Universidade Federal da Bahia
Escola Politécnica
Colegiado do Curso de Engenharia de Computação

Pedro Augusto Dultra Neves Correia

Aplicação de Redes Neurais Convolucionais na Classificação de Efeitos de Guitarra presentes em faixas de áudio

Orientador: Prof. Dr. Antônio Carlos Lopes Fernandes Júnior

Salvador-Ba – Brasil
13 de dezembro de 2023

Pedro Augusto Dultra Neves Correia

Aplicação de Redes Neurais Convolucionais na Classificação de Efeitos de Guitarra presentes em faixas de áudio

Monografia apresentada ao Curso de Graduação em Engenharia de Computação da Universidade Federal da Bahia como parte dos requisitos para a obtenção do grau de Engenheiro da Computação.

Orientador: Prof. Dr. Antônio Carlos Lopes Fernandes Júnior

Salvador-Ba – Brasil

13 de dezembro de 2023

Pedro Augusto Dultra Neves Correia

Aplicação de Redes Neurais Convolucionais na Classificação de Efeitos de Guitarra presentes em faixas de áudio

Monografia apresentada ao Curso de Graduação em Engenharia de Computação da Universidade Federal da Bahia como parte dos requisitos para a obtenção do grau de Engenheiro da Computação.

Trabalho aprovado. Salvador-Ba – Brasil, 13 de dezembro de 2023:

Orientador(a): Prof. Dr. Antônio Carlos Lopes Fernandes Júnior

Prof. Dr. Eduardo Furtado de Simas Filho

Me. Luiz Alberto Guimarães Viana

Salvador-Ba – Brasil
13 de dezembro de 2023

Este trabalho é dedicado a Anna Christina, minha mãe, e Ronaldo Correia, meu pai. É graças a eles que eu consegui chegar até aqui.

Agradecimentos

Este trabalho representação a finalização de um ciclo onde eu tive o privilégio de contar com o auxílio de diversas pessoas que me impactaram e influenciaram de diferentes maneiras durante a minha jornada. Agradeço muito à minha mãe, que sempre esteve ao meu lado por toda a minha vida, me dando todo o amor e suporte necessário para que eu pudesse focar nos meus estudos, em conjunto com o meu padrasto, Pedro. Ao meu pai, que nos deixou há pouco mais de 2 anos, e cujo exemplo em vida me permitiu ver o poder transformador da educação na vida de uma pessoa. A todos os meus irmãos, família, amigos, amigas e colegas de trabalho que ajudaram a moldar quem eu sou hoje.

Agradeço também a André, um grande amigo que esteve ao meu lado ao longo de praticamente todas as disciplinas nos últimos 7 anos e que me ajudou imensamente a lidar com as limitações de *hardware* que seriam um impeditivo para esse trabalho. A Ariel, com quem também pude cursar grande parte das disciplinas do curso e ter o privilégio de trabalhar juntos na mesma empresa. Aos meus amigos Bruno, David, Elias, Guilherme, Kauan, Lorena, Marcel, Marcos, Noel e Rafael, que desde o primeiro semestre em 2017 deixaram de ser apenas colegas de curso, e mesmo trilhando diferentes caminhos na vida acadêmica, se tornaram amigos para uma vida toda; assim como os amigos de curso Ana Clara, Diego, Igor, Paloma, Laurinne, Romualdo, Miguel e Rafael Pondé.

Esse trabalho também é dedicado a minha companheira Andreia, com quem hoje tenho o privilégio de dividir a minha vida e que esteve nela durante quase toda a minha graduação, compartilhando amor, momentos de felicidade e apoio emocional. E por fim, a todos os professores e professoras que participaram do processo de aprendizado durante toda a minha vida, com um destaque especial ao meu orientador, Antônio Carlos, que foi o principal responsável por me fazer perceber as interseções entre a engenharia e a música, uma das principais paixões da minha vida e parte essencial da minha existência.

"Se enxerguei um pouco mais longe,
é porque me coloquei nos ombros de gigantes".
(Isaac Newton)

Resumo

Extrair informações a respeito da aplicação de um efeito de pedal de guitarra a um áudio é uma das tarefas do campo da análise de conteúdo musical. Este trabalho investiga a viabilidade de utilização de duas representações de tempo-frequência de um áudio, a *Hybrid Constant Q-Transform* (HCQT) e o escalograma *wavelet* para a construção de um classificador baseado em uma rede neural convolucional capaz de identificar qual efeito está sendo aplicado em faixas de áudio pré-gravadas. São geradas 4 diferentes representações para dois *datasets* contendo áudios de guitarra com efeitos de pedal sendo aplicados a eles, isolados ou em conjunto com outros instrumentos, para realizar o processo de treinamento da rede neural, dividindo os conjuntos de dados fornecidos para a mesma através do método *k-fold* para calcular a acurácia de cada modelo. Foi possível obter um bom desempenho para a classificação utilizando a HCQT, com acurácia simples de 95,9% em um dos conjuntos de dados, superando o erro humano teórico para especialistas nesse problema e com um desempenho similar ao obtido por outros trabalhos recentes dessa mesma tarefa.

Palavras-chave: MIR, redes neurais, aprendizado de máquina, redes neurais convolucionais, HCQT, *wavelet*, escalograma, espectrograma, MFCC, GFCC, pedais de guitarra

Abstract

Extracting information about a guitar effects pedal being applied to an audio is one of the tasks available for the field of musical content analysis. This study investigates the viability of the usage of two time-frequency representations of an audio, the HCQT and the wavelet scalogram to build a classifier based on a convolutional neural network that's able to identify what effect is being applied in a pre-recorded audio. Four different representations are generated for two datasets that contain electric guitar recordings with pedal effects being applied to them, isolated or alongside other instruments, so that the neural network can be trained with the provided datasets being split through the k-fold method to calculate the accuracy of each model. The classifier built using the HCQT managed to perform well and was able to reach accuracy levels of 95.9% in one of the datasets, a value greater than what is achieved by a human expert on this problem, as well as a performance comparable to other recent studies for the same task.

Keywords: MIR, neural networks, machine learning, convolutional neural networks, HCQT, wavelet, scalogram, spectrogram, MFCC, GFCC, guitar pedals

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CWT *Continuous Wavelet Transform*
Transformada Wavelet Contínua

CNN *Convolutional Neural Network*
Rede Neural Convolucional

CQT *Constant Q-Transform*
Transformada-Q Constante

DCT *Discrete Cosine Transform*
Transformada Discreta de Cosseno

DAW *Digital Audio Workstation*
Estação de Trabalho de Áudio Digital

FIR *Finite Impulse Response*
Resposta ao Impulso Finita

GEC-GIM *Guitar Effect Classification — Guitar Instrument Mix*

GFCC *Gammatone Frequency Cepstral Coefficients*
Coeficientes Cepstrais de Frequência Gammatone

HCQT *Hybrid Constant Q-Transform*
Transformada-Q Constante Híbrida

IDMT-SMT *Fraunhofer Institute for Digital Media Technology - Semantic Music Technologies*

JPEG *Joint Photographic Experts Group*

LFO Low Frequency Oscillator
Oscilador de Baixa Frequência

LTI *Linear Time Invariant*
Linear Invariante no Tempo

LTV *Linear Time Variant*
Linear Variante no Tempo

MLP *Multi Layer Perceptron*
Perceptron de Múltiplas Camadas

MFCC *Mel Frequency Cepstral Coefficients*
Coeficientes Cepstrais da Frequência Mel

NLTI *Non-Linear Time Invariant*
Não-Linear Invariante no Tempo

ReLU *Rectified Linear Unit*
Unidade Linear Retificada

RGB *Red, Green and Blue*
Padrão de cores verde, vermelho e azul

SVM *Support Vector Machine*
Máquina de vetores de suporte

STFT *Short Time Fourier Transform*
Transformada de Fourier de Tempo Curto

TFTD Transformada de Fourier de Tempo Discreto

Lista de símbolos

α	Taxa de aprendizado
$\Delta\xi$	Janela de frequência
Δt	Janela de tempo
λ	Frequência
$\forall$	Notação matemática "para todo"
A	Nota Musical Lá
A_n	Nota Musical Lá com oitava de índice n
$A\#$	Nota Musical Lá Sustenido
B	Nota Musical Si
C	Nota Musical Dó
$C\#$	Nota Musical Dó Sustenido
cos	Cosseno
D	Nota Musical Ré
$D\#$	Nota Musical Ré Sustenido
E	Nota Musical Mi
F	Nota Musical Fá
$F\#$	Nota Musical Fá Sustenido
G	Nota Musical Sol
$G\#$	Nota Musical Sol Sustenido
Hz	Hertz
j	$\sqrt{-1}$
log_n	Logaritmo de base n
mod	Operador módulo

sin	Seno
t	Tempo
$w[m]$	Janela em tempo discreto
$x(t)$	Sinal em tempo contínuo
$x[n]$	Sinal em tempo discreto
u.t.	Unidade de tempo

Sumário

1 Introdução

A utilização de efeitos de pedal em guitarras elétricas é uma prática que se popularizou a partir dos anos 50, com o surgimento do primeiro pedal de efeitos fabricado em larga escala (KARREN, 2020), e continua a ser uma prática recorrente até os dias de hoje (REVEILLAC, 2017) por possibilitar a modificação do timbre desses instrumentos. Desde o momento em que o uso desses efeitos em guitarras passou a ser visto como uma característica marcante do gênero musical rock e seus subgêneros (HERBST, 2021), diferentes variações e combinações de efeitos continuaram a ser exploradas por músicos com os mais diversos níveis de habilidade e em composições de estilos musicais distintos, com o propósito de obter um som com características únicas em suas peças de música.

1.0.1 Justificativa

A informação de qual efeito foi aplicado no som da guitarra de uma determinada música não é algo amplamente divulgado, assim como outros detalhes subjacentes à produção musical; e a tarefa de obter essa informação se torna não-trivial por requerer experiência nas áreas de composição e produção musical. Isso faz com que sistemas computacionais de Análise de Conteúdo de Áudio que consigam extrair essa característica de forma automática se tornem ferramentas úteis para músicos amadores que fazem versões *cover* de músicas já existentes ou para iniciantes no aprendizado de música que almejam alcançar uma sonoridade similar a de um determinado artista, reduzindo o tempo e o esforço que seriam necessários para identificar qual efeito foi utilizado através da tentativa e erro em experimentos com diferentes pedais.

É possível encontrar na literatura científica alguns exemplos de sistemas inteligentes com a capacidade de categorizar um pedal de guitarra de forma *online*, através do envio de um conjunto de sinais de áudio, tratando o pedal como um sistema de caixa preta para extrair *features* relevantes do sinal de saída para realizar essa classificação (EICHAS; FINK; ZÖLZER, 2015). Também é possível encontrar outras abordagens de classificação, como a construção de sistemas baseados em *Support Vector Machine* (SVM) que classificam os efeitos aplicados em diferentes gravações de guitarra isolada (STEIN, 2010), e outros que utilizam uma *Convolutional Neural Network* (CNN) para desempenhar a mesma tarefa de classificação, porém desta vez em gravações contendo combinações de múltiplos instrumentos, além de extrair parâmetros numéricos de configuração dos efeitos (HINRICHS et al., 2022).

1.0.2 Objetivos

Dentre as diferentes abordagens e tarefas trazidas acima, este trabalho tem como objetivo empregar CNNs para construir um modelo capaz de classificar efeitos de guitarra em uma gravação de áudio contendo a guitarra isolada ou em conjunto com múltiplos instrumentos.

O presente trabalho tem como objetivos específicos:

- aplicar conhecimentos associados ao processamento digital de sinais para descrever e analisar características pertinentes a diferentes tipos de efeitos de pedal de guitarra;
- comparar as diferentes representações de sinais de áudio no domínio da frequência que foram empregadas no processo de construção e treinamento da CNN de (HINRICHS et al., 2022);
- propor a utilização de duas novas representações no domínio da frequência para realizar o treinamento da CNN, a HCQT e os escalogramas *wavelet*;
- realizar o treinamento da rede convolucional com as duas novas representações dos sinais de áudio em um *dataset* contendo gravações de guitarra isolada ou em conjunto com múltiplos instrumentos;
- comparar o desempenho do modelo obtido através das novas representações com os resultados dos modelos originais de referência.

1.0.3 Estrutura do trabalho

A divisão dos capítulos do trabalho foi feita da seguinte maneira: no Capítulo 2, serão apresentadas as bases teóricas utilizadas no trabalho, detalhando os conceitos por trás das representações de frequência, efeitos de guitarra e redes neurais convolucionais utilizados ao longo do trabalho. No Capítulo 3, é descrita a metodologia de construção e treinamento da rede neural, assim como o conjunto de dados que foi utilizado para treinamento e avaliação do modelo. Os resultados obtidos com o modelo final e discussões a respeito desses resultados são trazidos no Capítulo 4, e em seguida, o Capítulo 5 trará as conclusões finais desse trabalho.

2 Bases Teóricas

Neste capítulo serão apresentadas as bases teóricas necessárias para a compreensão dos tópicos abordados nos próximos capítulos desse trabalho e que fundamentaram as análises realizadas durante os experimentos e iterações do processo de construção do modelo capaz de classificar os efeitos de guitarra presentes nas amostras de áudio apresentadas a ele.

2.1 Representações tempo-frequência

Um sinal de áudio pode ser definido como uma função contínua $x(t)$ da variação de pressão sonora com o tempo (LERCH, 2012). O sistema auditivo humano consegue perceber apenas as oscilações de pressão que possuem uma frequência dentro da faixa de 20 Hz e 20 kHz. Isso significa que é possível obter uma representação discreta no tempo $x[n]$ de um sinal de áudio contínuo no tempo $x(t)$ contendo toda a informação que é captada como som pelos ouvidos humanos, desde que amostras deste sinal tenham sido coletadas com uma taxa superior ao dobro da maior frequência que seres humanos conseguem ouvir. Esse critério é apresentado no Teorema de *Nyquist-Shannon* (OPPENHEIM; SCHAFER, 2013) e é indispensável para a construção de qualquer tipo de sistema de telecomunicação por voz ou gravação de áudio que realiza uma transdução da pressão sonora em valores de tensão elétrica. Um diagrama de blocos para um sistema ideal que representa a etapa de amostragem do sinal em tempo contínuo para o tempo discreto pode ser vista na Figura 1.

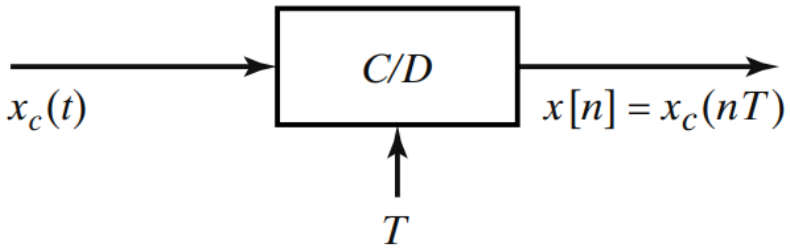


Fig. 1 – Diagrama de blocos de um sistema que amostra um sinal do tempo contínuo $x_c(t)$ como $x[n]$.

Fonte: (OPPENHEIM; SCHAFER, 2013)

Essas representações digitais de um sinal de áudio, costumeiramente obtidas com uma taxa de amostragem de 44,1 kHz, podem ser representadas no domínio do tempo, *i.e.*, uma medida discreta da sua intensidade ao longo de cada uma das amostras de tempo,

ou no domínio da frequência, onde usualmente calcula-se a potência de cada um dos diferentes valores de frequência presentes num sinal de áudio ao longo de janelas de tempo. As representações de áudio no domínio da frequência se tornam especialmente úteis para análises associadas ao tom e o entendimento do que o ouvido humano compreenderá como sons graves, médios e agudos.

