em background

import matplotlib
matplotlib.use("Agg")

# Pacotes
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import SGD
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation
from keras.layers.core import Flatten
from keras.layers.core import Dropout
from keras.layers.core import Dense
from keras import backend as K
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping
from imutils import paths
import matplotlib.pyplot as plt
import numpy as np
import argparse
import random
import pickle
import cv2
import os

print("TensorFlow trabalhando on Backend...")


# Inicializar os dados e os labels
print("[INFO] carregando imagens...")
data = []
labels = []
# entrar com o caminho das imagens
image_data_folder_path = "path/cA_2700Figures_random_perStyle/"
# embaralhamento das imagens
imagePaths = sorted(list(paths.list_images(image_data_folder_path)))
```

```
random.shuffle(imagePaths)

#loop sobre as imagens de entrada
for imagePath in imagePaths:
        # carregar a imagens e redimensionar para 64x64 pixels (requerimento para a
        # SmallVGGNet)
        image = cv2.imread(imagePath)
        image = cv2.resize(image, (64, 64))
        data.append(image)



        # extrair os "labels" das classes
        # atualiza a lista de "labels"
        label = imagePath.split(os.path.sep)[-2]
        labels.append(label)


# normaliza os valores dos pixels no range [0,1]
data = np.array(data, dtype="float") / 255.0

labels = np.array(labels)
print(labels)
print(labels.shape)
# Divis o das imagens para Treinamento e Testes
# 75% para Treinamento e 25% para Teste
(trainX, testX, trainY, testY) = train_test_split(data,labels,
test_size=0.25, random_state=42)

lb = LabelBinarizer()
trainY = lb.fit_transform(trainY)
testY = lb.transform(testY)

# construir o gerador de imagens para aumento de dados
aug = ImageDataGenerator(rotation_range=30, width_shift_range=0.1,
                         height_shift_range=0.1, shear_range=0.2, zoom_range=0.2,
                         horizontal_flip=True, fill_mode="nearest")


# dimens o das imagens
height = 64
width = 64
depth = 3

inputShape = (height, width, depth)
```

```
classes = len(lb.classes_)
model = Sequential()
# CONV => RELU => POOL layer set
model.add(Conv2D(32, (3, 3), padding="same",
                 input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
#model.add(Dropout(0.25))

# (CONV => RELU) * 2 => POOL layer set
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())


model.add(MaxPooling2D(pool_size=(2, 2)))
#model.add(Dropout(0.25))

# (CONV => RELU) * 3 => POOL layer set
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
#model.add(BatchNormalization())


model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
#model.add(Dropout(0.25))

# first (and only) set of FC => RELU layers
model.add(Flatten())
model.add(Dense(512)) #512
model.add(Activation("relu"))
model.add(BatchNormalization())
#model.add(Dropout(0.5))

# softmax classifier
model.add(Dense(classes))
model.add(Activation("softmax"))
```

```
model.summary()

# learning rate, n. de  pocas  para treinamento e batch size
INIT_LR = 0.01
EPOCHS = 1000
BS = 32

# iniciar modelo e otimzador
# ATENCAO: esta Rede ser  treinada por um base de dados multi-classes,
# por isso n o pode utilizar a perda "binary_crossentropy", porque
# isto    somente para 2-classes de classifica  o)=
print("[INFO] treinando a Rede...")
opt = SGD(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="categorical_crossentropy", optimizer=opt,
              metrics=["accuracy"])



# train the network
H = model.fit_generator(aug.flow(trainX, trainY, batch_size=BS),
                        validation_data=(testX, testY),
                        steps_per_epoch=len(trainX) // BS,
                        epochs=EPOCHS)


# avaliar a Rede
print("[INFO] avaliando a rede...")
predictions = model.predict(testX, batch_size=BS)
print(classification_report(testY.argmax(axis=1),
                            predictions.argmax(axis=1), target_names=lb.classes_))

# plotar a acur cia e perda do treinamento
N = np.arange(0, EPOCHS)
plt.style.use("ggplot")
plt.figure()
#plt.plot(N, H.history["loss"], label="train_loss")
#plt.plot(N, H.history["val_loss"], label="val_loss")
plt.plot(N, H.history["accuracy"], label="train_accuracy")
plt.plot(N, H.history["val_accuracy"], label="val_accuracy")
plt.title("Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.savefig("modelo30pc_cA1_1.png")

# salvar o modelo e "binarizar" os "labels" para o disco
print("[INFO] serializando os binarios da rede e dos labels...")
```

```
model.save("modelo30pc_cA1_1.model")
f = open("modelo30pc_cA1_1.pickle", "wb")
f.write(pickle.dumps(lb))
f.close()
```

Apêndice D – Código 4 Python - Previsão da Rede

```
   # -*- coding: utf-8 -*-
"""errorBars_30PC_acerto_v1.ipynb

# **Análise dos resultados/estilo**
"""


"""# Carregar as bibliotecas"""

