UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
SISTEMAS DE INFORMAÇÃO

MATHEUS LAUREANO

RECONHECIMENTO E ANÁLISE MUSICAL DE INSTRUMENTOS MELÓDICOS COM REDES NEURAIS

Florianópolis, SC
2022

MATHEUS LAUREANO

RECONHECIMENTO E ANÁLISE MUSICAL DE INSTRUMENTOS MELÓDICOS COM REDES NEURAIS

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação, da UNIVERSIDADE FEDERAL DE SANTA CATARINA, como requisito parcial para a Obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Elder Rizzon Santos

Florianópolis, SC
2022

MATHEUS LAUREANO

RECONHECIMENTO E ANÁLISE MUSICAL DE INSTRUMENTOS MELÓDICOS COM REDES NEURAIS

Trabalho de Conclusão de Curso apresentado ao curso de Sistemas de Informação, da UNIVERSIDADE FEDERAL DE SANTA CATARINA, como requisito parcial para a Obtenção do grau de Bacharel em Sistemas de Informação.

Florianópolis, 08 de Julho de 2022

BANCA EXAMINADORA

_____

Prof. Dr. Guilherme Alex Derenievicz
Universidade Federal do Paraná

_____

Prof. Dr. Rafael de Santiago
Universidade Federal de Santa Catarina

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
Hz	Hertz
MFCC	Mel Frequency Cepstral Coefficient
OBNN	Rede Neural Baseada em Otimização
OMR	Reconhecimento Óptico de Música
ReLu	Rectified Linear Unit
RN	Redes Neurais
RNA	Redes Neurais Artificiais
RP	Residual Phase

RESUMO

Esse projeto apresenta como a tecnologia de redes neurais pode auxiliar no processo de reconhecimento musical, mostra os detalhes de uma implementação de rede neural desenvolvida com auxílio de diversas ferramentas como a biblioteca librosa, tensorflow e scikit-learn. Também foi utilizado o método de extração Mel Frequency Cepstral Coefficients (MFCC) para extrair os coeficientes de espectro de áudio criando um vetor de características. Utilizando um conjunto de dados desenvolvido pelo autor, o projeto obteve 82% de acerto no reconhecimento das notas musicais treinadas. Esse projeto também apresenta uma comparação com o método tradicional de reconhecimento de notas, o qual trouxe uma taxa de acerto inferior. Apesar de pouca diferença, a rede neural se demonstrou eficiente para realizar esse conhecimento.

Palavras-chave: Análise Musical. Rede Neural. MFCC. Notas Musicais. Reconhecimento Musical. Aprendizagem de Máquina. Extração Características das Notas.

1 **INTRODUÇÃO**

A música é capaz de comunicar emoções básicas que são reconhecidas sem esforço em adultos, independentemente da formação musical e universalmente em diferentes culturas (HSIEH et al., 2012). O sistema auditivo humano nos dá a capacidade extraordinária de identificar instrumentos sendo tocados (afinados e desafinados) a partir de uma peça musical e também ouvir o ritmo/melodia do instrumento individual sendo tocado. Esta tarefa parece "automática" para nós, mas provou ser muito difícil de replicar em sistemas computacionais .

Uma maneira para reproduzir essa capacidade analítica em sistemas computacionais é por meio das Redes Neurais Artificiais (RNAs) que conforme Jain, Mao e Mohiuddin (1996), as RNAs são inspiradas por redes neurais biológicas e são sistemas de computação massivamente paralelos que consistem em um número extremamente grande de processadores simples com muitas interconexões. Os modelos de RNA tentam usar alguns princípios “organizacionais” que se acredita serem usados no cérebro humano. A identificação do ritmo/melodia de uma peça musical é parte essencial para o projeto e se espera que utilizando modelos de RNA consiga completar essa tarefa.

Alguns trabalhos apresentam o uso de RNAs para o reconhecimento óptico de música (OMR) que é o reconhecimento de símbolos musicais a partir de imagens como é o caso de (HOMENDA; LUCKNER, 2004) cujo estudo identificou que o método perceptron com multicamadas traz resultados relativamentes melhores se comparado com redes neurais probabilísticas e funções com base radial, o trabalho de (GEORGE, 2003) focou em reconhecer os padrões de notas manuscritas e o de (MIYAO; NAKANO, 1995) em identificar partes específicas (cabeça e haste) de notas musicais impressas. Outros trabalhos focam no reconhecimento a partir de áudios como é o caso de (SLIZOVSKAIA; GÓMEZ; HARO, 2017) cuja análise envolve vídeos gerados pelo usuário e (PONS et al., 2017) que apresenta análises de timbre usando redes neurais convolucionais, em seu trabalho, extraiu o tom e separou a voz de um acompanhamento musical usando algoritmo tandem, (DE JESUS GUTERRO-TURRUBIATES et al., 2014) e (TAMBOLI; KOKATE, 2019) fizeram uma classificação de notas utilizando redes neurais e identificaram que mesmo com um dataset contendo ruídos, amplitudes variadas e tons diferentes a rede neural conseguiu ser robusta o suficiente para distinguir as notas apesar desses problemas.

Existem diversos tipos de instrumentos, cada um possui suas características e dificuldades para a execução das notas. Para um músico evoluir em um

instrumento é necessário prática, e até meados de 1850 com a produção de partituras impressas em grande quantidade, apesar de a tradição mestre-discípulo ser mantida, os exercícios, mais técnicos que melódicos, passam a ser estudados a partir dos métodos musicais impressos. Estes possuem ênfase no desenvolvimento de habilidades técnicas (MCPHERSON; GABRIELSSON, 2002). Os métodos musicais geralmente são específicos para um grupo de instrumentos com características similares, os músicos utilizam esses métodos para aprimorar certas habilidades técnicas tais como velocidade, dinâmica e extensão musical.

Conforme Hallam (2006), a maior parte das aulas de instrumento é dirigida pelo professor, que determina o programa a ser seguido e seleciona o repertório que será executado bem como a maneira como ele deve ser executado. Nessas aulas, o estudo técnico muitas vezes é priorizado em detrimento das questões musicais e os questionamentos representam uma pequena proporção do tempo. Entretanto, parte essencial da aprendizagem do músico é a prática individual onde o aluno pode encontrar dificuldades e a necessidade de auxílio na execução.

A proposta deste trabalho é desenvolver um protótipo de análise musical para auxiliar o iniciante a atingir a maneira de execução correta apresentada no repertório por meio de feedbacks na prática individual utilizando RNAs que contribuam na identificação dos padrões musicais comparando com o áudio identificado e retornando informações dos resultados da comparação, para isso é necessário analisar a fundamentação teórica e trabalhos correlatos que se relacionem computação musical, definir experimentos com modelos de redes neurais para o reconhecimento e análise de notas musicais, a partir de uma entrada de áudio, que produzam um *feedback* com erros e acertos.

2 **OBJETIVOS**

2.1 OBJETIVO GERAL

Desenvolver um mecanismo para análise musical considerando os aspectos de afinação, tempo e dinâmica em um ou mais instrumentos melódicos para auxiliar o músico iniciante a executar o repertório na forma que o mesmo exige.