2.1.1 Espectrograma

O espectrograma é uma representação obtida através do cálculo da *Short Time Fourier Transform* (STFT) de um sinal de áudio. Essa transformada pode ser descrita como a aplicação da Transformada de Fourier de Tempo Discreto (TFTD) a um sinal de tempo discreto multiplicada por uma janela de tempo de igual largura; essa janela será deslocada no tempo ao longo de todo o sinal, tornando possível a obtenção dos valores de frequência presentes em cada um dos trechos do sinal. A equação (2.1) descreve o processo acima para um sinal discreto $x[n]$ e uma janela $w[m]$ para diferentes valores de frequência λ (OPPENHEIM; SCHAFER, 2013).

$$X[n, \lambda) = \sum_{m=-\infty}^{\infty} x[n+m] \cdot w[m] \cdot e^{-j\lambda m} \tag{2.1}$$

A partir desse resultado, é calculada a magnitude de cada janela de tempo, e com isso se obtém um gráfico tridimensional em que o eixo das abscissas indica o tempo, o eixo das ordenadas representa os valores de frequência e a escala de cores indica o valor da potência de cada frequência ao longo do tempo. A figura 4 apresenta um exemplo de um espectrograma construído para uma gravação de teclado digital, construído com uma janela de Hanning de 2048 amostras e com uma sobreposição de 25% entre as janelas.

É importante ressaltar que as outras representações de frequência serão visualmente similares ao espectrograma, diferindo apenas na escala utilizada para os valores de frequência e na medida de magnitude.

2.1.2 Cromagrama

O cromagrama é uma representação espectral que utiliza o conceito musical de escala cromática, que define o tom musical percebido pelos seres humanos em 12 classes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B) que se repetem ao longo da banda audível de frequência (LERCH, 2012). À medida que a escala de frequência é percorrida de forma crescente, cada um dos membros de uma classe de tom terá o dobro da frequência do representante anterior dessa classe.

Como é possível ver na Figura 3, as notas musicais de um piano servem como exemplo dessa escala , com um padrão de oitavas que se repete a cada 12 teclas, onde a

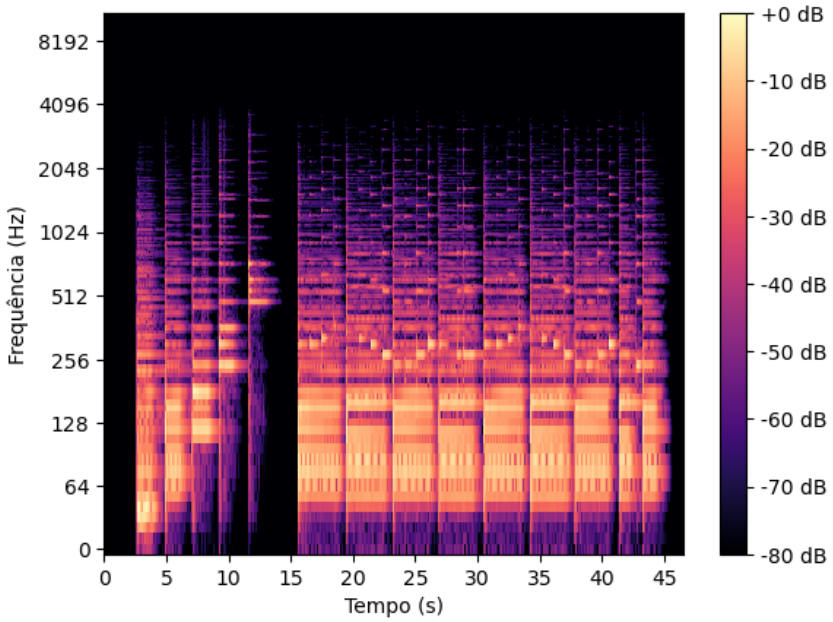


Fig. 2 – Espectrograma de uma gravação de teclado digital.

Fonte: Autoria própria

sexta tecla da primeira oitava tecla do piano pode ser denotada a partir do tom e frequência $A_1 = 55$ Hz, e a próxima da mesma classe será $A_2 = 110$ Hz, seguida por $A_3 = 220$ Hz, $A_4 = 440$ Hz e assim por diante, sempre aumentando em uma razão de 2.

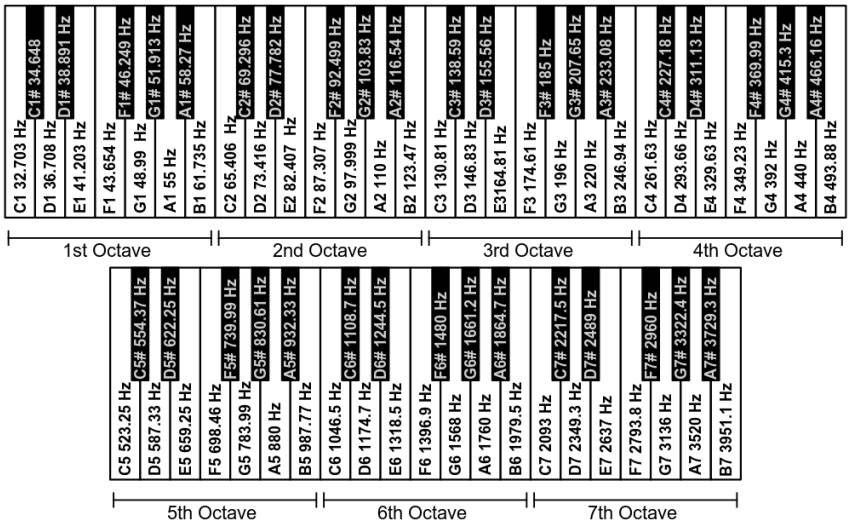


Fig. 3 – Oitavas de um piano e as frequências associadas a cada uma de suas teclas.

Fonte: (DIGILENT. . . , 2023)

Partindo desse princípio, o cromagrama será uma representação de frequência que agrupará a magnitude de todas as componentes de frequência de um sinal de áudio que

pertençam a uma mesma classe tonal em um único coeficiente, resultando em um gráfico que identifica 12 diferentes classes cromáticas obtidas por um somatório de magnitudes.

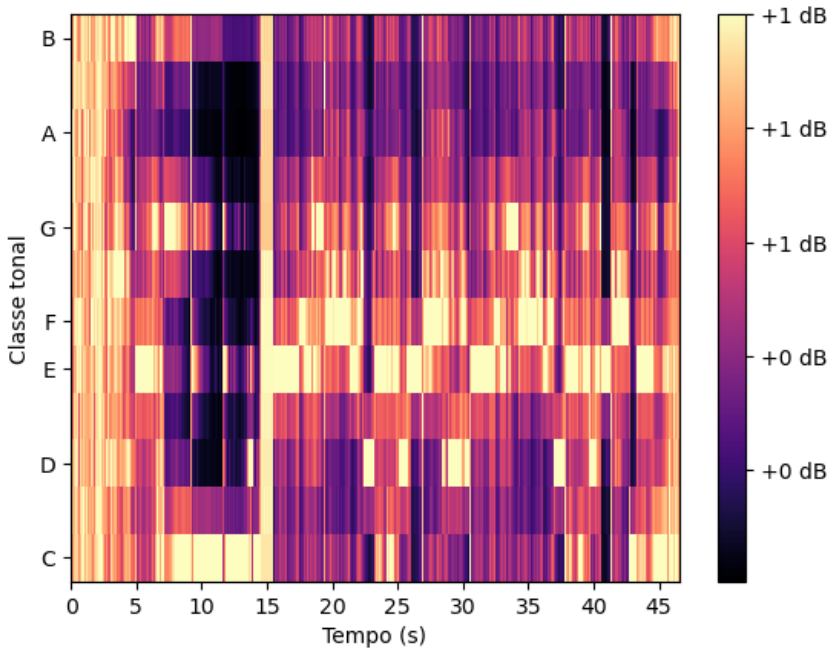


Fig. 4 – Cromagrama de uma gravação de teclado digital.

Fonte: Autoria própria

Implementações típicas do cromagrama envolvem o cálculo de uma STFT logarítmica y_{LF} que é composta apenas por 128 frequências que se encaixam na escala cromática e serão somadas para obter 12 *features* cromáticas da seguinte forma (MÜLLER, 2021):

$$C(n,c) = \sum_{p=0}^{p=128} y_{LF}(n,p), \forall (p \bmod 12 = c) \tag{2.2}$$

onde para essa equação, c assumirá valores no intervalo $[0:11]$ e n representa o índice dentro de uma classe cromática.

2.1.3 MFCC

A representação em MFCC se baseia em uma escala logarítmica de tons que estão diretamente relacionados à frequência de um sinal de áudio, o que permite que a escala se aproxime do que seria a percepção proveniente da audição humana a respeito de quão aguda ou grave é uma nota. Essa representação é tipicamente usada para sistemas de reconhecimento de fala (MÜLLER, 2021), mas também pode ser aplicada para processamento de sinais musicais (LERCH, 2012).

Para realizar o cálculo dos coeficientes cepstrais de um sinal, é necessário seguir alguns passos: primeiro, se obtém um espectro de frequências a partir da STFT. Esse espectro de frequências será convertido em frequências Mel através da utilização de um banco de filtros triangulares passa-faixa, onde as frequências centrais desses filtros serão espaçadas logaritmicamente a partir da seguinte função que mapeia uma frequência f em Hz para uma frequência m_S na escala Mel (LERCH, 2012):

$$m_{\mathrm{S}}(f) = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \text{ Hz} \tag{2.3}$$

Após isso, é calculada a potência de cada uma das frequências Mel em uma janela, o que permitirá calcular os coeficientes cepstrais através do cálculo da *Discrete Cosine Transform* (DCT) do logaritmo das magnitudes obtidas. Devido a forma como esses coeficientes cepstrais são calculados, é possível interpretá-los como um valor que representa o quanto o formato de um determinado sinal se aproxima de uma cossenoide com uma determinada frequência, servindo para detectar ecos em um sinal (OPPENHEIM; SCHAFER, 2013) e permitindo condensar essa informação a respeito do envelope espectral do sinal na forma de coeficientes numéricos (JEEVAN et al., 2017). Na figura 5, é possível ver uma representação em MFCC com 40 coeficientes, onde a escala de cores indica a a quefrência de cada coeficiente em uma unidade de tempo u.t.

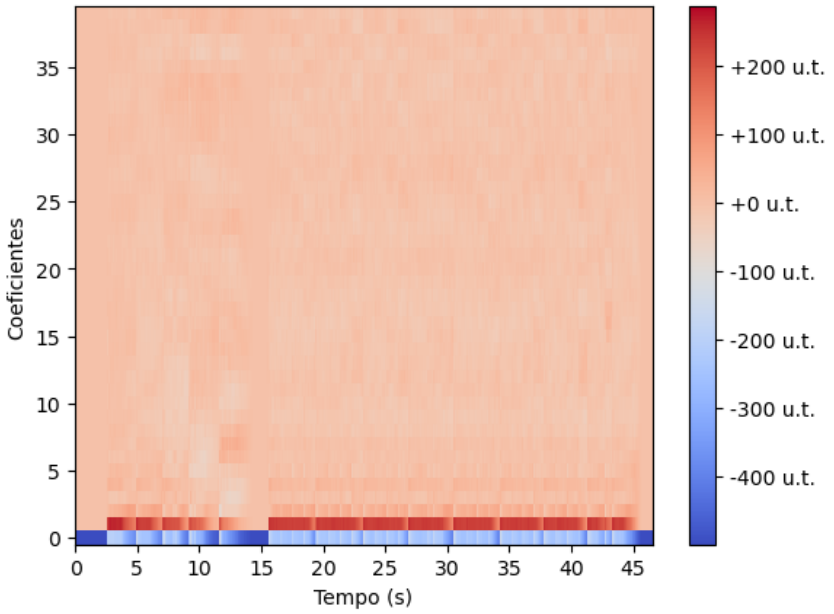


Fig. 5 – MFCC com 40 coeficientes de uma gravação de teclado digital.

Fonte: Autoria própria

A equação 2.4 mostra a definição matemática do j-ésimo coeficiente de MFCC onde $|X'(k', n)|$ representa a magnitude do espectro de frequências Mel. A partir da magnitude,

será calculada a transformada discreta do cosseno do seu logaritmo para o coeficiente de índice j através de uma quantidade $\mathcal{K}'$ de diferentes frequências da função cosseno. A quantidade de coeficientes calculados costuma variar de 4 a 20 (LERCH, 2012), porém pode assumir valores superiores.

$$v^{j}_{\text{MFCC}}(n) = \sum_{k'=1}^{\mathcal{K}'} \log\left(\left|X'\left(k', n\right)\right|\right) \cdot \cos\left(j \cdot \left(k' - \frac{1}{2}\right) \frac{\pi}{\mathcal{K}'}\right) \tag{2.4}$$

2.1.4 GFCC

Assim como a Escala Mel, é possível obter componentes cepstrais após realizar um mapeamento baseado em filtros de frequência para uma escala que simule a audição humana. A representação em *Gammatone Frequency Cepstral Coefficients* (GFCC) irá utilizar filtros *Gammatone* (LERCH, 2012) que serão multiplicados com o sinal no domínio da frequência. Após isso, é aplicada uma Transformada Inversa de Fourier para que o sinal volte para o domínio do tempo e seja sub-amostrado. Por fim, a raiz cúbica do sinal será aplicada ao valor absoluto do sinal e a sua DCT será calculada para se obter os coeficientes desse sinal, num processo similar ao que foi realizado para a MFCC (JEEVAN et al., 2017). A figura 11 exemplifica uma representação de uma GFCC com 40 coeficientes.

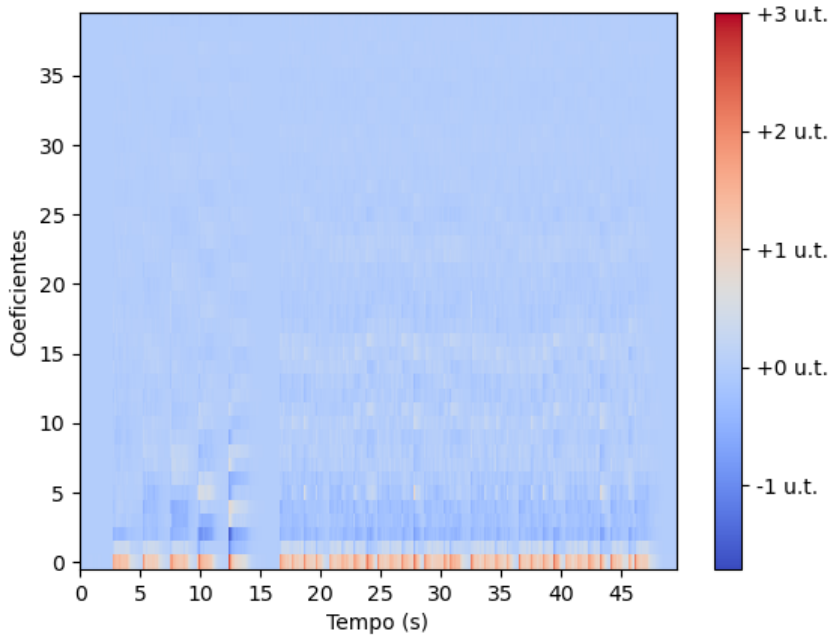


Fig. 6 – GFCC com 40 coeficientes de uma gravação de teclado digital.

Fonte: Autoria própria

2.1.5 Escalograma

O escalograma é uma representação em frequência que é obtida através do cálculo da magnitude da *Continuous Wavelet Transform* (CWT) aplicada a um sinal. Essa transformada tem como princípio a utilização de uma função *wavelet*, que define uma janela de tamanho variável que será convoluída com o sinal ao longo do tempo (DOMINGUES et al., 2016).

Ao se utilizar a transformada de Fourier, é necessário fazer uma relação de compromisso entre uma boa resolução de um evento no domínio do tempo ou uma boa resolução no domínio da frequência, já que quando a STFT é aplicada, a janela $w[m]$ definida para 2.1 possui um tamanho fixo ao longo de todo o sinal. A grande vantagem introduzida pela transformada *wavelet* é a possibilidade de realizar uma análise multirresolução do sinal por permitir uma variação do tamanho da janela. É possível ver nas figuras 2.1.5 e 8 uma comparação entre as janelas utilizadas na STFT com uma resolução tempo-frequência constante e as janelas obtidas através de *wavelets* com diferentes resoluções. Estão representadas através das cores verde, vermelha e azul na figura 2.1.5 as diferentes janelas *wavelet* com suas variações nas larguras temporal Δt e frequencial $\Delta\xi$, assim como a janela de larguras fixas da STFT na figura 8. Para as janelas *wavelet*, à medida que se deseja determinar valores maiores de frequência x, utiliza-se uma janela com um maior $\Delta\xi$ e menor Δt.