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

"""# 30 PC_1"""

# Blues

blues = pd.read_table('/path/1/resultadoBlues.txt', sep=':', header=None)
blues[1] = blues[1]/100
blues = blues.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretablues = blues.loc[blues['estiloPrevisto']=='blues']
print("Classificação certa Blues: \n\n", previsaoCorretablues)

porcentualAcerto = len(previsaoCorretablues)/len(blues)
print(f"PorcentualAcerto Blues = {porcentualAcerto:,.2%}")

# Classical

classical = pd.read_table('/path/1/resultadoClassical.txt', sep=':', header=None)
classical[1] = classical[1]/100
classical = classical.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretaClassical = classical.loc[classical['estiloPrevisto']=='classical']
print("Classificação certa Classical: \n\n", previsaoCorretaClassical)

porcentualAcerto = len(previsaoCorretaClassical)/len(classical)
print(f"PorcentualAcerto = {porcentualAcerto:,.2%}")

# Country

country = pd.read_table('/path/1/resultadoCountry.txt', sep=':', header=None)
```

```
country[1] = country[1]/100
country = country.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretacountry = country.loc[country['estiloPrevisto']=='country']
print("Classifica  o certa Country: \n\n", previsaoCorretacountry)

porcentualAcerto = len(previsaoCorretacountry)/len(country)
print(f"PorcentualAcerto Country = {porcentualAcerto:,.2%}")

# Disco

disco = pd.read_table('/path/1/resultadoDisco.txt', sep=':', header=None)
disco[1] = disco[1]/100
disco = disco.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretadisco = disco.loc[disco['estiloPrevisto']=='disco']
print("Classifica  o certa Disco: \n\n", previsaoCorretadisco)

porcentualAcerto = len(previsaoCorretadisco)/len(disco)
print(f"PorcentualAcerto Disco = {porcentualAcerto:,.2%}")

# Hiphop

hiphop = pd.read_table('/path/1/resultadoHiphop.txt', sep=':', header=None)
hiphop[1] = hiphop[1]/100
hiphop = hiphop.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretahiphop = hiphop.loc[hiphop['estiloPrevisto']=='hiphop']
print("Classifica  o certa Hiphop: \n\n", previsaoCorretahiphop)

porcentualAcerto = len(previsaoCorretahiphop)/len(hiphop)
print(f"PorcentualAcerto Hiphop = {porcentualAcerto:,.2%}")

# Jazz

jazz = pd.read_table('/path/1/resultadoJazz.txt', sep=':', header=None)
jazz[1] = jazz[1]/100
jazz = jazz.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretajazz = jazz.loc[jazz['estiloPrevisto']=='jazz']
print("Classifica  o certa Jazz: \n\n", previsaoCorretajazz)

porcentualAcerto = len(previsaoCorretajazz)/len(jazz)
print(f"PorcentualAcerto Jazz = {porcentualAcerto:,.2%}")

# Metal
```

```
metal = pd.read_table('/path/1/resultadoMetal.txt', sep=':', header=None)
metal[1] = metal[1]/100
metal = metal.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretametal = metal.loc[metal['estiloPrevisto']=='metal']
print("Classifica  o certa Metal: \n\n", previsaoCorretametal)

porcentualAcerto = len(previsaoCorretametal)/len(metal)
print(f"PorcentualAcerto Metal = {porcentualAcerto:,.2%}")

# Pop

pop = pd.read_table('/path/1/resultadoPop.txt', sep=':', header=None)
pop[1] = pop[1]/100
pop = pop.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretapop = pop.loc[pop['estiloPrevisto']=='pop']
print("Classifica  o certa Pop: \n\n", previsaoCorretapop)

porcentualAcerto = len(previsaoCorretapop)/len(pop)
print(f"PorcentualAcerto Pop = {porcentualAcerto:,.2%}")

# Reggae

reggae = pd.read_table('/path/1/resultadoReggae.txt', sep=':', header=None)
reggae[1] = reggae[1]
reggae = reggae.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretareggae = reggae.loc[reggae['estiloPrevisto']=='reggae']
print("Classifica  o certa Reggae: \n\n", previsaoCorretareggae)

porcentualAcerto = len(previsaoCorretareggae)/len(reggae)
print(f"PorcentualAcerto Reggae = {porcentualAcerto:,.2%}")

# Rock

rock = pd.read_table('/path/1/resultadoRock.txt', sep=':', header=None)
rock[1] = rock[1]
rock = rock.rename(columns={0:'estiloPrevisto', 1:'percentualAcerto'})

previsaoCorretarock = rock.loc[rock['estiloPrevisto']=='rock']
print("Classifica  o certa Rock: \n\n", previsaoCorretarock)

porcentualAcerto = len(previsaoCorretarock)/len(rock)
print(f"PorcentualAcerto Rock = {porcentualAcerto:,.2%}")

# Medir mais 9x
```