2.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são:

1. Analisar a fundamentação teórica e o estado da arte relacionada a análise e reconhecimento computacional musical.
2. Estabelecer quais modelos de RN são mais adequados para o reconhecimento e análise de notas musicais a partir de um dado áudio
3. Definir um ou mais experimentos (conjunto de dados, critérios de avaliação, etc.) para analisar os modelos quanto à sua capacidade de reconhecimento musical.
4. Analisar os resultados dos modelos quanto à análise do tempo, dinâmica e afinação.

2.3 PREMISSAS E RESTRIÇÕES

O foco do presente trabalho se limita aos instrumentos melódicos de sopro principalmente os instrumentos que fazem parte do naipe dos metais. O projeto tem como premissa o acesso a métodos musicais e acesso de execuções gravadas dos exercícios presente no método.

3 **FUNDAMENTAÇÃO TEÓRICA**

3.1 REDES NEURAIS ARTIFICIAIS

Um ser humano possui capacidade de realizar diversas ações, como conversar, ouvir músicas, realizar cálculos matemáticos, reconhecer rostos, entre outras. Nesse contexto de RNAs (FACELI et al., 2011), a capacidade de realizar tais ações está vinculada a múltiplas camadas de neurônios que interagem entre si.

As redes neurais artificiais são sistemas de processamento de dados e geração de saída que replica o sistema neural para desvendar relações não lineares em um grande conjunto de dados. Os dados podem ser sonoros, textuais e visuais (SHAH, 2020).

> Existem várias aplicações bem-sucedidas de técnicas de aprendizado de máquinas na solução de problemas reais, entre as quais podem ser citadas: Reconhecimento de palavras faladas, predição de taxas de cura de pacientes com diferentes doenças, detecção do uso fraudulento de cartões de crédito, condução de automóveis de forma autônoma em rodovias, ferramentas que jogam gamão e xadrez de forma semelhante a campeões, diagnóstico de câncer por meio da análise de dados de expressão gênica. (FACELI et al., 2011).

3.1.1 **Aprendizagem Não Supervisionada**

Essa abordagem nos permite abordar problemas com pouca ou nenhuma ideia do que os resultados devem ser, nessa abordagem não há nenhum "supervisor externo" que traga um *feedback* com base nos resultados da previsão.

Uma tarefa descritiva de agrupamento de dados, por exemplo, tem por meta encontrar grupos de objetos semelhantes no conjunto de dados. Outra tarefa descritiva é encontrar regras de associação que relacionam um grupo de atributos a outro grupo de atributos. (FACELI et al., 2011).

3.1.2 **Aprendizagem Supervisionada**

O termo supervisionado vem da simulação de presença de um "supervisor externo", que conhece a saída (rótulo) desejada para cada exemplo (conjunto de valores para os atributos de entrada). Com isso, o supervisor externo pode avaliar a capacidade da hipótese induzida de predizer o valor de saída para novos exemplos. (FACELI et al., 2011).

Para esse projeto foi utilizado a aprendizagem supervisionada, visto que o

objetivo é a classificação dos rótulos de cada entrada (notas), e esses rótulos são conhecidos.

3.1.3 **Funções de Ativação**

As funções de ativação realizam a conversão do sinal de entrada para um sinal de saída, definindo as conexões entre os neurônios artificiais. O objetivo das funções é adicionar uma não linearidade ao modelo matemático.

> As funções de ativação são um elemento extremamente importante das redes neurais artificiais. Elas basicamente decidem se um neurônio deve ser ativado ou não. Ou seja, se a informação que o neurônio está recebendo é relevante para a informação fornecida ou deve ser ignorada. (DATA SCIENCE ACADEMY, 2022).

Existem diversas funções, no projeto foi utilizado a Unidade Linear Retificada. Essa função limita inferiormente a saída para 0. A principal vantagem desta função é o baixo custo computacional em relação a outras funções (MAAS; HANNUN; NG, 2013).

3.2 ANÁLISE MUSICAL

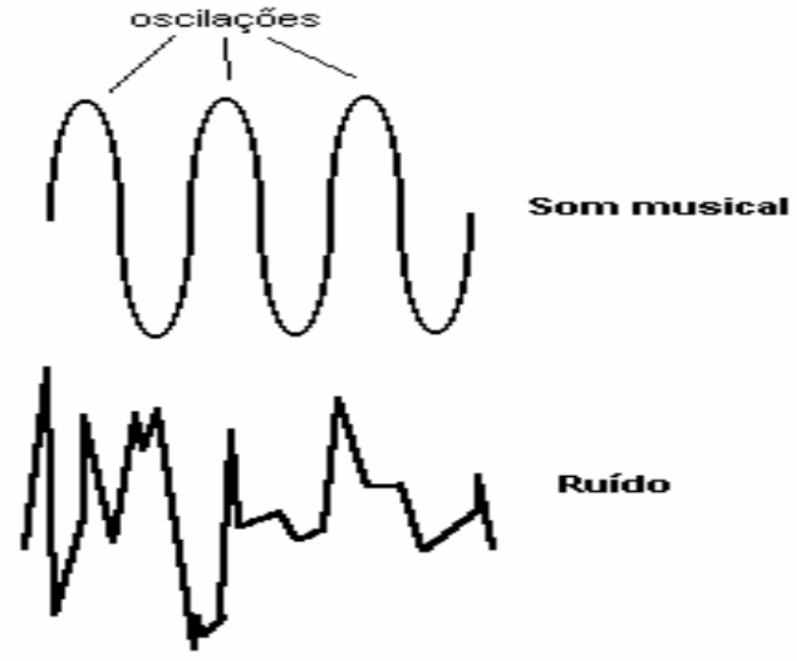
Essa seção apresenta definições e conceitos de análise musical utilizados no projeto como um todo.

3.2.1 **Som**

O som é a sensação produzida no ouvido pelas vibrações de corpos elásticos. Uma vibração põe em movimento o ar na forma de ondas sonoras que se propagam em todas as direções simultaneamente. Estas atingem a membrana do tímpano fazendo-a vibrar. Transformadas em impulsos nervosos, as vibrações são transmitidas ao cérebro que as identifica como tipos diferentes de sons (MED, 1996, p. 11).

Med comenta também que existem duas vibrações a regular e a irregular. A vibração regular produz sons de altura definida, chamados sons musicais ou notas musicais, como o som do piano, do violino, etc. A vibração irregular produz sons de altura indefinida chamados de barulhos. Por exemplo, som de avião, de automóvel, de uma explosão, etc. A Figura 1 mostra a representação das vibrações regulares e irregulares.

Figura 1 — Representação Gráfica das Ondas Sonoras



Fonte: Júnior, Faria e Yamanaka (2007, p. 11)

Med (1996, p. 11) explica que as características principais do som são altura, duração, intensidade e timbre.