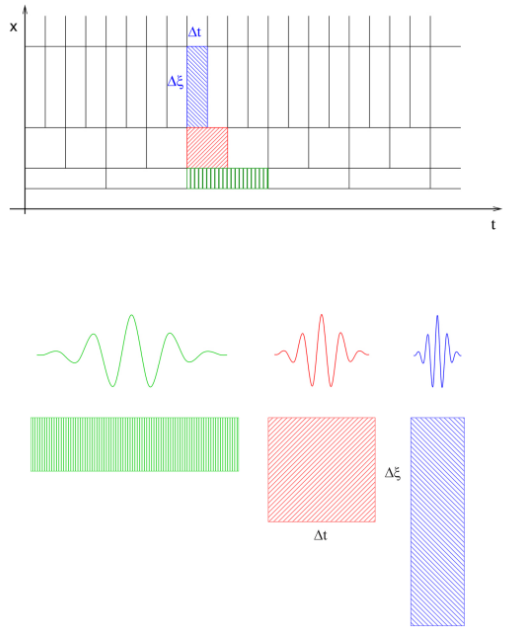


Fig. 7 – Relação entre tempo e frequência para diferentes janelas obtidas através da transformada *wavelet*.

Fonte: (DOMINGUES et al., 2016)

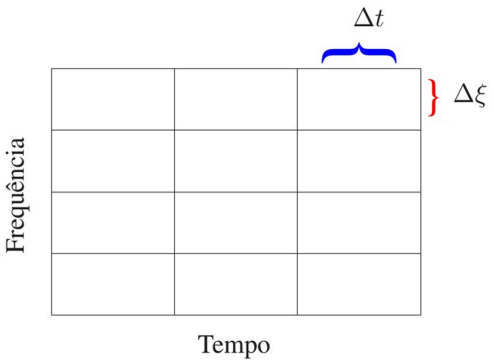


Fig. 8 – Relação entre a frequência e o tempo para uma janela da transformada de Fourier.

Fonte: (DOMINGUES et al., 2016)

A CWT de um sinal $f(t)$ pode ser definida da seguinte forma:

$$\mathcal{W}_f^{\psi}(a,\tau) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t)\psi^* \left(\frac{t-\tau}{a}\right) dt \tag{2.5}$$

onde a representa o valor de escala que irá dilatar ou contrair a janela da *wavelet*, τ representa a translação da janela ao longo do tempo, ψ^* é o complexo conjugado da função *wavelet* utilizada e por fim, $\mathcal{W}_f^{\psi}(a,\tau)$ é o coeficiente *wavelet* calculado (DOMINGUES et al., 2016).

Como é possível perceber de forma analítica, na equação 2.5, que define a CWT, quando a transformada é calculada com um valor grande para a escala a, uma versão da *wavelet* com uma janela dilatada na dimensão temporal estará sendo utilizada, permitindo detectar eventos de baixa frequência. De maneira análoga, para valores de escala a pequenos, a janela será contraída no tempo, permitindo detectar eventos de alta frequência.

Essas características a tornam uma transformada adequada para extrair informações de sinais não-estacionários, como sinais de áudio (DOMINGUES et al., 2016), já que eventos sonoros que ocorrem em um mesmo sinal com diferentes frequências associadas poderão ser bem representados através da transformada. É possível encontrar exemplos na literatura científica da CWT sendo utilizada para a estimação de *pitch* (GERSEM; MOOR; MOONEN, 1997) (KUMAR; KUMAR, 2020) ou para determinar o andamento musical de faixas musicais (VIANA; JÚNIOR; FILHO, 2023).

Por fim, ao realizar o cálculo da potência de um conjunto de coeficientes *wavelet* com diferentes valores de escalas aplicadas a um mesmo sinal, é possível obter um escalograma (DOMINGUES et al., 2016) contendo informações de eventos com diferentes frequências.

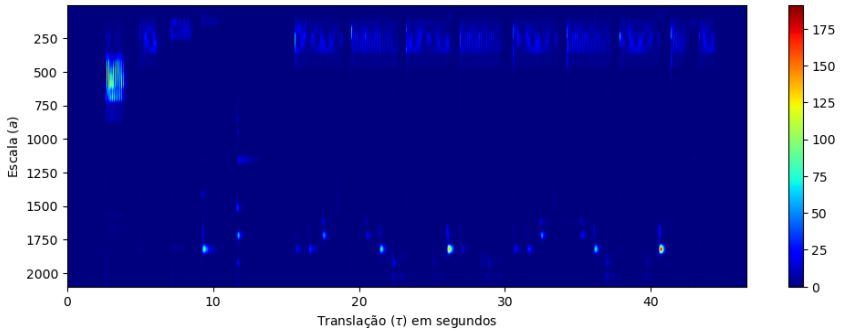


Fig. 9 – Escalograma com 40 escalas no intervalo de valores de 1,7 até 2147 obtido a partir de uma gravação de teclado digital, com a utilização de uma *wavelet* da família Shannon.

Fonte: Autoria própria

2.1.6 HCQT

A *Constant Q-Transform* (CQT) é uma transformada que, assim como a CWT, irá utilizar janelas de tamanho variável para obter melhores resoluções, podendo até mesmo ser considerada uma transformada *wavelet* (WANG et al., 2019) (SCHÖRKHUBER; KLAPURI, 2010). Como um diferencial, a escala de frequência utilizada é uma escala logarítmica que reflete as frequências fundamentais da escala cromática, de forma que as janelas de frequência a serem analisadas estarão espaçadas de uma forma mais similar à percepção humana.

Todas as janelas utilizadas para o cálculo da transformada possuirão um fator de qualidade Q que será constante e é definido como a razão $Q = \frac{f_k}{\delta_k}$ entre o coeficiente de frequência central f_k que será calculada e a largura espectral de banda δ_k da janela do *bin* de frequência de índice k (LERCH, 2012). Dessa forma, à medida que frequências mais elevadas são calculadas, o valor de δ_k também aumenta de forma a manter o mesmo fator de qualidade e obter uma melhor resolução temporal, resultando em janelas com um número de amostras $N[k] = \frac{s}{\delta_k} = Q\frac{s}{f_k}$ cada vez menor, onde s indica a taxa de amostragem. (WANG et al., 2019). A seguinte equação define como será feito o cálculo de $X_{\text{CQT}}[k, m]$ para um *bin* de frequência de índice k com um deslocamento entre janelas m (WANG et al., 2019):

$$X_{\text{CQT}}[k, m] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k, n-m]x[n]e^{\frac{-j2\pi Qn}{N[k]}} \tag{2.6}$$

Uma variação dessa transformada, conhecida como a HCQT foi apresentada por (WANG et al., 2019) como uma versão mais computacionalmente eficiente de calcular a CQT. Ela consiste em um cálculo da CQT tradicional para frequências mais baixas, e para frequências mais altas a partir de um limiar onde a largura da janela $N[k]$ é superior ao dobro do intervalo de amostras L entre janelas, passa-se a utilizar a primeira janela com largura superior a esse intervalo para todas as frequências acima dela.

2.2 Efeitos de guitarra

Efeitos de guitarra são sistemas com a capacidade de alterar características sonoras provenientes de uma guitarra ou contrabaixo elétrico, permitindo que um músico consiga extrair sons com diferentes qualidades de um mesmo instrumento com a utilização de um ou mais desses efeitos. Os efeitos podem ser aplicados ao som de um instrumento de forma analógica, com pedais constituídos de circuitos que irão interagir diretamente com os sinais elétricos e fornecer um sinal modificado na sua saída (EICHAS; FINK; ZÖLZER, 2015), ou de forma digital, seja com pedais que realizam a conversão do sinal analógico para um sinal digital e o filtram em tempo real ou através de modelos que replicam o

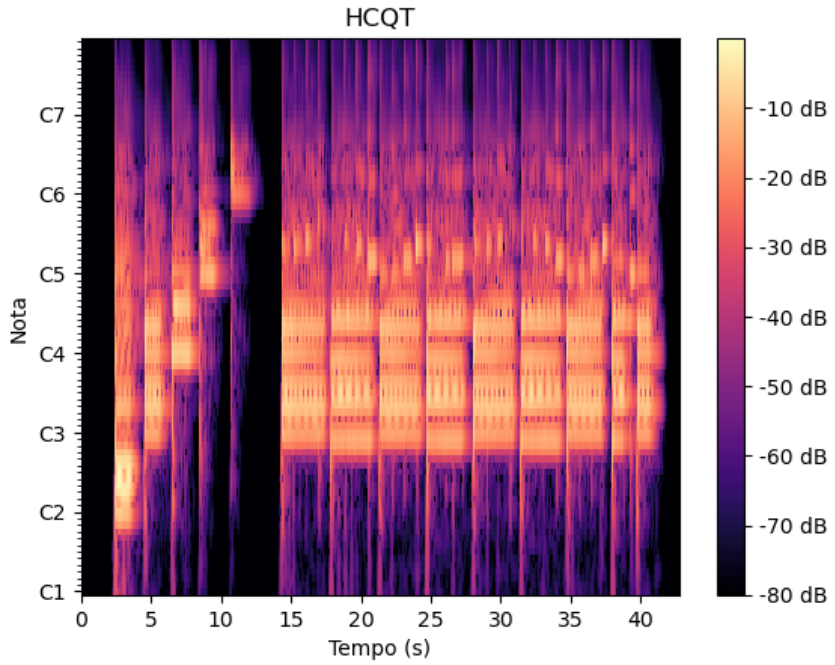


Fig. 10 – HCQT com 84 *bins* de frequência obtida a partir de uma gravação de teclado digital.

Fonte: Autoria própria

funcionamento de pedais analógicos que podem ser aplicados a gravações de áudio como parte do processo de pós-produção de uma música.

De acordo com (EICHAS; FINK; ZÖLZER, 2015), é possível agrupar a maioria dos efeitos em três categorias distintas, com base na natureza dos sistemas que os definem. Para os efeitos que foram utilizados nesse estudo, a divisão é feita da seguinte forma:

- *Linear Time Invariant* (LTI), Linear Invariante no Tempo - *Slapback Delay*, *Feedback Delay*, *Reverb*
- *Linear Time Variant* (LTV), Linear Variante no Tempo - *Tremolo*, *Vibrato*, *Chorus*, *Flanger*, *Phaser*
- *Non-Linear Time Invariant* (NLTI), Não-Linear Invariante no Tempo - Distorção, *Overdrive*

2.2.1 *Linear Time Invariant* (LTI)

A primeira categoria de efeitos de guitarra definida como LTI, se caracteriza por dois conceitos fundamentais para o estudo de sinais e sistemas.

O primeiro deles é a linearidade de um sistema, que pode ser definido através de duas propriedades que devem ser atendidas simultaneamente por um sistema para que ele possa ser considerado linear. A primeira delas é a propriedade da aditividade, que requer que o valor de saída de um sistema ao receber como entrada dois valores $x_1[n]$ e $x_2[n]$ somados deve ser igual ao valor da soma das saídas de cada sinal individualmente aplicado como entrada desse sistema. A segunda delas é a homogeneidade, onde é necessário que, para um sinal $x[n]$, a saída do sistema $y[n]$ multiplicada por uma constante a seja igual à saída de $y[n]$ para esse mesmo sinal $x[n]$ multiplicado por a.

Essas duas propriedades de um sistema de tempo discreto com entradas $x_1[n]$ e $x_2[n]$ podem ser condensadas na equação 2.7 do princípio da superposição (OPPENHEIM; SCHAFER, 2013):

$$T\{ax_1[n] + bx_2[n]\} = aT\{x_1[n]\} + bT\{x_2[n]\} \tag{2.7}$$

Já a invariância no tempo está associada ao comportamento da saída em função da entrada em diferentes momentos no tempo. Espera-se que para um sinal $x[n]$ cujo valor numérico é igual a uma versão deslocada no tempo $x[n] = x[n - n_0]$, a sua saída $y[n]$ deverá ser igual a $y[n - n_0]$ para todo e qualquer valor de deslocamento de amostras n_0 (OPPENHEIM; SCHAFER, 2013).

2.2.1.1 *Slapback Delay*

O *Slapback Delay* é implementado através da aplicação de uma versão atrasada do sinal original multiplicado por um fator de ganho w, sendo considerado um efeito de *delay* simples.

$$y(t) = x(t) + w \cdot x(t - t_0) \tag{2.8}$$

O efeito pode ser percebido de forma audível como uma única e rápida repetição aplicada ao som original. A literatura aponta que o intervalo de tempo utilizado para esse repetição costuma ser pequeno, com valores típicos para t_0 contidos nos intervalos de $10ms$ a $25ms$ (ZÖLZER, 2002) ou de $60ms$ a $150ms$ (REISS; MCPHERSON, 2014).

2.2.1.2 *Feedback Delay*

Assim como o *Slapback*, o *Feedback Delay* pode ser definido como a adição entre o sinal original e uma versão atrasada, com a diferença que é introduzida uma realimentação na saída que faz com que o efeito de *delay* seja aplicado mais de uma vez ao sinal, gerando um efeito de múltiplos ecos.

$$y(t) = x(t) + w \cdot y(t - t_0) \tag{2.9}$$

Na construção do sistema que aplica esse efeito, estará presente novamente o valor t_0 de atraso e o valor de ganho w, onde este último não só influenciará na intensidade da repetição ouvida, como também afetará diretamente no número de repetições que serão aplicadas.

2.2.1.3 *Reverb*

O efeito de *Reverb* ou Reverberação simula a sensação acústica de um ambiente físico onde devido à presença de paredes e objetos, o som que chega a um receptor seja constituído de uma combinação do áudio original e de múltiplas versões atrasadas e atenuadas desse mesmo som, como resultado da reflexão das ondas sonoras nesse ambiente (REISS; MCPHERSON, 2014).

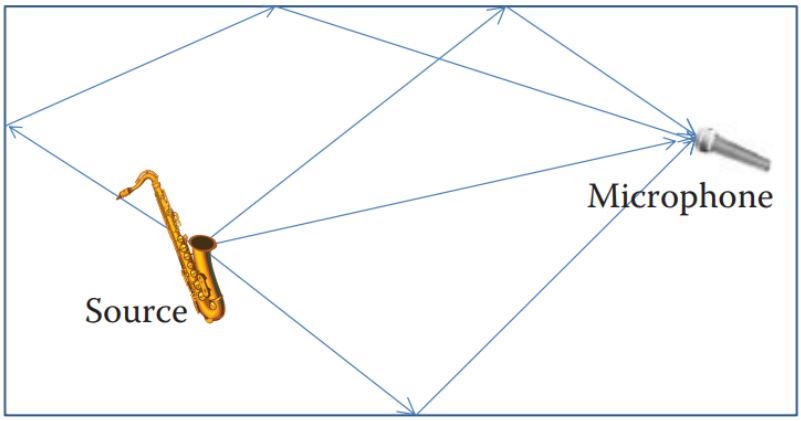


Fig. 11 – Efeito de reverberação ilustrado através dos diferentes caminhos percorridos pela onda sonora entre a fonte emissora e o receptor.

Fonte: (REISS; MCPHERSON, 2014)

É possível interpretar esse efeito como vários *delays* diferentes sendo aplicados simultaneamente; porém devido à complexidade introduzida pelos parâmetros geométricos e da taxa de absorção da energia mecânica de cada um dos materiais presentes em um determinado ambiente, esse efeito costuma ser implementado com base em uma modelagem da resposta ao impulso daquele ambiente no formato de um filtro *Finite Impulse Response* (FIR). Dessa forma, ele pode ser representado matematicamente pela convolução do sinal com a resposta ao impulso h_{rev} modelada:

$$y(t) = x(t) * h_{rev}(t) \tag{2.10}$$

2.2.2 *Linear Time Variant* (LTV)

Os efeitos LTV são constituídos de sistemas que atendem ao princípio da linearidade, porém a sua entrada e saída não possuem uma relação de invariância no tempo, devido à presença de um Low Frequency Oscillator (LFO) que irá introduzir variações na saída periodicamente (EICHAS; FINK; ZÖLZER, 2015). Dessa forma, mesmo para valores constantes de entrada caracterizados por $x[n] = x[n - n_0]$, a saída assumirá valores diferentes.