- Altura - determinada pela frequência das vibrações, isto é, da sua velocidade. Quanto maior for a velocidade da vibração, mais agudo será o som.
- Duração - extensão de um som; é determinada pelo tempo de emissão das vibrações.
- Intensidade - amplitude das vibrações; é determinada pela força ou pelo volume do agente que as produz. É o grau de volume sonoro.
- Timbre - combinação de vibrações determinadas pela espécie do agente que as produz. O timbre é a "cor" do som de cada instrumento ou voz, derivado da intensidade dos sons harmônicos que acompanham os sons principais.

Todo e qualquer som musical tem, simultaneamente, as quatro propriedades (MED, 1996, p. 12). E com base nesses parâmetros é possível realizar o reconhecimento dos instrumentos musicais e suas notas.

Na música ocidental existem dois tipos de sistemas musicais, o natural e o temperado e ambos definem o semitom como o menor intervalo adotado entre duas notas.

> O sistema natural é fundamentado em cálculos acústicos e define com precisão o número de vibrações para cada nota e as relações entre elas e apesar de ser mais afinado é bastante complexo. O sistema temperado representa o abandono da perfeição da afinação absoluta no sistema natural em favor do uso do sistema cromático, a escala temperada consiste na divisão da oitava em doze semitons (Dó, Dó#/Réb, Ré, Ré#/Réb, Mi, Fá, Fá#/Solb, Sol, Sol#/Láb, Lá, Lá#/Sib, Si.) (MED, 1996, p. 30-31).

Para esse trabalho vai ser utilizado o sistema temperado devido a sua praticidade.

3.2.2 **Afinação**

A afinação é uma questão central para a prática musical de vários instrumentistas e cantores. É com base nela que todos os instrumentos são padronizados, Freire (2016) explica que a estrutura de um sistema de afinação depende da descrição exata das frequências de cada uma das notas de uma escala e de quais são os critérios e os parâmetros para a construção desta escala.

> Na área de performance musical, o nome de Hertz encontra-se fortemente associado à escolha de uma nota de referência para a afinação de um instrumento ou de um conjunto musical. A referência A = 440 Hz indica que a nota Lá3, com frequência de 440 ciclos por segundo, será utilizada como parâmetro para a afinação de um instrumento. (FREIRE, 2016).

A escolha de uma afinação é essencial para padronizar as notas identificadas, para esse projeto vai ser utilizado como referência A = 440 Hz.

3.2.3 **Instrumentos Transpostos**

Instrumento transposto é qualquer instrumento musical que, por qualquer razão, tem suas notas anotadas na partitura em altura diferente daquela que realmente soa.

O trompete afinado em Bb por exemplo, se você tocar uma nota C no trompete vai soar um Bb, por isso é chamado de Trompete em Bb. Qualquer nota tocada pelo trompete vai soar uma segunda menor do que a que está escrita, um A soa como um G (DUNNETT).

Neste projeto as frequências e notas utilizadas são as escritas para o trompete, um exemplo: Quando o trompete toca a nota C, ele está executando uma sonoridade de aproximadamente 233 Hz, que soa como a nota Bb. No projeto chamamos a nota Bb como nota real e C a nota do trompete.

3.2.4 **Representação digital do som**

Segundo (MILETTO et al., 2004) a digitalização é o processo de representar numericamente grandezas analógicas (áudio, vídeo e imagem) em computadores, que são máquinas que trabalham na notação binária.

O processo de digitalização do som, é realizado em quatro etapas:

- Filtragem: Limita uma faixa de frequência, onde valores acima da frequência máxima são descartados.
- Amostragem: Faz com que o sinal analógico passe por um circuito amostrador, e deste circuito distribua taxas de amostragens do som a ser digitalizado
- Quantização: Arredonda os valores de nível de amplitude para o valor mais próximo quando ficam entre dois valores digitais, os pulsos gerados na amostragem passam a ser convertidos em números.
- Gravação dos dados obtidos em arquivo.

3.2.5 **Mel Frequency Cepstral Coefficients (MFCC)**

Segundo (LALITHA et al., 2015, p. 30) Mel Frequency Cepstrum (MFC) é a representação linear do cosseno transformado de um espectro de potência logarítmica de curto prazo do sinal de voz em uma escala de frequência Mel não linear. E (SOUZA, 2020) explica que a extração do MFCC é um método para extrair os coeficientes de um espectro de áudio, permitindo criar um vetor de características que fornecem uma melhor representação do sinal, já que exploram a frequência que os humanos ouvem.

Essa etapa de extração para gerar vetores de características é fundamental para que se obtenha sucesso nas outras etapas.

4 **TRABALHOS RELACIONADOS**

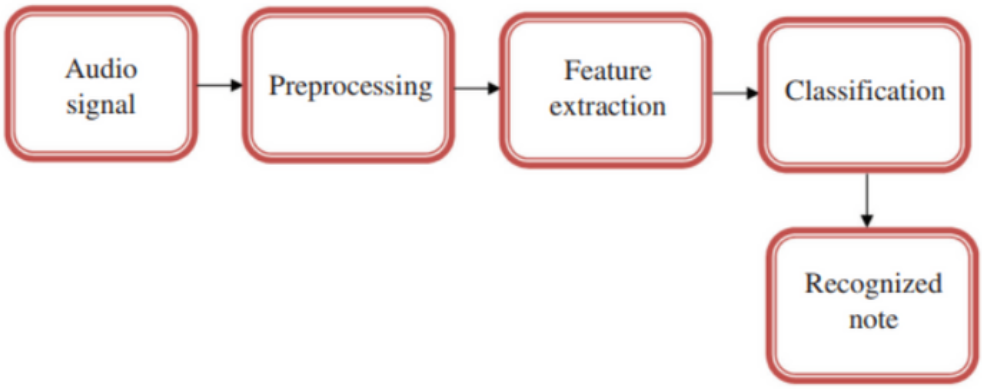
Os trabalhos relacionados foram selecionados com o objetivo de levantar métricas e métodos similares aos que foram utilizados nesse projeto, tais como, a utilização de pré-processamento com MFCC, análise musical similar, reconhecimento musical e inteligência artificial na educação musical.

4.1 AN EFFECTIVE OPTIMIZATION-BASED NEURAL NETWORK FOR MUSICAL NOTE RECOGNITION (TAMBOLI; KOKATE, 2019).

O trabalho teve como objetivo desenvolver um método para reconhecimento de notas musicais baseado na estrutura de classificação usando uma *optimization-based neural network* (OBNN). Foi utilizado um dataset um banco de dados de notas musicais que contou com vários gêneros de músicas como blues, eletrônica, folk, country, jazz, pop e rock.

A amostra de áudio é experimentado na etapa de pré-processamento e, em seguida, o ruído na amostra é removido no filtro mediado. Na etapa de extração de características, são separados dois tipos de características, *residual phase* (RP) e *mel frequency cepstral coefficient* (MFCC), são abstraídos do sinal filtrado. Por fim, a classificação é realizada nas características extraídas pelo algoritmo OBNN para reconhecimento de notas musicais.

Diagrama 1 — Diagrama de bloco do método proposto por Tamboli e Kokate



Fonte: Tamboli e Kokate (2019, p. 3)

Foram utilizadas quatro métricas de avaliação de estrutura para classificação binária: precisão, recall / sensibilidade e especificidade.

O estudo conseguiu classificar as notas musicais com acurácia, problemas como ruídos, amplitude variável e tom estavam presentes nas amostras de áudio, mesmo assim a rede neural foi capaz de distinguir esses problemas. O sistema

consegue reconhecer as notas fundamentais do acorde, demonstrando a validade da proposta abordada e a robustez da rede neural a sinais ruidosos.

4.2 PITCH ESTIMATION FOR MUSICAL NOTE RECOGNITION USING ARTIFICAL NEURAL NETWORKS (DE JESUS GUTERRO-TURRUBIATES ET AL., 2014).

Esse trabalho propõe um sistema de reconhecimento das notas musicais baseado nas modificações harmônicas e RNA. Foi aplicado uma redução na amostragem para converter o sinal da taxa de 44.100 Hz para 2.100 Hz. FFT (Fast Fourier Transform) foi utilizado para obter o espectro do sinal, o algoritmo HPS (Harmonic Product Spectrum) é implementado para aprimorar a amplitude de frequência fundamental. Em seguida, um método de redução de dimensionalidade baseado em variações é usado para extrair informações relevantes do sinal de entrada. No trabalho, os sinais de áudio foram extraídos de um banco de dados próprio que foi construído usando uma guitarra elétrica como fonte de áudio.

A classificação é realizada por uma RN feed-forward ou MLP (Multi-Layer Perceptron). Os resultados experimentais apresentam classificação precisa com pouco processamento do sinal de entrada. Além da abordagem proposta, apresenta robustez suficiente para classificar notas musicais provenientes de diferentes instrumentos musicais.

Neste estudo, as notas musicais foram classificadas com precisão. Considerando que os sinais de áudio da vida real funcionam em ambientes estruturados e não estruturados, o problema envolve fatores que afetam seriamente o desempenho dos sistemas inteligentes. Problemas como ruído, amplitude variável e cor do tom foram discriminadas pela rede neural.

O algoritmo HPS no estágio de pré-processamento apresentou resultados eficientes devido às suas habilidades de extração de recursos. Além disso, esse algoritmo possibilitou a classificação de notas provenientes de outros instrumentos reais, como piano, saxofone, violino, violão, mesmo quando a RNA foi treinada apenas com arquivos de áudio gravados em uma guitarra elétrica.

O MLP foi usado para classificação e foi conformado com 20 neurônios ocultos na primeira camada e 10 unidades ocultas na segunda. O sistema final alcançou uma precisão de reconhecimento de 97:5%.

Experimentos adicionais foram conduzidos usando acordes em vez de notas simples. O sistema consegue reconhecer a nota fundamental do acorde, demonstrando validade da abordagem proposta e a robustez da rede neural e sinais ruidosos.

4.3 INTELIGÊNCIA ARTIFICIAL NA EDUCAÇÃO MUSICAL (BARROS; LUZ; BAIA, 2019)

O trabalho tem como objetivo geral auxiliar os discentes de música através da indicação de sua performance, os objetivos específicos do artigo são apresentar como a tecnologia de redes neurais artificiais podem auxiliar o processo de ensino-aprendizagem, como realizar o treinamento de uma rede neural para reconhecer notas musicais e apresenta os resultados alcançados.

O artigo tem como abordagem inicial a descrição do som, e apresenta seus parâmetros como, intensidade, altura, timbre e envelope. Logo depois, demonstra como é a representação digital do som e as quatro etapas do processo de digitalização, que são: Amostragem, filtragem, quantização e gravação de dados.

Conforme descreve nos seus objetivos específicos, o artigo apresenta qual foi o método de desenvolvimento do trabalho, apresentando um fluxograma com os passos para realizar o treinamento de uma rede neural para reconhecer notas musicais. Foi criado uma interface gráfica para apresentar os resultados dos cálculos em forma de porcentagem. O dataset foi concebido a partir do som das notas do violão em um computador e gravado com o aplicativo.

Como resultado a rede neural artificial concebida obteve um desempenho médio de 94,44% de acerto no reconhecimento das notas musicais, reconhecendo 10 notas das 12 notas treinadas.

4.4 MUSICAL INSTRUMENT RECOGNITION IN USER-GENERATED VIDEOS USING A MULTIMODAL CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE (SLIZOVSKAIA; GÓMEZ; HARO, 2017)

O trabalho apresenta um método de reconhecimento de instrumentos musicais em vídeos gerados por usuários, aborda um cenário do mundo real com vários desafios de pesquisa, ou seja, a análise de vídeos gerados por usuários que variam em termos de condições e qualidade de gravação e podem conter vários instrumentos que soam simultaneamente e ruído de fundo.

A abordagem do trabalho não se concentra apenas na análise de informações de áudio, mas explora as informações multimodais incorporadas nos domínios de áudio e visual. Para isso, foi desenvolvido uma arquitetura de Rede Neural Convolucional (CNN) que combina representações aprendidas de ambas as modalidades em um estágio de fusão tardia. A abordagem é treinada e avaliada em dois conjuntos de dados de vídeo em grande escala: YouTube-8M e FCVID. As arquiteturas propostas demonstram resultados avançados no reconhecimento de

objetos de áudio e vídeo, oferecem robustez adicional às modalidades ausentes e permanecem computacionalmente baratos para treinar.

Os resultados demonstram que a rede audio-only (apenas áudio) e a abordagem multimodal dos autores apresenta desempenho muito próximo da taxa de desempenho humano para o subconjunto de instrumentos musicais do conjunto de dados YouTube-8M anotado automaticamente. Isso ilustra o fato de que as pessoas podem não apenas determinar o conceito de vídeo com base em pistas visuais, mas também nas auditivas. Além disso, os modelos considerados audio-only superam claramente os modelos video-only (apenas vídeo) e a rede multimodal tem um desempenho melhor do que aqueles baseados em uma única modalidade em um dos conjuntos de dados considerados, ilustrando a vantagem de várias modalidades.

4.5 RECONHECIMENTO DE EMOÇÕES ATRAVÉS DA FALA UTILIZANDO REDES NEURAIS (SOUZA, 2020).

Souza (2020) após uma etapa de experimentos apresenta as redes neurais: Rede neural recorrente, rede neural convolucional e *gated recurrent unit* para classificar emoções através da fala. Para treinar esses modelos foi utilizado o conjunto de dados *Ryerson Audio-Visual Database of Emotional Speech and Song* (RAVDESS), que construiu o ambiente para avaliação e testes, este passou por uma operação de transformação alongamento.

O conjunto de dados utilizado para o treinamento é composto por 1440 arquivos na língua inglesa norte americana, cada emoção é reproduzida em três intensidades e separados em sete: calma, feliz, triste, raiva, medo, surpresa e nojo. Este foi dividido 75% para treinamento e 25% para validação.