O LFO pode ser definido como qualquer sinal periódico que oscile em função do tempo e tenha uma frequência de oscilação abaixo de 20 Hz (REISS; MCPHERSON, 2014). As equações 2.11 e 2.12 descrevem, respectivamente, a versão senoidal de tempo contínuo e a versão de tempo discreto de um LFO com amplitude W, frequência f e frequência de amostragem f_s.

$$X_{LFO}(t) = W \sin(2\pi f t) \tag{2.11}$$

$$X_{LFO}[n] = W \sin(2\pi f n / f_s) \tag{2.12}$$

2.2.2.1 *Tremolo*

O efeito de *Tremolo* é uma modulação na amplitude do sinal de áudio, fazendo com que ela aumente e diminua em função de uma oscilação periódica introduzida por um LFO com uma função de onda que será tipicamente senoidal, triangular ou quadrada (EICHAS; FINK; ZÖLZER, 2015).

$$y(t) = x(t) \cdot \alpha x_{LFO}(t) \tag{2.13}$$

Para isso, é feita uma multiplicação do sinal de áudio original pelo sinal modulante x_{LFO}, que terá frequências de 0.5 Hz até 20 Hz (REISS; MCPHERSON, 2014) (ZÖLZER, 2002), e cujo parâmetro de profundidade α assume valores entre 0 e 1 e irá determinar qual será o incremento ou redução no volume audível do sinal.

2.2.2.2 *Vibrato*

O *Vibrato* é um efeito caracterizado por uma oscilação na frequência do sinal de áudio. De forma similar ao efeito *Doppler*, onde o aumento na distância de uma fonte emissora de ondas sonoras faz com que a frequência do sinal de áudio diminua, o efeito de *Vibrato* será gerado a partir de um *delay* variável que irá aumentar ou diminuir o seu parâmetro de tempo através de um LFO. Diferentemente dos efeitos de *delay* abordados

anteriormente, o efeito não será somado ao sinal original, e sim aplicado diretamente a ele, como é possível ver na equação 2.14 (EICHAS; FINK; ZÖLZER, 2015):

$$y(t) = x(t - x_{LFO}(t)) \tag{2.14}$$

Os parâmetros de *delay* e frequência do LFO tipicamente assumirão valores entre $5ms$ e $10ms$ e 5 Hz e 14 Hz, respectivamente (DUTILLEUX, 1998).

2.2.2.3 *Chorus*

O *Chorus* é um efeito de pedal que, como o nome indica, simula o som que seria obtido por um coro de instrumentos tocando em uníssono. Por ser humanamente impossível sincronizar dois instrumentos de forma perfeita durante uma *performance* musical, o efeito simula essa característica através da introdução de uma ou mais versões atrasadas do sinal original com valores de *delay* que oscilarão periodicamente através de um LFO e serão somados com um ganho w ao sinal original.

$$y(t) = x(t) + w \cdot x(t - x_{LFO}(t)) \tag{2.15}$$

É possível considerar o *Chorus* como uma combinação de um sinal original com a aplicação de um *Vibrato* (EICHAS; FINK; ZÖLZER, 2015), com a diferença que os valores de *delay* LFO, que costuma ser senoidal, normalmente estarão numa faixa de $10ms$ a $25ms$ (ZÖLZER, 2002) ou $5ms$ a $30ms$ (REISS; MCPHERSON, 2014), e cuja frequência estará entre 3 Hz e 20 Hz.

2.2.2.4 *Flanger*

O *Flanger* é um efeito similar em construção ao *Chorus* mas que distingue-se pela forma como ele é captado pela percepção auditiva humana (REISS; MCPHERSON, 2014). Matematicamente, ele também pode ser definido como a adição entre o sinal original e uma versão que é atrasada por um *delay* que varia em função de um LFO. Uma das grandes diferenças em relação ao efeito de *Chorus* é a possibilidade de implementá-lo com uma arquitetura de realimentação do sinal atrasado, como é possível ver na equação 2.16:

$$y(t) = x(t) + w \cdot x(t - x_{feedback}(t)) \tag{2.16}$$

Onde $x_{feedback}(t)$ é definido como o seguinte sinal realimentado:

$$x_{feedback}(t) = x(t - x_{LFO}(t)) + w_{FB} \cdot x_{feedback}(t - x_{LFO}(t)) \tag{2.17}$$

Outras características que diferenciam o *Flanger* são menores valores de *delay* que variam de $0ms$ a um máximo de $15ms$ (EICHAS; FINK; ZÖLZER, 2015), gerando interferências construtivas e destrutivas marcados por picos e vales no sinal de áudio. Além disso, apesar de o efeito de *Flanger* poder contar com ciclos de *feedback* na sua implementação, o efeito usualmente se limita a ter um único sinal atrasado adicional.

2.2.2.5 *Phaser*

O *Phaser*, que possui esse nome devido à alteração na fase das frequências presentes em um sinal de áudio. Ele também é um efeito que gera picos e vales no sinal de áudio, porém alcança isso através da utilização de um filtro passa-tudo que não irá alterar a magnitude das frequências do sinal, mas sim introduzir atrasos na fase de cada uma delas (REISS; MCPHERSON, 2014).

O atraso que será introduzido na fase das diferentes frequências irá depender do valor da frequência central do filtro; valor esse que irá variar em função de um LFO. A saída resultante desse efeito é obtida através de uma soma de uma versão original do sinal com a versão que é filtrada, e pode ser descrita pela equação 2.18 (EICHAS; FINK; ZÖLZER, 2015):

$$y(t) = d \cdot x(t) + w \cdot \mathrm{AP}^{M} \left\{ (x(t), x_{\mathrm{LFO}}(t)) \right\} \tag{2.18}$$

Onde d e w são valores entre 0 e 1 que indicam as proporções do sinal original e filtrado que serão utilizados na saída e AP é o filtro passa-tudo de ordem M que terá sua frequência central alterada em função de x_{LFO}, para que seja feita uma convolução entre AP e o sinal $x(t)$. A ordem do filtro influencia na quantidade de picos presentes no sinal (EICHAS; FINK; ZÖLZER, 2015).

2.2.3 *Non-Linear Time Invariant* (NLTI)

A última classe de efeitos irá utilizar funções não-lineares para realizar alterações no sinal de entrada, o que faz com que eles não atendam ao princípio da superposição, mas continuem atendendo ao princípio da invariância ao tempo. Esses efeitos são conhecidos por aumentarem a amplitude do sinal até que eles cheguem aos valores máximos de tensão para sistemas analógicos ou quantidade de bits para sistemas digitais, gerando uma saturação ou *clipping* do sinal (REISS; MCPHERSON, 2014).

2.2.3.1 Distorção

A distorção, que costuma englobar diferentes categorias de efeitos não-lineares subdivididos entre *overdrive*, distorção e *fuzz* com base no grau de distorção introduzido

por cada um deles (EICHAS; FINK; ZÖLZER, 2015), é definida matematicamente como a aplicação de uma função não-linear $f(x)$ a um sinal de entrada $x(t)$.

A curva característica da função de distorção irá resultar em um *clipping* mais pronunciado, definido como *hard clipping*, por promover grandes variações no valor de amplitude do sinal de saída. Uma função típica utilizada para o efeito de distorção é dada por:

$$f(x) = \begin{array}{ccc} -1 & para & Gx \leq -1 \\ Gx & para & -1 < Gx < 1 \\ 1 & para & Gx \geq 1 \end{array} \tag{2.19}$$

no qual G representa um ganho controlável que afeta o quão pequena uma variação no valor de x irá gerar um *clipping*. (REISS; MCPHERSON, 2014)

2.2.3.2 *Overdrive*

O *overdrive* é uma versão de distorção que apresenta uma saturação mais suavizada do que os demais. O *soft clipping* é atingido com a utilização de uma função que possui porções lineares e não-lineares, fazendo com que valores mais baixos de x sejam pouco afetados pela distorção. Abaixo está descrita uma equação que pode ser utilizada para representar o efeito de *overdrive*:

$$f(x) = \begin{array}{ccc} 2x & para & 0 \leq x < 1/3 \\ 1 - (2 - 3x)^2/3 & para & 1/3 \leq x < 2/3 \\ 1 & para & 2/3 \leq x \leq 1 \end{array} \tag{2.20}$$

O fenômeno da suavização pode ser visualizado pela amplitude de um sinal de áudio no domínio do tempo. A figura 12 apresenta a diferença no formato de um mesmo sinal de áudio após os efeitos de *overdrive* e distorção serem aplicados a ele.

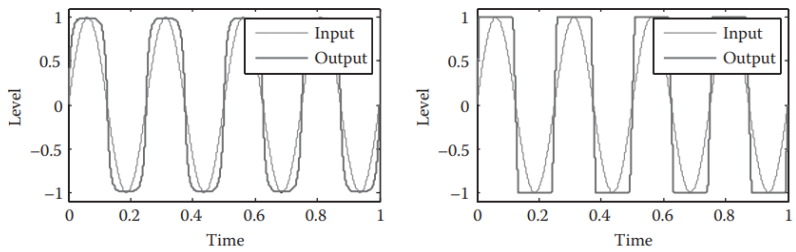


Fig. 12 – Faixa de áudio após passar por *soft* e *hard clipping*.

Fonte: (REISS; MCPHERSON, 2014)

2.3 CNN

Antes de conceituar uma *Convolutional Neural Network* (CNN), é necessário trazer à tona o conceito de redes neurais artificiais. Uma rede neural artificial é um modelo computacional que se inspira no funcionamento do cérebro humano para definir unidades computacionais denominadas neurônios artificiais (FACELI et al., 2011), também conhecidos como *perceptrons* (HAYKIN, 2009), que irão trabalhar em conjunto. Assim como o neurônio pode ativar um sinal na saída de uma de suas sinapses a partir de impulsos elétricos recebidos em seu dendritos, o *perceptron* pode ser definido matematicamente como uma função matemática que calcula um valor de saída a partir de uma soma ponderada por pesos w de suas entradas x a uma constante de viés b, como é possível ver na figura 13. A função de saída do *perceptron* é denominada função de ativação e costuma ser do tipo não-linear, tendo como exemplos a *Rectified Linear Unit* (ReLU), *softmax* e sigmoide (GOODFELLOW; BENGIO; COURVILLE, 2016).

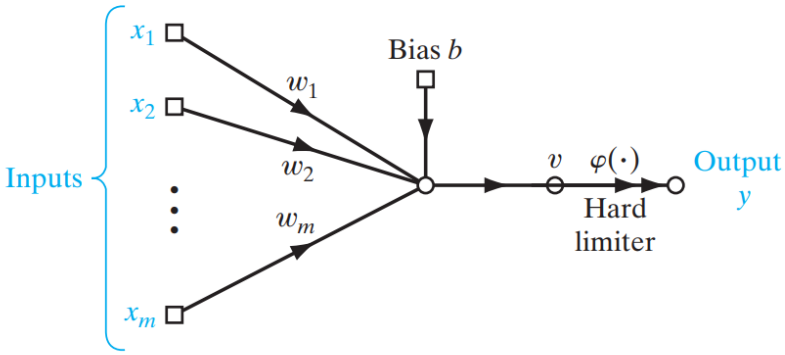


Fig. 13 – Representação da saída de um *perceptron* em função de suas entradas, pesos e viés.

Fonte: (HAYKIN, 2009)

Sempre que a expressão rede neural for utilizada neste trabalho, ela estará se referindo àquelas que são artificiais. Uma rede neural pode ser constituída pela combinação de múltiplos *perceptrons* dispostos em uma sequência contendo uma camada de entrada, uma ou mais camadas intermediárias e uma camada de saída (FACELI et al., 2011); estrutura conhecida como MLP, que comumente será do tipo *feedforward*, onde as entradas do neurônio de uma camada serão exclusivamente saídas dos neurônios de camadas anteriores, e totalmente conectada, onde todas os neurônios de uma camada estarão ligados a todos os neurônios das camadas seguintes, sem realimentações (HAYKIN, 2009). A arquitetura de um MLP está descrita na figura 14 .

No processo de construção de uma rede neural, objetiva-se encontrar uma função de

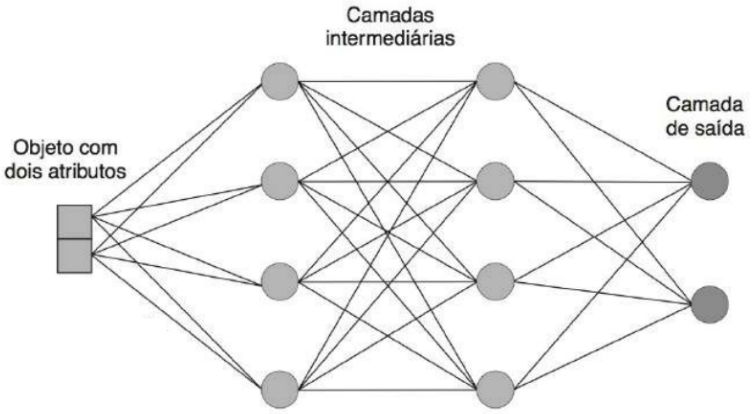


Fig. 14 – Representação de um MLP com duas camadas intermediárias.

Fonte: (FACELI et al., 2011)

predição $f^*(x)$ que se aproxima de uma função real $f(x)$ de interesse de uma determinada tarefa e que estabelece uma relação entre uma entrada x e uma categoria (para problemas de classificação) ou valores numéricos (para problemas de regressão) $y = f^*(x)$ na sua saída (GOODFELLOW; BENGIO; COURVILLE, 2016). Para obter essa função, cuja complexidade influencia na quantidade de camadas e de neurônios necessários para aprender padrões não-lineares, é necessário realizar um processo de treinamento da rede neural que defina valores de pesos w e viés b adequados para cada um dos seus neurônios. Esse processo de treinamento se resume a um problema de otimização do valor de erro de uma função custo que indica o quão próxima uma predição realizada pela rede está do valor de saída esperado para uma entrada x (GOODFELLOW; BENGIO; COURVILLE, 2016).

No processo de aprendizado pertencente ao paradigma supervisionado (FACELI et al., 2011) serão fornecidos exemplos de um *dataset* de treinamento para os quais as entradas x e saídas y já foram previamente mapeadas para que, iterativamente, os valores de pesos da rede sejam ajustados de forma a reduzir o erro da função custo com a expectativa de, indiretamente, melhorar uma medida associada. Uma função de custo tipicamente utilizada é a entropia cruzada, que calcula a diferença entre a distribuição de probabilidade de uma saída y com os valores preditos (GOODFELLOW; BENGIO; COURVILLE, 2016). Com base no valor obtido para a função custo para cada iteração dos dados de treinamento, será possível calcular novos valores para todos os pesos dos neurônios de um MLP através de um algoritmo denominado *backpropagation*, que conta com diferentes implementações e foi responsável por viabilizar a utilização de MLPs. (FACELI et al., 2011).

O processo de treinamento é descrito por um conjunto de diferentes hiperparâmetros (GOODFELLOW; BENGIO; COURVILLE, 2016) que influenciam na duração, velocidade e na capacidade de obter uma curva de função de custo que convirja para um valor de

erro pequeno. Dentre eles, destacam-se a taxa de aprendizado α, que pondera os ajustes nos pesos calculados a cada iteração; as épocas, que representam a quantidade de vezes na qual todos os exemplos do conjunto de testes serão apresentados à rede neural (HAYKIN, 2009); e o tamanho dos lotes, que definem a quantidade de exemplos de treinamento que devem ser apresentados à rede neural antes que os novos pesos sejam calculados.

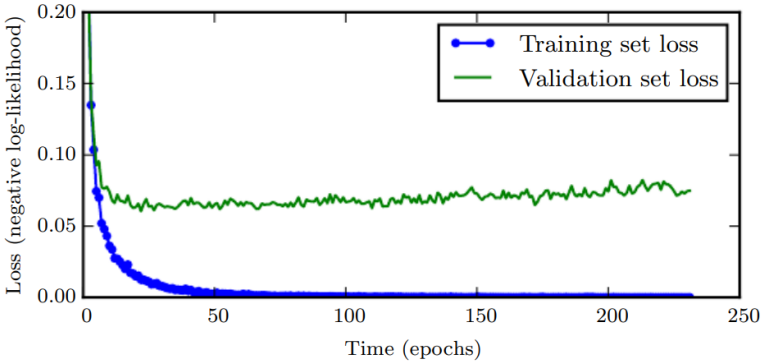


Fig. 15 – Exemplo de curvas do erro da função custo calculadas para os conjuntos de treinamento e validação.

Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016)

Na figura 15, é possível ver a curva de treinamento de uma rede neural, onde é observado que o erro de uma função custo calculada para o conjunto de treinamento vai gradualmente diminuindo com o passar das épocas até se aproximar de zero, enquanto o mesmo não ocorre com o erro para o conjunto de validação, que é definido como uma porção reduzida dos dados que não é utilizada no processo de treinamento, mas serve para avaliar a capacidade de generalização do modelo de rede neural quando apresentado a dados não vistos anteriormente. O momento em que não há mais redução no erro para o conjunto de validação indica o momento em que o processo de treinamento pode ser interrompido prematuramente (*early stopping*, pois significa que a rede está começando a realizar um sobreajuste (*overfitting*) para os dados de treinamento, o que a faz lentamente perder a capacidade de generalização e se tornar mais especializada apenas no conjunto de dados de treinamento. É possível identificar nesse exemplo que à medida que a quantidade de épocas avança, o erro da função custo para o conjunto de validação aumenta, indicando uma piora na capacidade preditiva do modelo. (GOODFELLOW; BENGIO; COURVILLE, 2016).

Com base nos conceitos apresentados acima, é possível definir uma CNN, que se refere a um tipo especial de rede neural voltada para o processamento de dados com duas ou mais dimensões a exemplo de imagens e séries temporais (GOODFELLOW; BENGIO; COURVILLE, 2016). Antes dos dados serem processados pelas camadas típicas de um

MLP, esse tipo de rede irá processar as suas entradas com formato multidimensional através de operações de convolução ou correlação cruzada matriciais realizadas a partir de *kernels*, que são definidos como matrizes de dimensão reduzida com valores internos de peso a serem treinados e que serão multiplicados pelos valores de entrada, como representado na figura 16. Ao conjunto de *kernels* com diferentes pesos de uma mesma camada, dá-se o nome de filtros (GOODFELLOW; BENGIO; COURVILLE, 2016).

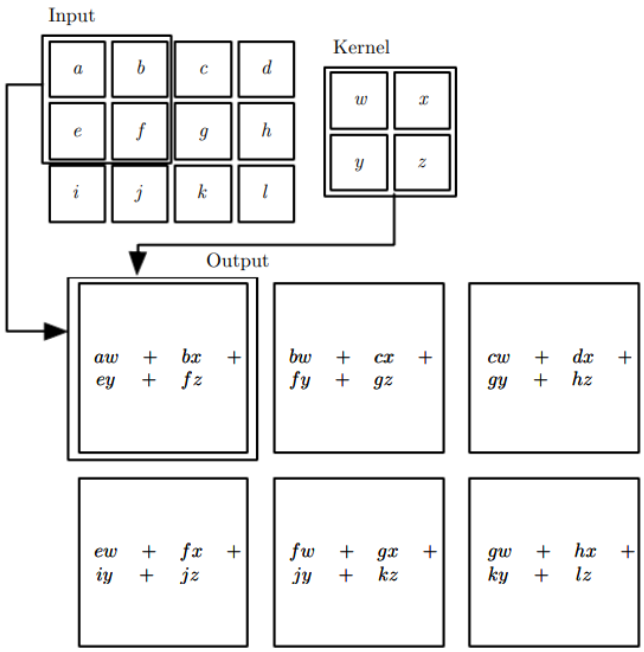


Fig. 16 – Representação da convolução de um *kernel* de tamanho 2x2 por uma matriz de entrada de uma CNN.

Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016)

As camadas de uma rede convolucional contam com três etapas: a etapa de convolução, descrita acima; a etapa de ativação, que irá utilizar os resultados da operação de convolução como entradas de funções de ativação não-linear, e uma etapa de *pooling*, definida por operações matriciais que condensam informações da vizinhança de uma região da matriz de entrada de maneira estatística, tipicamente através de operações de *max pooling* ou *average pooling*, que selecionam os maiores valores e calculam uma média dos valores presentes naquela matriz, respectivamente (GOODFELLOW; BENGIO; COURVILLE, 2016). A utilização de camadas convolucionais para lidar com dados bidimensionais apresenta diversas vantagens, destacando-se a redução de dimensionalidade proporcionada pelas operações de *pooling* e a redução da quantidade total de pesos para dados de entrada muito grandes (HAYKIN, 2009).

Existem diferentes maneiras de construir as arquiteturas de CNNs para as mais

variadas tarefas, onde as mais conhecidas são aplicadas no reconhecimento de imagens e temos como exemplo as arquiteturas VGG16 (SIMONYAN; ZISSERMAN, 2015), GoogLeNet (SZEGEDY et al., 2014) e a AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2017). Na figura 17, é possível observar as camadas com suas respectivas dimensões de uma CNN de arquitetura VGG16, composta por 6 camadas convolucionais e de *max pooling* que vão de *conv1* até *conv6*, e de 3 camadas totalmente conectadas, denotadas por *fc6*, *fc7* e *fc8*.

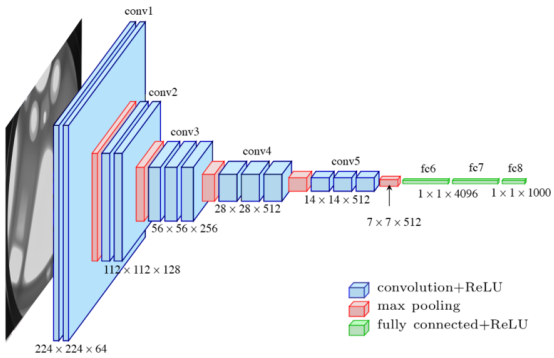


Fig. 17 – Arquitetura VGG16 de CNN, contendo um conjunto de camadas convolucionais e totalmente conectadas.

Fonte: (FERGUSON et al., 2017)

3 Metodologia

Neste capítulo, serão destrinchados materiais, técnicas, métricas e outros aspectos relevantes para a construção do modelo computacional capaz de identificar os efeitos de pedal presentes em trechos de áudio. Iniciaremos com uma descrição dos *datasets* contendo áudios com diferentes efeitos de pedal. Em seguida, será abordado o processo de construção das representações em frequência a partir dos áudios presentes nos *datasets* e que serão utilizados para treinar a rede. Após isso, iremos entrar em detalhes a respeito das camadas e da arquitetura da CNN, e por fim, como se deu o processo de treinamento da rede para a tarefa de classificação.

3.1 *Datasets*

Os dois *datasets* utilizados nesse trabalho são o GEC-GIM, construído por (HINRICHS et al., 2022), e o IDMT-SMT, introduzido em (STEIN, 2010). Ambos são *datasets* idealizados especificamente para a tarefa de classificação de efeitos de pedal e são constituídos de faixas de áudio no formato .wav de curta duração, amostrados a 44,1 kHz e com *bitrate* de 705kbps, contendo o som de guitarras tocando uma nota com diferentes efeitos sendo aplicados individualmente em cada faixa.

3.1.1 GEC-GIM

O GEC-GIM é uma coleção de 15840 *samples* de 2s de áudio que foram gerados utilizando uma *Digital Audio Workstation* (DAW) com *plugins* de instrumentos *sample-based*, *i.e.*, o som gerado pelos *plugins* é obtido com base em gravações de instrumentos reais.

As 15840 amostras foram obtidas utilizando:

- 2 guitarras diferentes obtidas por *plugins sample-based*, *Sonivox Bright ElectricGuitar* e *Ample Guitar LP*
- 11 efeitos diferentes (*Chorus*, *Distortion*, *Feedback Delay*, *Flanger*, Equalização, *Overdrive*, *Phaser*, *Reverb*, *Slapback Delay*, *Tremolo*, *Vibrato*)
- 60 combinações diferentes dos parâmetros de intensidade dos efeitos
- 12 combinações de instrumentos diferentes:
  - Guitarra

- Guitarra e baixo
- Guitarra e teclado
- Guitarra e bateria tocando chimbal fechado
- Guitarra e bateria tocando chimbal aberto
- Guitarra e bateria tocando caixa
- Guitarra e bateria tocando bumbo
- Guitarra e bateria tocando prato
- Guitarra, baixo e bateria
- Guitarra, teclado e bateria
- Guitarra, baixo e teclado
- Guitarra, baixo, teclado e bateria

Assim como as guitarras, os outros instrumentos também foram obtidos a partir de *plugins sample-based*: *Bitsonic Keyzone Classic* para o teclado, *Ample Bass P Lite* para o baixo, e *Manda Audio MT POWER DrumKit 2* para a bateria (HINRICHS et al., 2022).

Em cada faixa de áudio, a guitarra e teclado tocaram uma única vez as notas E2 e E3 e o baixo tocou uma única vez as notas E1 e E2, com a nota sendo tocada por 2 segundos, enquanto a bateria tocou os seguintes padrões representados na figura 18 por um total de 2 segundos.

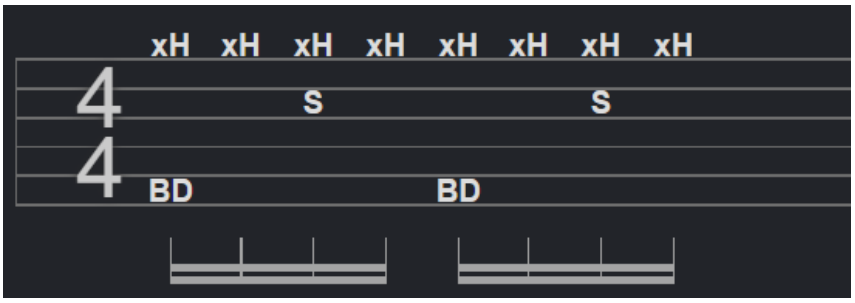


Fig. 18 – Tablatura contendo o padrão de bateria tocado nas faixas por 2 segundos. **BD** indica o bumbo, **S** indica a caixa e **xH** indica o chimbal fechado.

Fonte: Autoria própria

Para a aplicação dos efeitos, os *plugins* de efeitos de guitarra e os parâmetros de cada um deles que foram utilizados na DAW estão detalhados na tabela 1.

3.1.2 IDMT-SMT

O IDMT-SMT é composto por um total de 55044 faixas de áudio. Desse total, 20592 são gravações de notas de baixo monofônicas, 20592 gravações de guitarras monofô-

Tab. 1 – *Plugins* de guitarra utilizados no GEC-GIM.

Fabricante	**Nome do *Plugin***	**Efeito**	**Parâmetros**
Kjaerhus Audio	Classic Chorus	*Chorus*	Taxa, profundidade
Kjaerhus Audio	Classic Chorus	*Vibrato*	Taxa, profundidade
Kjaerhus Audio	Classic Delay	*Feedback delay*	Tempo, realimentação, *mix*
Kjaerhus Audio	Classic Delay	*Slapback delay*	Tempo, *mix*
Kjaerhus Audio	Classic Flanger	*Flanger*	Taxa, profundidade, realimentação
Kjaerhus Audio	Classic Phaser	*Phaser*	Taxa, profundidade
Kjaerhus Audio	Classic Reverb	*Reverb*	Tamanho do ambiente, *mix*
SimulAnalog	Equalizer	Equalizador	Graves, médios, agudos
SimulAnalog	Tube Screamer	*Overdrive*	Ganho, tom
PechenegFX	Pecheneg Tremolo	*Tremolo*	Taxa, profundidade
Buzzroom	OctBUZ	*Reverb*	Ganho, tom

Fonte: (HINRICHS et al., 2022)

nicas, e 13860 gravações de guitarras polifônicas, tocando mais de uma mesma nota ao mesmo tempo. Para manter uma equivalência do conteúdo presente nos *datasets*, assim como em (HINRICHS et al., 2022), apenas as gravações de notas de guitarras sendo tocadas individualmente foram escolhidas. Portanto, apenas as 20592 gravações de guitarra monofônica foram utilizadas, que nesse *dataset* foram obtidas através de instrumentos reais sendo tocados.

As 20592 amostras foram obtidas utilizando:

- 2 guitarras diferentes, *Schecter Diamond C-1 Classic* e *Chester Stratocaster*
- 2 diferentes configurações dos pré-amplificadores de cada guitarra
- 2 maneiras diferentes de tocar a corda da guitarra (*fingerpick* suave ou, palhetada)
- 12 efeitos diferentes (*Chorus*, Distorção, *Feedback Delay*, *Flanger*, *Overdrive*, *Phaser*, *Reverb*, *Slapback Delay*, *Tremolo*, *Vibrato* e Nenhum efeito (dividido entre Equalização e Nenhum efeito)
- 3 combinações diferentes dos parâmetros de intensidade dos efeitos
- 13 diferentes casas dentre as 6 cordas da guitarra, compreendendo as notas que vão de E2 até E5

As faixas de guitarra também duram 2 segundos, embora devido ao fato de terem sido gravadas com instrumentos reais, há pequenas diferenças entre o momento em que a gravação é iniciada e o momento em que o som do instrumento é reproduzido. Os efeitos presentes nas gravações também foram aplicados com o auxílio de uma DAW.

Enquanto no *dataset* GEC-GIM existe uma classe de efeito que é descrita como um equalizador, sem nenhum efeito em específico sendo aplicado, para o *dataset* IDMT-SMT os áudios que se encaixam em classe similar são subdivididos entre áudios limpos, sem nenhum tipo de pós-processamento, e áudios nos quais foi aplicado um *plugin* de DAW que simula um amplificador real, sendo então considerados como áudios que passaram por um efeito de Equalizador. Dessa forma, teremos 11 e 12 classes para os *datasets* GEC-GIM e IDMT-SMT, respectivamente.

3.2 Representações tempo-frequência construídas

O classificador construído por (HINRICHS et al., 2022) foi treinado com 4 diferentes representações no domínio da frequência das amostras de áudio: espectrograma, cromagrama, MFCC e GFCC. Para o presente trabalho, foram geradas duas diferentes representações em frequência para o treinamento da rede neural, a HCQT e o escalograma. Ambas as representações foram obtidas a partir de versões normalizadas dos áudios, em um processo similar ao feito por (HINRICHS et al., 2022).

3.2.1 HCQT

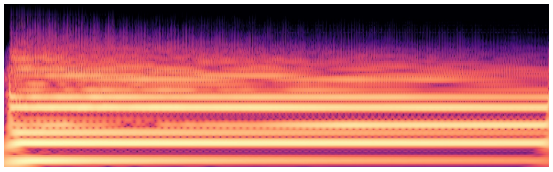
Para as HCQTs, foram geradas duas representações com parâmetros distintos através do uso da biblioteca *librosa* (LIBROSA. . . , 2023):

- HCQT-C1: 12 *bins* de frequência por oitava, 111 *bins* de frequência no total, utilizando uma janela de Hanning com uma frequência mínima a ser calculada de 32.7 Hz.
- HCQT-SUB-C0: 12 *bins* de frequência por oitava, 111 *bins* de frequência no total, utilizando uma janela de Hanning com uma frequência mínima a ser calculada de 10 Hz.

O tipo de janela utilizada foi escolhido com base na implementação padrão para a HCQT utilizada por (WANG et al., 2019), e tanto ela como a quantidade de *bins* de frequência por oitava são utilizados como padrão na biblioteca. Foi aplicado um fator de escala de 0,5 no tamanho da janela que foi selecionado de maneira empírica, treinando a CNN com iterações mais rápidas do modelo utilizado apenas uma fração do *dataset* para avaliar se a resolução de banda das janelas era suficiente para permitir que a rede conseguisse aprender a classificar os exemplos com as representações fornecidas.

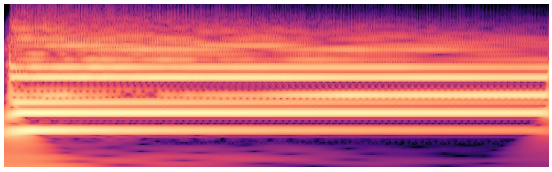
Durante essas avaliações, constatou-se que a CQT original não estava apresentando uma capacidade de aprendizado satisfatória, sendo substituída em favor da HCQT devido à sua melhor resolução para altas frequências e sua maior eficiência computacional (WANG et al., 2019), permitindo gerar as representações de uma forma mais rápida.

Fig. 19 – HCQT-C1 obtida a partir do áudio ag_G+B_Overdrive40_27_.wav do GEC-GIM.



Fonte: Autoria própria

Fig. 20 – HCQT-SUB-C0 obtida a partir do áudio ag_G+B_Overdrive40_27_.wav do GEC-GIM.