Dentre as etapas do desenvolvimento foi utilizado a técnica MFCC para extração de características que apresentou uma melhoria na captação das intenções por trás da fala. Foi utilizado também uma técnica de aumento de dados para melhorar o desempenho da rede neural, dentre as operações de transformação do áudio a que apresentou uma melhoria no desempenho foi o alongamento.

Foram experimentados quatro modelos cujo resultado apresentou a rede RNN-GRU alcançou desempenho melhor se comparado com uma CNN constatando assim que a rede que possui o melhor resultado foi a RNN-GRU com 79,69% de acurácia.

4.6 OUTROS TRABALHOS RELEVANTES

Klapuri (2011) discute técnicas para indução e correspondência de padrões em áudio musical descrevendo alguns métodos para extrair e recuperar esses padrões. Uma das técnicas utilizadas para a extração do objeto de interesse da música é o MFCC que representa o conteúdo timbral de um sinal em termos de distribuição de energia espectral, e aborda a indução de padrões que lida com o problema de detectar estruturas sequenciais repetidas na música e aprender o padrão subjacente a essas repetições, para isso, utiliza diversas técnicas como clusterização, matriz de auto distância, algoritmos de Lempel-Ziv-Welch (que substitui os padrões encontrados por códigos) e Modelos de Markov para previsão de sequência.

4.6.1 **PATTERN INDUCTION AND MATCHING IN MUSIC SIGNALS (KLAPURI, 2011).**

Este artigo discute técnicas para indução e correspondência de padrões em áudio musical. Em todos os níveis da música - harmonia, melodia, ritmo e instrumentação - a sequência temporal de eventos pode ser subdividida em padrões mais curtos que ás vezes são repetidos e transformados. São descritos métodos para extrair esses padrões de sinais de áudio musical (indução de padrões) e métodos computacionalmente viáveis para recuperar padrões semelhantes de um grande banco de dados de músicas (correspondência de padrões).

Para a extração do objeto de interesse da música, o artigo discute a utilização de um dos mais conhecidos que é o MFCC que representa o conteúdo timbral de um sinal em termos da distribuição de energia espectral, depois o artigo aborda a indução de padrões que lida com o problema de detectar estruturas sequenciais repetidas na música e aprender o padrão subjacente a essas repetições, para isso, utiliza diversas técnicas como clusterização, matriz de auto distância, algoritmos de Lempel-Ziv-Welch (que substitui os padrões encontrados por códigos) e Modelos de Markov para previsão de sequência.

Por último o artigo apresenta um capítulo que considera o problema de procurar em um banco de dados de música segmentos semelhantes a um determinado padrão, onde aborda o problema de alinhamento temporal na comparação de padrões, correspondência de padrões melódicos, padrões em dados de afinação polifônica, sequências de acordes e padrões e ritmos de bateria.

5 **DESENVOLVIMENTO**

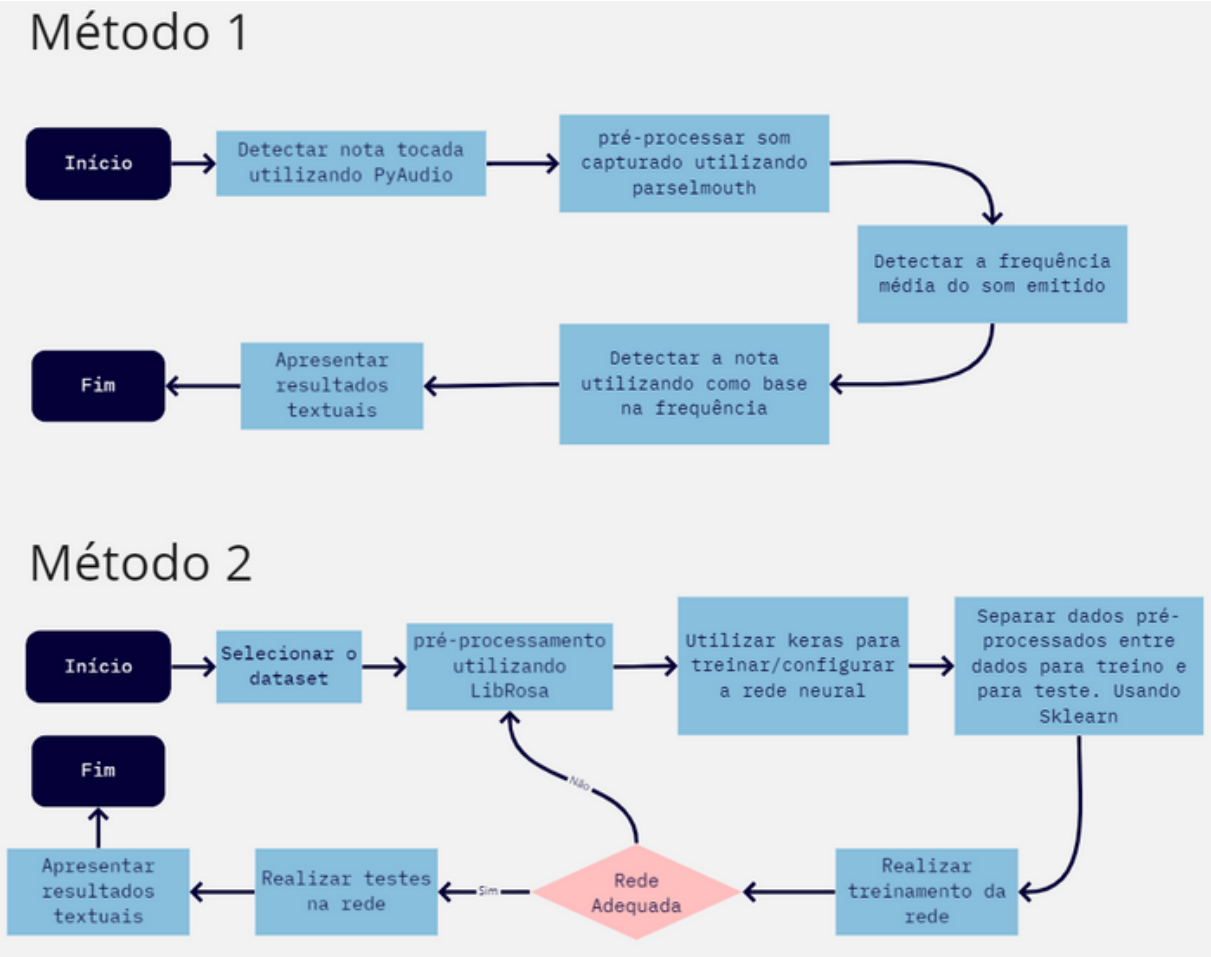
Neste capítulo, serão apresentadas as atividades realizadas para atingir os objetivos do trabalho, como conjunto de dados foram utilizados áudios gravados pelo autor em conjunto com áudios de exercícios online separando cada nota musical através do programa *Audacity*.

5.1 VISÃO GERAL DA PROPOSTA

A Figura 2 apresenta um resumo dos dois métodos utilizados nesse projeto, os detalhes de cada etapa serão descritos seções posteriores.

O primeiro método tem como objetivo detectar as notas musicais utilizando como base a frequência da nota. O segundo método utiliza como base uma rede neural.

Figura 2 — Desenvolvimento do trabalho



Fonte: O autor (2022)

A construção de dataset para o método 2 consta em recolher as notas musicais gravadas pelo autor e de exercícios musicais publicados online através do site desenvolvido por Rapacciuolo (2006).

Como pré-processamento esses exercícios foram exportados para o *Audacity* e foram cortadas apenas as notas C, D e E. Após exportar essas notas o pré-processamento dos dois métodos se difere.

O método 1 é mais tradicional e utiliza como base a nota tocada pelo usuário, a frequência da gravação é retirada automaticamente pelo *parselmouth* que depois retorna o valor com base em uma tabela de frequências publicada por Suits (1998).

O método 2 é mais complexo, utiliza uma biblioteca no pré-processamento para utilizar técnicas de MFCC para exibir as características das notas musicais, esses dados são colocados dentro da rede neural configurada e separados entre dados de validação (teste) e de treino.

As configurações de pré-processamento e da rede neural foram alteradas toda vez que os resultados não foram satisfatórios para assim conseguir apresentar os resultados mediante a acurácia.

5.2 CONJUNTO DE DADOS

Não foi necessário utilizar dados para o primeiro método, neste o método é analisado mediante gravação, ou seja, a nota é tocada diretamente no microfone e este é salvo e analisado.

Para o segundo método, foi necessário extrair as notas musicais dó, ré e mi de áudios de exercícios de trompetes, para isso foram acessados alguns meios sendo eles: gravações publicadas no site http://www.trumpetexercises.com/, vídeos no youtube e gravações de autoria própria.

A maioria das gravações são dos mesmos exercícios extraídos do livro de método para trompete Arban. Foram selecionados nove exercícios cada um com sua característica específica:

- Staccato: Página 22, exercício 48
- Legato: Página 39, exercício 01
- Notas curtas (0,5 tempo): Página 17, exercício 29
- Notas curtas (0,25 tempo): Página 26, exercício 15 e Página 59, exercício 03
- Notas longas (4 tempos): Página 12, final do exercício 07
- Notas longas (2 tempos): Página 12, exercício 09
- Notas longas (1 tempo): Página 14, exercício 17
- Compasso 6/8: Página 34, exercício 32

Desses exercícios foram extraídos os seguintes rótulos:

- Notas Dó 4 (C4): 43 notas
- Notas Dó 5 (C5): 111 notas
- Notas Ré 4 (D4): 70 notas
- Notas Ré 5 (D5): 111 notas
- Notas Mi 4 (E4): 94 notas
- Notas Mi 5 (E5): 54 notas

Todas essas notas possuem duração diferentes, estão em ritmos diferentes e a única coisa em comum é a frequência. Para separar as notas dos exercícios foi utilizado o programa *Audacity.* A separação de notas foi feita manualmente selecionando o trecho correto da nota executada no exercício.

5.3 FERRAMENTAS

O projeto foi desenvolvido com o auxílio de algumas ferramentas que serão detalhadas para facilitar o entendimento ao longo do projeto.

5.3.1 **Audacity**

É um software livre de edição digital de áudio, disponível na maioria dos sistemas operacionais, possui diversas ferramentas para edição de áudio, foi selecionado para o projeto devido a sua facilidade em cortar e exportar trechos de áudios. Esse programa foi utilizado para isolar as notas musicais.

5.3.2 **LibROSA**

Ferramenta desenvolvida por (MCFEE et al., 2015), Librosa é uma biblioteca para realizar análise de áudios e música, fornece uma interface fácil para extrair certas características e, para utilizar é necessário importar dentro do código e acessar o arquivo musical. Com o Librosa é possível acessar recursos como a extração do MFCC de cada nota musical.

5.3.3 **TensorFlow**

O TensorFlow é uma plataforma de código aberto para *machine learning*. Ele

possui um ecossistema abrangente e flexível de ferramentas, bibliotecas e recursos da comunidade que permite aos pesquisadores levar adiante *machine learning* e aos desenvolvedores criar e implantar aplicativos com tecnologia de *machine learning*.

Nesse projeto usamos a API do Keras que já implementa alguns dos modelos comumente utilizados, agilizando assim o desenvolvimento.

5.3.4 **PyAudio**

O PyAudio é uma biblioteca python que permite o acesso do python ao microfone independente do sistema operacional, no projeto ele é utilizado no primeiro método, onde ele grava alguns segundos de música para determinar se qual a nota que o usuário tocou.

5.3.5 **Scikit-Learn**

Scikit-learn é uma biblioteca python que é desenvolvida especificamente para aplicação prática de machine learning. Ela possui várias ferramentas de fácil utilização para análise preditiva de dados, possui código aberto e foi construída sobre bibliotecas bastante utilizadas no python, como NumPy, SciPy e Matplotlib. No projeto é utilizado para separar os dados entre dados de treino e de teste.

5.3.6 **Parselmouth**

Parselmouth é uma biblioteca python do software Praat, este software possui vários recursos voltados à análise fonética, como: Análise verbal, síntese verbal, segmentação e classificação, entre outros.

No projeto ele foi utilizado para sintetizar o som digitalmente e capturar o tom da gravação.

5.4 EXTRAÇÃO DE CARACTERÍSTICAS

Essa etapa é fundamental para que se obtenha sucesso em desenvolvimentos que visam fazer uma classificação, principalmente os que utilizam redes neurais e, como apresentado na seção 3.2.3, esse projeto conta com a utilização do método MFCC, que fornece um vetor dos atributos presentes no espectro.

Para auxiliar e facilitar a interação com o algoritmo MFCC, foi utilizado a biblioteca LibROSA, apresentada na seção 5.3.2.

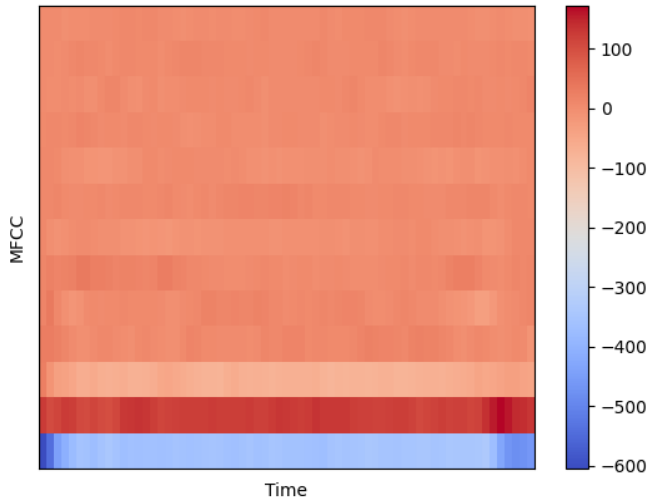
Os valores de entrada para o algoritmo foram definidos conforme documentação da biblioteca e testes realizados:

- **n_mfcc**: número de características retornadas do áudio de entrada
- **n_fft**: tamanho do sinal numa janela de tempo
- **hop_length**: representa o número de amostras entre quadros sucessivos, ex: as colunas de um espectrograma.