Fonte: Autoria própria

A motivação para a construção de 2 HCQTs com diferentes frequências iniciais é investigar se com a introdução de uma maior quantidade de *bins* de baixa frequência, o classificador conseguiria obter uma melhor distinção entre os diferentes efeitos da categoria LTV que são marcados pela utilização de um LFO em suas implementações. Levando isso em conta, assim como o fato de que em ambos os *datasets* as frequências tocadas nas notas de guitarra se concentram nas faixas entre E_2 (82.4 Hz) e E_5 (659 Hz), espera-se que não haja perda de informação relevante ao se ignorar alguns *bins* de frequência mais elevadas.

3.2.2 Escalograma

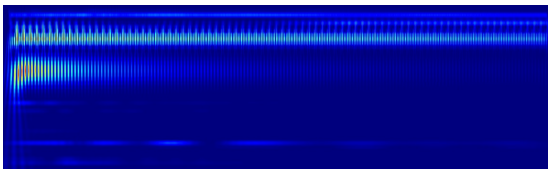
Para os escalogramas, a biblioteca *pywavelets* (PYWAVELETS..., 2023) foi utilizada para realizar o cálculo da CWT com duas diferentes famílias de wavelets, a Morlet e a Shannon (DOMINGUES et al., 2016):

- Morlet_40: Escalograma construído a partir de uma família de *wavelets* Morlet com 40 escalas no intervalo $[2, 57, ..., 2147, 2200]$ com um espaçamento aproximadamente igual entre si, compreendendo as frequências de $[19500, 684, ..., 18.17, 17.71]$ Hz para áudios amostrados a 44,1 kHz.
- Shannon_40: Escalograma construído a partir de uma família de *wavelets* Shannon com 40 escalas no intervalo $[1.7, 57, ..., 2047, 2100]$ com um espaçamento aproximadamente igual entre si, compreendendo as frequências de $[21882, 652, ..., 18.17, 17.71]$ Hz para áudios amostrados a 44,1 kHz.

As duas famílias apresentam formatos e características distintas, sendo possível obter uma melhor resolução frequencial para a família de *wavelets* Shannon e uma melhor resolução temporal para a Morlet (DOMINGUES et al., 2016). Dessa forma, é possível avaliar qual das características resolutivas se mostrará mais efetiva para esse problema de classificação. Também levou-se em consideração o fato de ambas terem apresentado um bom resultado quando utilizadas no treinamento de uma CNN aplicada destinada a uma diferente finalidade, a de estimativa de andamento musical (VIANA; JÚNIOR; FILHO,

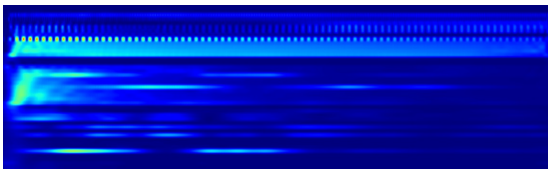
2023), onde escalogramas com 40 escalas dessas duas famílias de *wavelets* apresentaram os melhores resultados. As escalas utilizadas para o problema de classificação de efeitos de pedal foram ajustadas para cobrirem todo o espectro de frequências detectável pela audição humana, além de uma pequena faixa de frequência abaixo dos 20 Hz para tentar obter uma melhor distinção entre os efeitos baseados em LFOs. Como o processo de cálculo dos escalogramas se mostrou uma tarefa com um alto custo computacional, não foi possível gerar escalogramas com uma quantidade de escalas superiores a 40 para o presente trabalho.

Fig. 21 – Escalograma Morlet_40 obtido a partir do áudio ag_G+B_Overdrive40_27_.wav do GEC-GIM.



Fonte: Autoria própria

Fig. 22 – Escalograma Shannon_40 obtido a partir do áudio ag_G+B_Overdrive40_27_.wav do GEC-GIM.



Fonte: Autoria própria

3.3 Arquitetura da CNN

Para a tarefa de classificação de efeitos de pedal, a arquitetura da CNN utilizada foi a mesma proposta por (HINRICHS et al., 2022), que é baseada na *FxNet* proposta por (COMUNITÀ; STOWELL; REISS, 2020) para essa mesma tarefa.

Após a camada de entrada, a rede conta com dois conjuntos de camadas do tipo convolucional com *kernel* 3x3, normalização em lote e *max pooling* com *kernel* 2x2, com 32 filtros para o primeiro conjunto e 64 filtros no segundo conjunto.

Logo em seguida às camadas convolucionais, é introduzida uma camada de *dropout* que durante o processo de treinamento irá desativar aleatoriamente 30% das entradas, e é feito um achatamento para gerar dados unidimensionais que servirão de entrada para as camadas densas, que representam as camadas completamente conectadas de redes neurais artificiais multicamadas (FACELI et al., 2011). Serão utilizadas 3 camadas densas, onde as 2 camadas intermediárias possuirão 64 neurônios cada e utilizarão a ReLU como função de ativação e a camada de saída possuirá 11 neurônios (12 neurônios para o modelo treinado no IDMT-SMT), 1 para cada classe, e uma função de ativação *softmax*. Entre cada uma das camadas densas, serão feitas normalizações em lote e desativações *dropout* de 30% dos neurônios.

Para esta arquitetura, foram utilizadas imagens com as mesmas dimensões utilizadas para os espectrogramas de (HINRICHS et al., 2022), resultando em escalogramas e HCQTs

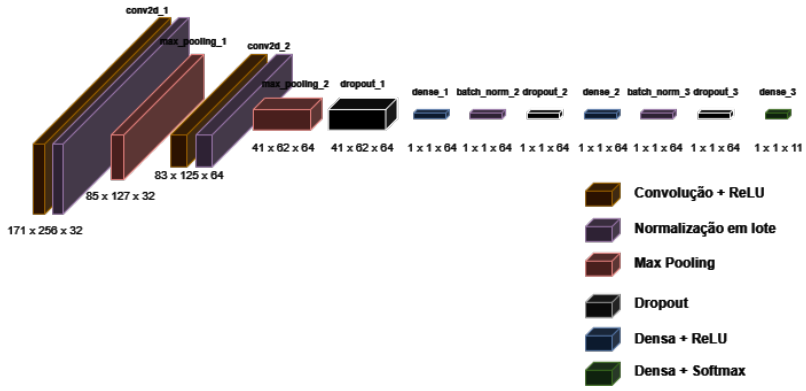


Fig. 23 – Arquitetura da CNN proposta por (HINRICHS et al., 2022) para a classificação de efeitos de áudio.

Fonte: Autoria própria

com dimensões de 256x173x3 e um total de até 10,437,324 parâmetros para a CNN. Uma diferença fundamental em relação à CNN apresentada por (HINRICHS et al., 2022) é a representação dos dados utilizados na camada de entrada; para este trabalho, foram utilizadas imagens no formato *Joint Photographic Experts Group* (JPEG) construídas a partir dos valores normalizados de amplitude das representações ao invés dos valores numéricos absolutos. A alteração no formato de representação dos dados de entrada foi motivada por limitações associadas ao carregamento dos dados para treinamento em memória, onde as representações numéricas ocupavam um maior espaço em *bytes* do que as imagens JPEG, já que essas últimas passam por processos de compressão (CORKE, 2017). Isso fez com que fosse necessário introduzir uma camada de normalização dos valores de entrada para que eles fossem convertidos de valores numéricos no padrão *Red, Green and Blue* (RGB) *8-bit* tradicional de imagens, de 0 a 255, (CORKE, 2017) para valores decimais normalizados entre 0 e 1, mais adequados para serem utilizados pela CNN no processo de treinamento. Essa camada representa a única diferença entre a CNN original e a utilizada nesse trabalho.

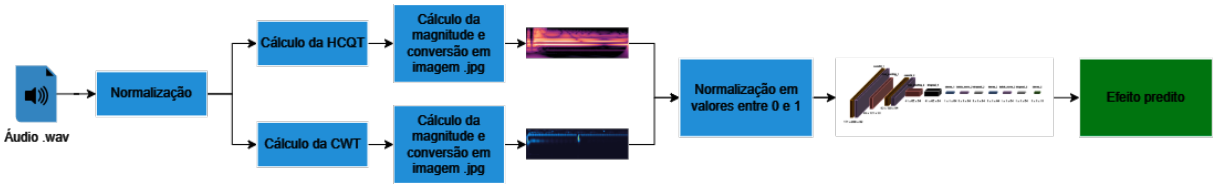


Fig. 24 – Fluxograma do caminho percorrido por um áudio para que a CNN possa ser treinada.

Fonte: Autoria própria

3.4 Processo de treinamento da rede

O treinamento da rede foi implementado através de códigos na linguagem de programação python e utilizando os *frameworks* Tensorflow e Keras (TENSORFLOW..., 2023) para a construção das camadas da CNN.

Assim como grande parte dos parâmetros do treinamento da rede, a função de custo utilizada foi a mesma utilizada por (HINRICHS et al., 2022), a entropia cruzada, bastante comum para problemas de classificação de múltiplas classes. O otimizador Adam (GOODFELLOW; BENGIO; COURVILLE, 2016) foi utilizado com taxa de aprendizado α de 0.00001, em contraste com os valores de 0.001 do artigo de referência. Experimentos foram feitos com diferentes valores de α, e esse valor precisou ser diminuído gradualmente até que se chegasse a esse número pois as primeiras tentativas de treinamento da rede estavam apresentando um elevado *overfitting* no conjunto de treinamento associado a uma oscilação no valor da função de custo para o conjunto de validação. (GOODFELLOW; BENGIO; COURVILLE, 2016) considera α como o mais importante dos hiperparâmetros de um modelo, embora o número de camadas e o número de neurônios também sejam hiperparâmetros importantes. Optou-se, com o propósito de não promover grandes alterações na arquitetura original da CNN de (HINRICHS et al., 2022), por buscar o melhor valor possível nesse trabalho para α através de experimentos com diferentes valores, fazendo com que outros hiperparâmetros fossem pouco explorados e se mantivessem inalterados.

A rede foi treinada por um máximo de 100 épocas com lotes contendo 64 amostras, com a utilização de uma estratégia de *early stopping* implementada através do Keras que monitorava a variação da função de custo para o conjunto de validação. Caso ela não apresentasse uma variação negativa de 0.5 após 30 épocas, o treinamento seria interrompido naquele momento pois não havia mais ganhos relevantes de acurácia no conjunto de validação e para evitar que ocorresse um *overfitting* no conjunto de treinamento, cuja função de custo continuava decrescendo continuamente. A medida avaliada ao longo do treinamento foi a acurácia, calculada como a razão entre as predições corretas e o total de predições feitas, e o treinamento foi dividido em 5 subconjuntos com a utilização da técnica *K-fold*, fornecendo 80% dos exemplos disponíveis de cada *dataset* para realizar o treinamento da CNN e outros 20% para avaliar o modelo, num processo conhecido como validação cruzada (HAYKIN, 2009). Essa divisão de *folds* foi a mesma realizada por (HINRICHS et al., 2022) e foi realizado um cálculo da média com intervalos de confiança de 95% para que esse trabalho pudesse obter resultados que pudessem ser comparados, Cada um dos diferentes modelos foi treinado com o GEC-GIM e o IDMT-SMT de forma separada, resultando em um total de 40 *folds* e processos de treinamento.

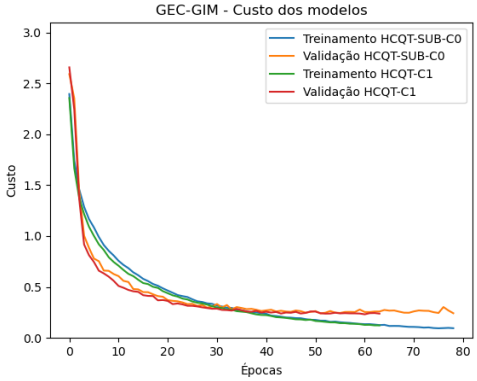
4 Resultados e Discussões

Os resultados obtidos após o treinamento dos 4 diferentes modelos de CNN serão apresentados nesse capítulo, onde além de uma avaliação da acurácia de cada um dos classificadores, os seus respectivos desempenhos para as diferentes classes do problema proposto serão analisados.

4.1 Resultados do treinamento

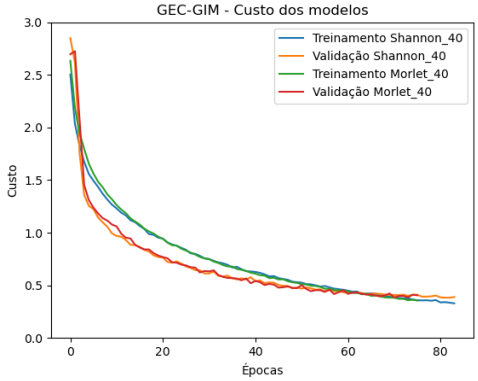
Os gráficos exibidos nas figuras 25, 26, 27 e 28 apresentam a curva do valor calculado pela função de custo de entropia cruzada ao longo das épocas para a HCQT e os escalogramas nos conjuntos de treinamento e validação, e nos gráficos das figuras 29, 30, 31 e 32 e a curva da evolução da acurácia para o mesmo cenário acima. Todos os gráficos utilizam o resultado dos treinamentos para o primeiro dos 5 *folds*.

Fig. 25 – Valor da função de custo da HCQT para o GEC-GIM



Fonte: Autoria própria

Fig. 26 – Valor da função de custo do escalograma para o GEC-GIM



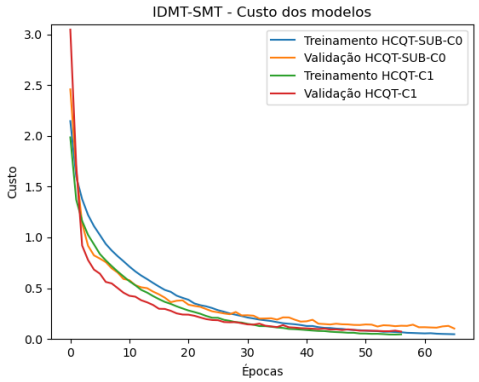
Fonte: Autoria própria

Ao longo do treinamento dos diferentes modelos, a estratégia de *early stopping* com uma paciência de 30 épocas fez com que o treinamento fosse interrompido entre as épocas 60 e 80, após os valores de custo no conjunto de validação não apresentarem uma melhoria, evitando que o modelo passasse por um *overfitting* no conjunto de dados de treinamento.