Foram testados alguns valores para o tamanho do vetor de características e estes apresentaram uma diferença significativa nos testes realizados, por isso foi optado por utilizar o número de 40 características. Alguns valores foram testados também no tamanho da janela de tempo, e o valor cujo testes foram melhores foi de 2048. No caso das amostras entre quadros sucessivos, não foram apresentadas diferenças significativas nos experimentos, mantendo assim o valor de 512 definido como padrão pela documentação.

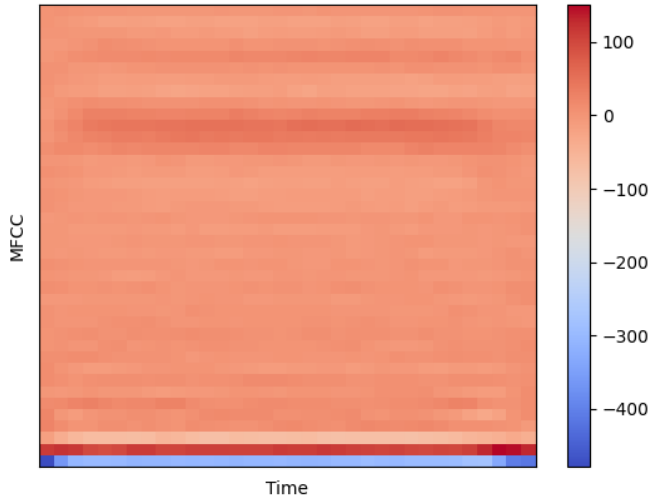
As figuras abaixo representam a comparação dos vetores de características durante o tempo do mesmo arquivo.

Figura 3 — Vetor de características ao longo do tempo: 13 características



Fonte: O autor (2022)

Figura 4 — Vetor de características ao longo do tempo: 40 características



Fonte: O autor (2022)

Ao analisar as figuras nós vemos que com apenas 13 categorias não se é possível identificar os detalhes do mapa de calor apresentados na figura 3 entre os valores 100 e -100. Com essas características apresentadas para a rede neural, é possível distinguir melhor as notas.

5.5 ARQUITETURAS PROPOSTAS

A proposta é comparar uma rede neural de classificação e um método de análise de frequência que consiste em reconhecer o som através da frequência média de um arquivo de áudio. Comparando esses métodos para obter qual arquitetura obtém melhor desempenho utilizando a métrica de precisão (acurácia). Com base nisso, abaixo serão apresentados os modelos que serão treinados e avaliados.

5.5.1 **Modelo 1: Análise de Frequência**

O primeiro modelo utiliza um método muito similar aos afinadores digitais hoje em dia, o objetivo dele é detectar a frequência do som e com base nela informar qual é a nota tocada.

A tabela a seguir mostra quais foram as frequências usadas para representar cada nota, foi utilizado como base a tabela de notas publicada por Suits (1998).

Tabela 1 — Frequências

Nota Real	Nota Trompete	Frequência Ideal	Largura de banda
Bb3	C4	233.08 Hz	226.54 Hz - 240.01 Hz
Bb4	C5	466.16 Hz	453.08 Hz - 480.02 Hz
C4	D4	261.63 Hz	254.285 Hz - 269.405 Hz
C5	D5	523.25 Hz	508.565 Hz - 538.81 Hz
D4	E4	293.66 Hz	285.42 Hz - 302.395 Hz
D5	E5	587.33 Hz	570.85 Hz - 604.79 Hz

Fonte: O autor (2022)

A tabela mostra por exemplo que se o som apresentar uma frequência de 230Hz ele deve retornar um texto visual mostrando que a nota é Bb3 (ou C4 para o trompete).

Nesse modelo foi utilizado também um retorno textual que apresenta se a nota está afinada ou não, para isso é verificado o quão próximo da frequência ideal a nota tocada se encontra, sendo que o mais próximo representa mais afinado.

5.5.2 **Modelo 2: Rede Neural**

Tabela 2 — Arquitetura da Rede Neural

Camada	Tipo	Neurônios	Ativação	Regularizador	Dropout
Entrada	Flatten	-	-	-	-
1	Dense	1024	ReLu	L2 - 0.001	0.3
2	Dense	512	ReLu	L2 - 0.001	0.3
3	Dense	256	ReLu	L2 - 0.001	0.3
4	Dense	64	ReLu	L2 - 0.001	0.3
Saída	Dense	12	Softmax	-	-

Fonte: O autor (2022)

A Tabela 2 apresenta as camadas da rede neural desenvolvida. A escolha de cada uma foi com base em testes realizados e obtiveram a melhor precisão.

Os modelos foram construídos utilizando camadas do tipo *Dense*, que é um tipo padrão para camadas densamente conectadas. A camada de entrada é do tipo *Flatten* e tem como argumento o (*input_shape*) que é do uma tupla de inteiros representando o formato de dado analisado.

As outras camadas utilizam ReLU como função de ativação que é o Rectified

Linear Unit, essa função de ativação vai retornar diretamente o valor se for positivo e caso não for, vai retornar zero. Geralmente ela possui melhor desempenho, possui uma boa convergência e reduz a probabilidade de gradiente de fuga.

Também foram adicionados regularizadores nas camadas com o intuito de aplicar penalidades para reduzir a complexidade da rede neural e reduzir o *overfitting* (ou sobreajuste). O regularizador escolhido foi o L2 (ou queda de peso), que resumidamente transforma os pesos um pouco menores a cada vez que uma atualização ocorre.

Com o intuito de reduzir o *overfitting* foi adicionado *dropout* em cada camada, este transforma um *input* aleatório em 0 a cada atualização durante a fase de treino, diminuindo a complexidade da rede e reduzindo o *overfitting*.

A última camada possui *softmax* como função de ativação, essa função tem como objetivo normalizar a saída dos valores da rede neural.

5.5.2.1 Conjunto de treinamento

Dentro de todos os dados apresentados na seção 5.2 foram separados entre treinamento e validação, onde 70% foi utilizado para validação e 30% para treinamento, para isso foi utilizado uma biblioteca do scikit-learn que separa aleatoriamente os dados em subconjuntos de teste e validação.

5.5.2.2 Configurações finais

Existem alguns algoritmos que otimizam a conversão do modelo. O modelo foi otimizado usando o algoritmo chamado Adam que vai minimizar a função de perda em relação aos parâmetros da rede, de acordo com Kinga e Ba (2017) Adam é um método computacionalmente eficiente, tem pouca necessidade de memória, invariante ao reescalonamento diagonal de gradientes e é adequado para problemas grandes em termos de dados/parâmetros.

As métricas de perda tem como objetivo calcular a quantidade que um modelo deve procurar minimizar durante o treinamento, para o modelo foi utilizado a métrica probabilística chamada *Sparse Categorical Crossentropy* que calcula a perda de entropia cruzada entre os rótulos e as previsões, é recomendado utilizar essa função de perda de entropia cruzada quando houver duas ou mais classes de rótulos.