4.2 Acurácia dos modelos

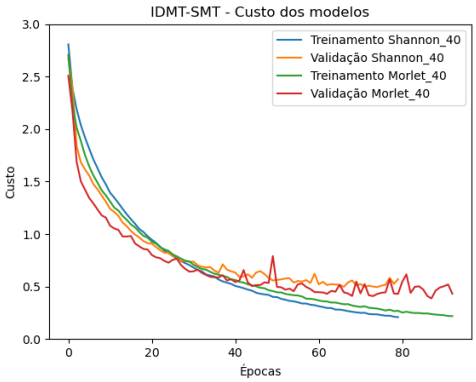
A tabela 2 lista as acurácias obtidas por (HINRICHS et al., 2022) para cada uma das diferentes representações tempo-frequência que servirão como *baseline* comparativo, assim como as 4 novas representações tempo-frequência obtidas nesse trabalho, com um

Fig. 27 – Valor da função de custo da HCQT para o IDMT-SMT



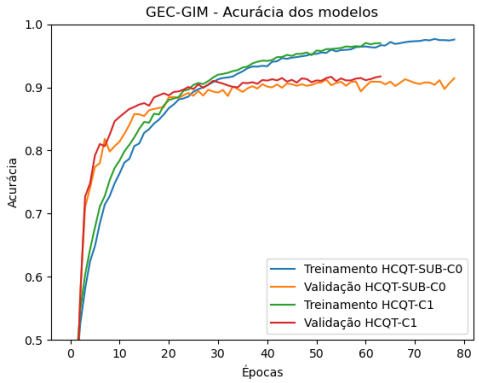
Fonte: Autoria própria

Fig. 28 – Valor da função de custo do escalograma para o IDMT-SMT



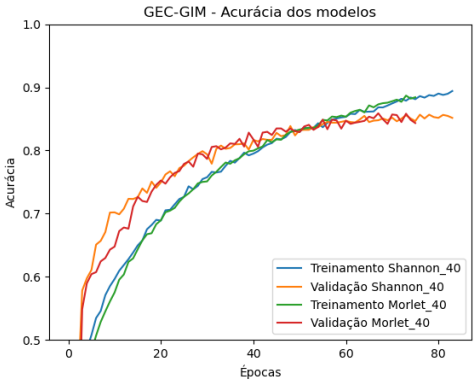
Fonte: Autoria própria

Fig. 29 – Valor da acurácia da HCQT para o GEC-GIM



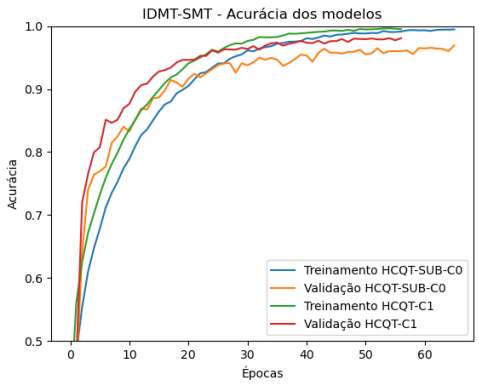
Fonte: Autoria própria

Fig. 30 – Valor da acurácia do escalograma para o GEC-GIM



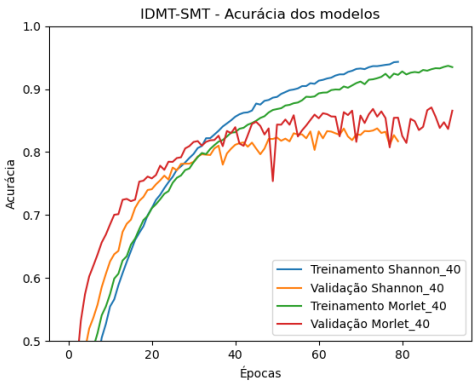
Fonte: Autoria própria

Fig. 31 – Valor da acurácia da HCQT para o IDMT-SMT



Fonte: Autoria própria

Fig. 32 – Valor da acurácia do escalograma para o IDMT-SMT



Fonte: Autoria própria

intervalo de confiança de 95%. Os valores totais de acurácia média e desvio padrão foram calculados utilizando as 5 versões do *K-fold* para cada modelo, onde o valor de acurácia foi calculado com base na razão entre as previsões corretas do modelo após o mesmo ser apresentado ao seu respectivo conjunto de dados de validação e os rótulos reais. Dessa forma, há a garantia de que o desempenho do modelo está sendo avaliado com base em dados que não foram utilizados durante o processo de treinamento da CNN, e serve como um indicativo de que o modelo não está se especializando apenas nos exemplos nos quais ele foi treinado (GOODFELLOW; BENGIO; COURVILLE, 2016).

Tab. 2 – Resultados da acurácia da CNN para cada uma das diferentes representações e *datasets*.

Representação tempo-frequência	**GEC-GIM**	**GEC-GIM (SD=FD)**	**IDMT-SMT**
Espectrograma (HINRICHS et al., 2022)	90.0 ± 0.57 %	95.8 ± 0.52 %	97.4 ± 0.7 %
MFCC	87.7 ± 0.52 %	93.4 ± 0.32 %	96.5 ± 0.13 %
GFCC	24.7 ± 27.3 %	28.7 ± 30.47 %	97.4 ± 0.3 %
Cromagrama	87.0 ± 0.64 %	92.9 ± 0.12 %	86.2 ± 0.2 %
HCQT-C1	89.5 ± 0.70 %	95.3 ± 0.66 %	95.1 ± 1.03 %
HCQT-SUB-C0	**90.4 ± 0.40 %**	**95.7 ± 0.47 %**	**95.9 ± 0.39 %**
Escalograma Morlet_40	83.1 ± 0.65 %	89.4 ± 0.59 %	80.9 ± 0.99 %
Escalograma Shannon_40	84.4 ± 0.57 %	90.3 ± 0.52 %	77.9 ± 0.82 %

Além do desempenho de cada modelo no *dataset* no qual ele foi treinado, a tabela 2 introduz um caso especial para o *dataset* GEC-GIM, em que as duas classes de efeitos de *delay*, o *Slapback* e o *Feedback*, são consideradas como uma classe só. Essa abordagem foi também utilizada por (HINRICHS et al., 2022) durante a avaliação dos resultados da rede pois foi possível notar que devido à variação de parâmetros durante a construção do GEC-GIM, alguns efeitos de *Feedback* contavam com apenas uma repetição muito rápida, o que fazia com que os dois efeitos fossem praticamente indistinguíveis até mesmo para especialistas.

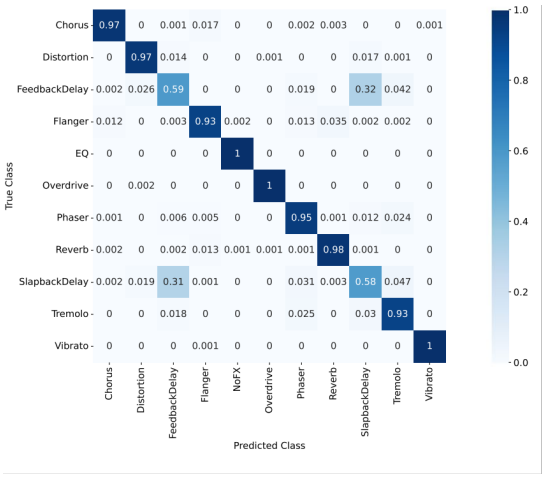
Os valores de acurácia mostram que as duas representações de HCQT obtiveram um bom desempenho em ambos os *datasets*, com valores de acurácia superiores ao limiar de 95%, que foi definido por (HINRICHS et al., 2022) como a acurácia humana esperada para um especialista em efeitos de áudio. Por outro lado, os escalogramas apresentaram resultados abaixo do esperado em ambos os *datasets*. Destaca-se o fato de que enquanto a maioria das outras representações obtiveram melhores valores de acurácia no IDMT-SMT quando comparados com o GEC-GIM, apenas os escalogramas e o cromagrama apresentaram um pior desempenho nesse *dataset*. Ao se considerar que, por sua definição, o cromagrama irá agrupar as frequências da escala cromática e condensá-las em um único coeficiente, a observação de um comportamento similar com os escalogramas pode ser um indicativo de que as escalas escolhidas e a quantidade das mesmas não estão representando suficientemente bem a totalidade das frequências presentes nos áudios, já que o IDMT-SMT conta com uma maior variedade de notas de guitarra sendo reproduzidas em comparação

com o GEC-GIM.

4.3 Matrizes de confusão

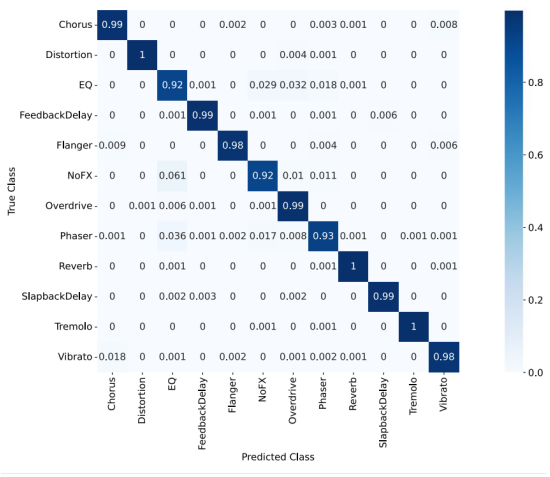
Nas figuras 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, estão apresentadas as matrizes de confusão de cada um dos modelos, assim como as matrizes de confusão para o melhor modelo obtido por (HINRICHS et al., 2022). As classes preditas pelo modelo e as classes verdadeiras estão dispostas como colunas e linhas da matriz, respectivamente; e cada célula da matriz contém, dentre o total de exemplos para cada uma das classes verdadeiras de uma linha, um percentual que indica quantas vezes aquela classe foi predita para as classes dispostas nas colunas. Na diagonal principal da matriz, onde há uma correspondência entre a classe verdadeira e a classe predita, é possível encontrar o valor da acurácia para cada uma das classes de efeitos.

Fig. 33 – Matriz de confusão para espectrograma GEC-GIM



Fonte: (HINRICHS et al., 2022)

Fig. 34 – Matriz de confusão para espectrograma IDMT-SMT

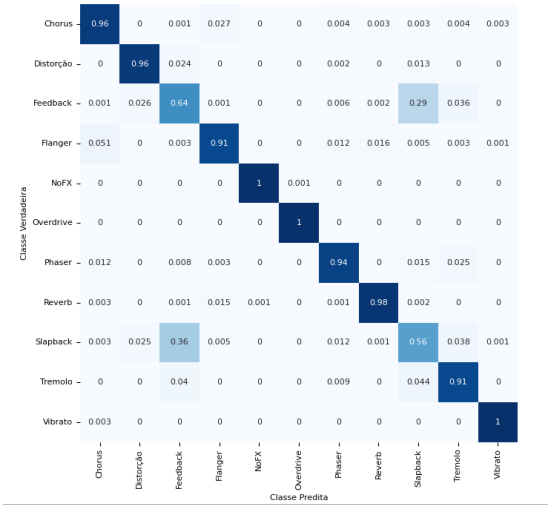


Fonte: (HINRICHS et al., 2022)

Uma característica marcante para todas as matrizes de confusão no conjunto de dados GEC-GIM é a queda na acurácia das duas classes de *delay*, o *Slapback* e o *Feedback*. Como uma consequência direta da maneira como os parâmetros foram definidos durante a construção do *dataset*, é possível observar que as predições incorretas para uma dessas duas classes se concentram majoritariamente na outra, e por isso há um grande aumento na acurácia do modelo quando essas duas classes passam a ser consideradas como uma só.

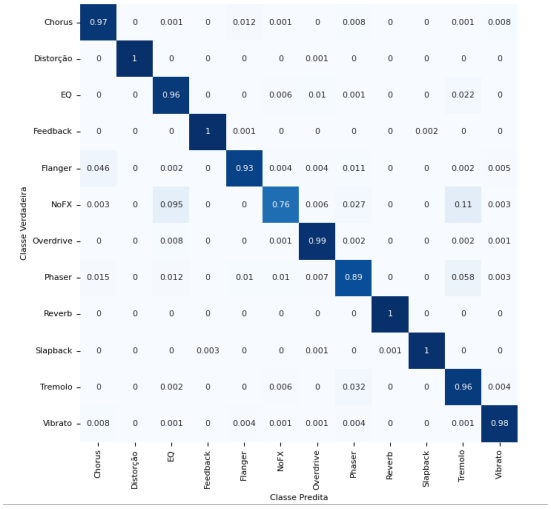
Apesar de a HCQT-C1 e a HCQT-SUB-C0 terem apresentado resultados próximos ou superiores aos obtidos por (HINRICHS et al., 2022) para o GEC-GIM, a acurácia no conjunto IDMT-SMT de nenhum dos modelos novos criados nesse trabalho conseguiu superar a obtida pelas representações de espectrograma, MFCC e GFCC. Analisando a matriz de confusão, é possível perceber que os modelos treinados com a HCQT tiveram pequenas

Fig. 35 – Matriz de confusão para HCQT-C1 GEC-GIM



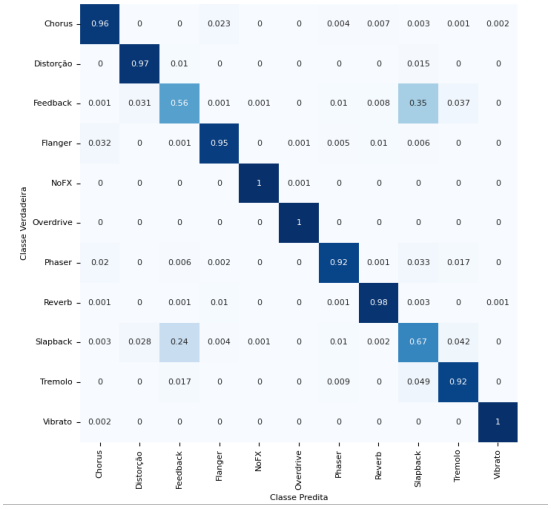
Fonte: Autoria própria

Fig. 36 – Matriz de confusão para HCQT-C1 IDMT-SMT



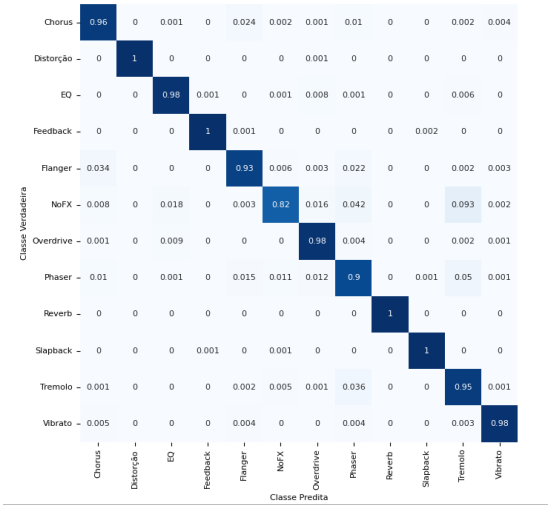
Fonte: Autoria própria

Fig. 37 – Matriz de confusão para HCQT-SUB-C0 GEC-GIM



Fonte: Autoria própria

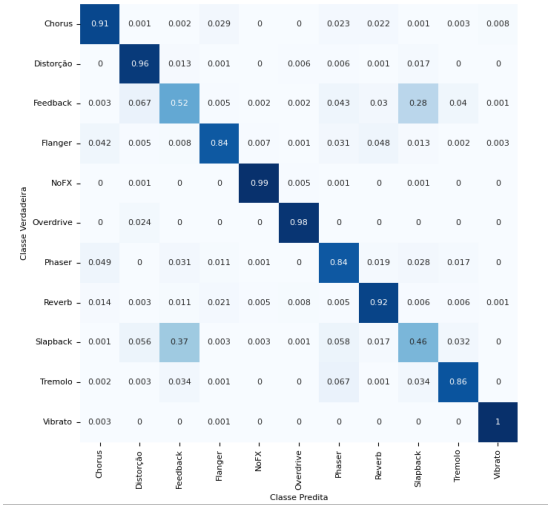
Fig. 38 – Matriz de confusão para HCQT-SUB-C0 IDMT-SMT



Fonte: Autoria própria

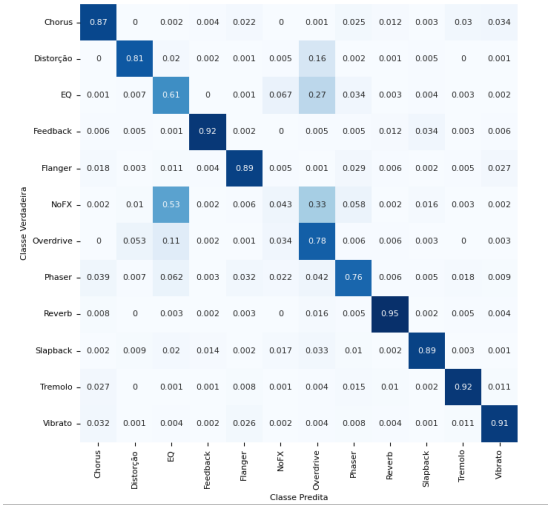
quedas na acurácia da classe *NoFX*, que consiste em áudios sem nenhum pós-processamento aplicado a eles, e quedas muito acentuadas para essa mesma classe nos modelos baseados em escalogramas, acompanhados de quedas moderadas nos efeitos de *Overdrive*, *Phaser* e Equalização. Uma possível causa pode ser a existência de um desbalanceamento de dados (FACELI et al., 2011) no conjunto IDMT-SMT, onde todas as classes apresentam um total de 1872 exemplos, mas as classes de Equalização e *NoFX* possuem uma quantidade de classes igual a 2/3 e 1/3 do total das outras classes, respectivamente. Dessa forma, o

Fig. 39 – Matriz de confusão para Shannon_40 GEC-GIM



Fonte: Autoria própria

Fig. 40 – Matriz de confusão para Shannon_40 IDMT-SMT

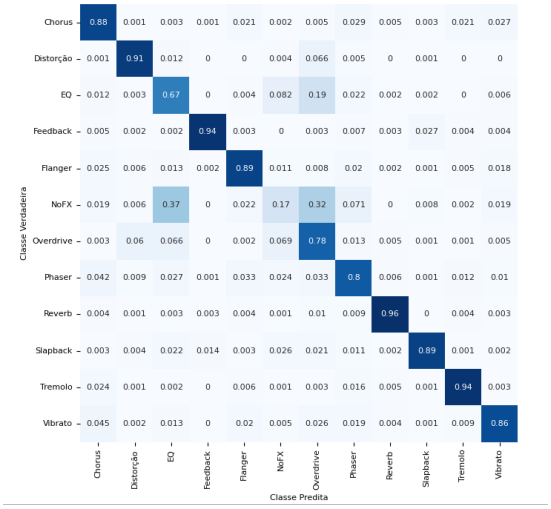


Fonte: Autoria própria

Fig. 41 – Matriz de confusão para Morlet_40 GEC-GIM

Fonte: Autoria própria

Fig. 42 – Matriz de confusão para Morlet_40 IDMT-SMT



Fonte: Autoria própria

modelo pode ter aprendido mais características a respeito das outras classes em detrimento dessas. Além disso, especificamente para os escalogramas, percebeu-se que os valores da função de custo oscilaram de forma excessiva nas últimas épocas do treinamento, o que significa que a função pode ter encontrado um valor mínimo local a partir desse momento e não conseguiu aprender o suficiente a respeito de todas as classes.

5 Conclusão

Com base nos resultados obtidos nesse trabalho, é possível afirmar que as duas representações em frequência utilizadas nesse trabalho, a HCQT e o escalograma *wavelet*, permitem o treinamento de classificadores de efeitos de guitarra. Tanto a HCQT-SUB-C0, quanto a HCQT-C1 apresentaram valores de acurácia muito próximos ao do modelo treinado com espectrogramas no *dataset* GEC-GIM por (HINRICHS et al., 2022) e conseguiram superar as outras representações. Embora a acurácia tenha sido um pouco menor no *dataset* IDMT-SMT, há a possibilidade de esse resultado ter sido causado pelo desbalanceamento da quantidade de exemplos para alguns dos efeitos, e portanto, é um problema que pode ser evitado com a geração manual de novos exemplos de áudios para esses efeitos, sem a necessidade de alteração na forma como as representações estão sendo geradas.

Isso não impede que se busque melhorias nas representações utilizadas; foi possível notar uma limitação no cálculo da HCQT pela biblioteca *librosa* (LIBROSA. . . , 2023), que não permitiu que fossem obtidas representações com uma quantidade total de *bins* de frequência superiores a 111, o que resultou em pequenas perdas de informação potencialmente relevantes para a rede neural, assim como a impossibilidade de calcular a transformada para frequências inferiores a $10Hz$. Novas implementações do algoritmo da transformada que não apresentem essa restrição podem eventualmente gerar representações em frequência mais completas para serem submetidas ao classificador.

Outros caminhos a serem explorados em trabalhos futuros também envolvem a geração de imagens com dimensões menores para que os modelos obtidos deixem de ter a grande quantidade de pesos e parâmetros treináveis que eles possuem, igual àquela dos modelos treinados com espectrograma; além disso, é necessário avaliar se há diferenças na *performance* das representações que foram geradas caso sejam utilizados formatos de imagens que não passem por um processo de compressão ou ao utilizar os valores absolutos dos coeficientes de magnitude ao invés de imagens, como feito por (HINRICHS et al., 2022).

As alternativas citadas acima não foram avaliadas nesse trabalho por limitações de *hardware* e também podem ser aplicadas no contexto da geração dos escalogramas *wavelet*. Antes da possibilidade de utilização dessa representação ser destacada, é necessário investigar se melhores resultados podem ser obtidos com a utilização de uma quantidade maior de escalas cobrindo diferentes faixas de frequência no áudio, assim como a experimentação de diferentes intervalos para as escalas.

E por fim, destaca-se a importância de avaliar o desempenho dos modelos quando

treinados em conjuntos de dados compostos de exemplos de músicas reais, que ainda precisam ser construídos para esse problema, para permitir a construção de ferramentas que possam ser utilizadas por músicos amadores ou produtores musicais que desejam extrair esse tipo de informação de canções já existentes.

Referências

COMUNITÀ, M.; STOWELL, D.; REISS, J. D. Guitar effects recognition and parameter estimation with convolutional neural networks. p. 4, 2020. Citado na página 52.

CORKE, P. *Robotics, Vision and Control - Fundamental Algorithgms in Matlab*. [S.l.]: Springer, 2017. v. 2. 295, 363 p. ISBN 9783319544120. Citado na página 53.

DIGILENT, Learn.Digilentinc Music Theory Basics. 2023. Acesso em 02/11/2023. Disponível em: <https://learn.digilentinc.com/Documents/400>. Citado na página 27.

DOMINGUES, M. et al. Explorando a transformada wavelet contínua. *Revista Brasileira de Ensino de Física, vol. 38*, p. 1–6, 11–13, 2016. Citado 3 vezes nas páginas 31, 32 e 51.

DUTILLEUX, P. Filters, delays, modulations and demodulations: A tutorial. *Proceedings of DFX98, Digital Audio Effects Workshop, Barcelona, Spain*, p. 4–11, 1998. Citado na página 38.

EICHAS, F.; FINK, M.; ZÖLZER, U. Empirical explorations of guitar players' attitudes towards their equipment and the role of distortion in rock music. *Proceedings of the 18th International Conference on Digital Audio Effects (DAFx-15)*, p. 1–3, 5, Nov 2015. Citado 7 vezes nas páginas 23, 33, 34, 37, 38, 39 e 40.

FACELI, K. et al. *Inteligência Artificial - Uma Abordagem de Aprendizado de Máquina*. [S.l.]: LTC, 2011. 34, 107-113, 117-119 p. ISBN 9788521618805. Citado 4 vezes nas páginas 41, 42, 52 e 59.

FERGUSON, M. et al. Automatic localization of casting defects with convolutional neural networks. p. 1726–1735, December 2017. Citado na página 45.

GERSEM, P. D.; MOOR, B. D.; MOONEN, M. Applications of the continuous wavelet transform in the processing of musical signals. *Proceedings of 13th International Conference on Digital Signal Processing*, p. 1–4, 1997. Citado na página 32.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. 168-171, 174, 178, 246-248, 274-275, 308-309, 330-352, 427-430 p. Citado 6 vezes nas páginas 41, 42, 43, 44, 54 e 57.

HAYKIN, S. *Neural Networks and Learning Machines*. [S.l.]: Pearson Education, 2009. v. 3. 21-22, 48, 123-128, 171, 201 p. ISBN 9780131471399. Citado 4 vezes nas páginas 41, 43, 44 e 54.

HERBST, J.-P. Empirical explorations of guitar players' attitudes towards their equipment and the role of distortion in rock music. *Columbia Academic Commons*, p. 1, March 2021. Citado na página 23.

HINRICHS, R. et al. Convolutional neural networks for the classification of guitar effects and extraction of the parameter settings of single and multi-guitar effects from instrument mixes. *EURASIP Journal on Audio, Speech, and Music Processing*, October 2022. Citado 14 vezes nas páginas 14, 23, 24, 47, 48, 49, 50, 52, 53, 54, 55, 57, 58 e 61.

JEEVAN, M. et al. Robust speaker verification using gfcc based i-vectors. *Proceedings of the International Conference on Signal, Networks, Computing, and Systems: ICSNCS 2016*, p. 86–88, October 2017. Citado 2 vezes nas páginas 29 e 30.

KARREN, C. Analog versus digital guitar pedals, shaping guitar tones and sparking debates. *California State University Monterey Bay, Capstone projects and Master's theses*, p. 4, May 2020. Citado na página 23.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. p. 84–90, June 2017. Citado na página 45.

KUMAR, N.; KUMAR, R. Wavelet transform-based multipitch estimation in polyphonic music. *Heliyon*, p. 1–2, 2020. Citado na página 32.

LERCH, A. *An Introduction to Audio Content Analysis*. [S.l.]: John Wiley and Sons, 2012. v. 1. 23-24, 51-53, 80-87, 101-102 p. ISBN 9781118266823. Citado 6 vezes nas páginas 25, 26, 28, 29, 30 e 33.

LIBROSA, página da web contendo documentação. 2023. Acesso em 28/09/2023. Disponível em: <https://librosa.org/>. Citado 2 vezes nas páginas 50 e 61.

MÜLLER, M. *Fundamentals of Music Processing Using Python and Jupyter Notebooks*. [S.l.]: Springer Nature Switzerland, 2021. v. 2. 123-128, 181 p. ISBN 9783030698072. Citado na página 28.

OPPENHEIM, A. V.; SCHAFER, R. W. *Processamento em tempo discreto de sinais*. [S.l.]: Pearson Education do Brasil Ltda, 2013. v. 3. 94-95, 478-480, 584-585 p. ISBN 9788581431024. Citado 4 vezes nas páginas 25, 26, 29 e 35.

PYWAVELETS, página da web contendo documentação. 2023. Acesso em 16/10/2023. Disponível em: <https://pywavelets.readthedocs.io/en/latest/index.html>. Citado na página 51.

REISS, J. D.; MCPHERSON, A. *Audio Effects: Theory, Implementation and Application*. CRC Press, 2014. 21-56, 125-127, 167-170, 253-267 p. ISBN 9781466560291. Disponível em: <https://books.google.com.br/books?id=mlHSBQAAQBAJ>. Citado 6 vezes nas páginas 35, 36, 37, 38, 39 e 40.

REVEILLAC, J.-M. *Musical Sound Effects: Analog and Digital Sound Processing*. [S.l.]: John Wiley and Sons, 2017. Citado na página 23.

SCHÖRKHUBER, C.; KLAPURI, A. Constant-q transform toolbox for music processing. *Proc. 7th Sound and Music Computing Conf.*, p. 1–2, January 2010. Citado na página 33.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. April 2015. Citado na página 45.

STEIN, M. Automatic detection of multiple, cascaded audio effects in guitar recordings. *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx-10)*, Sep 2010. Citado 2 vezes nas páginas 23 e 47.

SZEGEDY, C. et al. Going deeper with convolutions. September 2014. Citado na página 45.

TENSORFLOW, página da web contendo documentação. 2023. Acesso em 01/09/2023. Disponível em: <https://www.tensorflow.org/>. Citado na página 54.

VIANA, L. A. G.; JÚNIOR, A. C. L. F.; FILHO, E. F. de S. Estimativa de andamento musical através de escalogramas wavelet e redes neurais convolucionais. *XVI BRAZILIAN CONFERENCE ON COMPUTATIONAL INTELLIGENCE - CBIC 2023*, October 2023. Citado 2 vezes nas páginas 32 e 52.

WANG, M. et al. Hybrid constant-q transform based cnn ensemble for acoustic scene classification. *Proceedings of APSIPA Annual Summit and Conference 2019*, p. 1–3, November 2019. Citado 2 vezes nas páginas 33 e 50.

ZÖLZER, U. *DAFX - Digital Audio Effects*. John Wiley and Sons, 2002. 13-14, 69-71, 77 p. ISBN 0471490784. Disponível em: <https://books.google.com.br/books?id=mlHSBQAAQBAJ>. Citado 3 vezes nas páginas 35, 37 e 38.

APÊNDICE A – Códigos

Abaixo estão alguns dos códigos na linguagem python utilizados no processo de construção e treinamento da CNN.

A.1 Geração das HCQTs

```
def get_hcqt(dr, dataset):
    os.chdir(os.path.join(DATA_PATH, dataset))
    os.chdir(dr)
    for file_name in os.listdir(os.getcwd()): # percorre todos os arquivos
        ↪ do dataset
        if file_name.endswith(".wav"): # verifica se o arquivo possui o
            ↪ formato .wav
            y, sr = librosa.load(file_name, sr=None) # carrega o arquivo
                ↪ de musica
            y = librosa.util.normalize(y) # realiza normalizacao
            C = np.abs(librosa.hybrid_cqt(y, sr=sr, bins_per_octave=12,
                ↪ n_bins=111, hop_length=32, filter_scale=0.5, fmin=10)) #
                ↪  calculo da HCQT

            # plot da HCQT que sera salva em um arquivo .jpg
            DPI = 200
            fig = plt.figure(dpi=DPI, figsize=(6, 2))
            plt.axis('off')
            plt.tight_layout()
            ax = plt.gca()
            ax.xaxis.set_major_locator(matplotlib.ticker.NullLocator())
            ax.yaxis.set_major_locator(matplotlib.ticker.NullLocator())
            img = librosa.display.specshow(librosa.amplitude_to_db(C, ref=
                ↪ np.max), sr=sr)
            plt.savefig(file_name + '.jpg', pad_inches=0, bbox_inches='
                ↪ tight', transparent=True)
            plt.cla()
            plt.clf()
            plt.close(fig)
```

A.2 Geração dos escalogramas

```
def get_scalogram(dr, dataset):
    os.chdir(os.path.join(DATA_PATH, dataset))
    os.chdir(dr)
    # definicao das escalas
    s = np.array([1.7, 57, 112, 167, 222, 277, 332, 387, 442, 497, 552,
        ↪ 607, 662, 717, 772, 827, 882, 937, 992, 1047, 1102, 1157, 1212,
        ↪  1267, 1322, 1377, 1432, 1487, 1542, 1597, 1652, 1707, 1762,
        ↪ 1817, 1872, 1927, 1982,2037, 2092, 2147])
    # wavelet utilizada
    wav = 'shan'
    for file_name in os.listdir(os.getcwd()): # percorre todos os arquivos
        ↪  do dataset
        if file_name.endswith(".wav"): # verifica se o arquivo possui o
            ↪ formato .wav
            y, sr = librosa.load(file_name, sr=None) # carrega o arquivo
                ↪ de musica
            y = librosa.util.normalize(y) # realiza normalizacao
            W, _ = pywt.cwt(y, s, wav, method='fft') # calcula os
                ↪ coeficientes wavelets para cada uma das escalas s
            W = np.abs(W)**2 # calculo da potencia do escalograma

            # plot do escalograma que sera salvo em um arquivo jpg
            DPI = 200
            fig = plt.figure(dpi=DPI, figsize=(6, 2))
            plt.axis('off')
            plt.tight_layout()
            ax = plt.gca()
            ax.xaxis.set_major_locator(matplotlib.ticker.NullLocator())
            ax.yaxis.set_major_locator(matplotlib.ticker.NullLocator())
            plt.imshow(W, extent=[0, len(y), s[-1], s[0]], cmap='jet',
                ↪ aspect='auto')
            plt.savefig(file_name + '.jpg', pad_inches=0, bbox_inches='
                ↪ tight', transparent=True)
            plt.cla()
            plt.clf()
            plt.close(fig)
```

A.3 Construção do modelo

```
def create_model(input_dim = (173, 256, 3), output_dim = 11, kernel_size
    ↪ = (3, 3), n_nodes = 64, n_filters = 32):
    tensorflow.keras.backend.clear_session()
    model = []
    model = models.Sequential()
    # primeira camada convolucional
    model.add(layers.Conv2D(n_filters, kernel_size=kernel_size, activation
        ↪ ='relu', input_shape=input_dim))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    # segunda camada convolucional
    model.add(layers.Conv2D(n_filters, kernel_size=kernel_size, activation
        ↪ ='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(layers.Dropout(0.3))
    model.add(layers.Flatten())
    # camadas densas intermediarias
    model.add(layers.Dense(n_nodes, activation='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.Dropout(0.3))

    model.add(layers.Dense(n_nodes, activation='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.Dropout(0.3))

    # camada de saida
    model.add(layers.Dense(output_dim, activation='softmax'))
    model.compile(loss='categorical_crossentropy',
                  optimizer=optimizers.Adam(learning_rate=0.00001), # taxa
                      ↪ de aprendizadode 0.00001
                  metrics=[metrics.CategoricalAccuracy()]) # acuracia
                      ↪ categorica
    return model
```

A.4 Treinamento do modelo

```
def train_model(model, train_data, train_labels, test_data, test_labels):

    # tamanho do batch size
    BS = 64

    #definicao da estrategia de early stopping
    callback = tf.keras.callbacks.EarlyStopping(
    monitor="val_loss",
    min_delta=0.1,
    patience=30,
    verbose=1,
    mode="min",
    baseline=None,
    restore_best_weights=True,
    )

    # treinamento da rede por 100 epochs com early stopping sendo
       ↪ registrado como callback
    history = model.fit(train_data, epochs=100, batch_size=BS, verbose=2,
       ↪ validation_data = test_data, callbacks=[callback])
    return history.history
```