Epoch é o número de vezes que os dados passam por todas as camadas de rede. O número atribuído para o modelo foi de 100.

6 **IMPLEMENTAÇÃO**

Esse capítulo vai ser separado em duas partes para facilitar o entendimento, onde será explicado o código e alguns resultados de cada método.

6.1 MÉTODO 1: ANÁLISE DE FREQUÊNCIA

6.1.1 **Captura do áudio**

Para a detecção de áudio, foi utilizado como base o exemplo da documentação do próprio PyAudio, inicialmente foram separados algumas variáveis a serem usadas em todo o arquivo:

Código 1 — Captura do áudio - Variáveis

```
1   import pyaudio
2   import wave
3
4
5   CHUNK = 1024
6   FORMAT = pyaudio.paInt16
7   CHANNELS = 2
8   RATE = 44100
9   RECORD_SECONDS = 5
10  WAVE_OUTPUT_FILENAME = "output.wav"
```

Fonte: O autor (2022)

- CHUNK é o número de quadros dentro do buffer
- FORMAT é o tipo de inteiro que vai ser utilizado na aplicação
- CHANNELS indica a quantidade que cada quadro vai utilizar, nesse caso 2.
- RATE representa o número de amostras coletadas por segundo
- RECORD_SECONDS é o tempo que o programa vai gravar do microfone
- WAVE_OUTPUT_FILENAME é o nome do arquivo que vai ser salvo.

Sabendo de todas essas variáveis vamos para a função de detecção do som.

Código 2 — Captura do áudio - Função

```
12  def detect_audio():
13      print("Você tem {} segundos...".format(RECORD_SECONDS))
14      p = pyaudio.PyAudio()
15
16      stream = p.open(format=FORMAT,
17                      channels=CHANNELS,
18                      rate=RATE,
19                      input=True,
20                      frames_per_buffer=CHUNK)
21
22      print("* Gravando")
23
24      frames = []
25
26      for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
27          data = stream.read(CHUNK)
28          frames.append(data)
29
30      print("* Gravação finalizada")
31
32      stream.stop_stream()
33      stream.close()
34      p.terminate()
35
36      wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
37      wf.setnchannels(CHANNELS)
38      wf.setsampwidth(p.get_sample_size(FORMAT))
39      wf.setframerate(RATE)
40      wf.writeframes(b''.join(frames))
41      wf.close()
```

Fonte: O autor (2022)

O código acima cria um fluxo de áudio (*stream)* usando o PyAudio, e cada quadro de buffer é salvo dentro de um *array* onde depois que o tempo de fluxo termina, ele salva o array em um arquivo *.wav*.

6.1.2 **Média de frequência**

Essa função identifica um arquivo com diferentes tipos de frequência gravadas, então retorna a média dessa frequência.

Código 3 — Média de Frequência - Função

```
23  # Pega a média de tons dentro da música
24  def average_pitch(pitch):
25      pitch_values = pitch.selected_array['frequency']
26      pitch_values = np.trim_zeros(pitch_values)
27      pitch_values = list(filter(lambda num: num !=0 and num != 0.0, pitch_values))
28
29      if sum(pitch_values) == 0 or len(pitch_values) == 0:
30          print("Som não detectado")
31          return 0
32
33      average = sum(pitch_values) / len(pitch_values)
34      # sp for single_pitch
35      if LOG_TYPE == 'full':
36          for sp in pitch_values:
37              print("Hz: {}".format(sp))
38
39      if LOG_TYPE == 'full' or LOG_TYPE == 'short':
40          print("Média de Hz: {}".format(average))
41
42      return average
```

Fonte: O autor (2022)

Como parâmetro ele precisa de um arquivo que provém de uma função do *Parselmouth* chamada *to_pitch(),* essa função retorna um objeto que possui algumas propriedades como a frequência, nas primeiras linhas é selecionado essa característica, transformado em um array cujos valores são todas as frequências detectadas no arquivo de áudio.

Logo depois salvamos uma variável com a média simples dos valores para retornar o valor da média.

6.1.3 **Detecção da nota**

Para essa função usamos como base os valores explicados na seção 5.5.1:

Código 4 — Detecção notas

```
51  def detect_note(pitch):
52      # Bb3 - 233.08 Hz |- 6.54| |+ 6.93|
53      if pitch > 226.54 and pitch < 240.01:
54          return "C4"
55
56      # Bb4 - 466.16 Hz |- 13.08| |+ 13.86|
57      if pitch > 453.08 and pitch < 480.02:
58          return "C5"
59
60      # C4 - 261.63 Hz |- 7.345| |+ 7.775|
61      if pitch > 254.285 and pitch < 269.405:
62          # afinacao = pitch - 261.63
63          return "D4"
64
65      # C5 - 523.25 Hz |- 14.685| |+ 15.56|
66      if pitch > 508.565 and pitch < 538.81:
67          return "D5"
68
69      # D4 - 293.66 Hz |- 8.24| |+ 8.735|
70      if pitch > 285.42 and pitch < 302.395:
71          return "E4"
72
73      # D5 - 587.33 Hz |- 16.48| |+ 17.46|
74      if pitch > 570.85 and pitch < 604.79:
75          return "E5"
76
77      # return "Nota fora do escopo"
78      return "-"
```

Fonte: O autor (2022)

Como parâmetro de entrada é utilizado o valor da função descrita na seção 6.1.2. O valor da extensão completa aceita para detectar a nota é determinado seguindo essas equações:

$$ValorMínimo = F_{atual} - \left| \frac{(F_{ant} - F_{atual})}{2} \right| \quad (1)$$

$$ValorMáximo = \left| \frac{(F_{prox} - F_{atual})}{2} \right| + F_{atual} \quad (2)$$

Onde F_{atual} é igual a frequência/tom atual que está querendo medir, F_{ant} é a frequência/tom da nota anterior e F_{prox} representa a frequência/tom da próxima nota. Um exemplo seria a nota Bb3, conforme publicado por Suits (1998) sabemos que a frequência dele é 233.08, a frequência da nota anterior é 220 e a frequência da próxima nota é 246.96. Nós chegamos então no seguinte resultado, o valor da frequência mínima para ser identificada como Bb3 é de 226.54 e o valor máximo é de 240.01.

6.1.4 **Resultado**

Com o objetivo de chegar a um resultado de acurácia para comparação, foi necessário utilizar o mesmo conjunto de dados como entrada para a função descrita na seção 6.1.2 ao invés de usar a detecção de áudio descrita na seção 6.1.1.

Cada arquivo de áudio que representa a nota foi separado dentro de pastas com a respectiva nota, como exemplo, todas as notas C4 se encontram dentro da pasta com o mesmo nome (C4).

O próximo trecho de código apresenta um objeto criado para guardar as informações necessárias para calcularmos a acurácia:

Código 5 — Objeto Acurácia Individual

```
acuraciaIndividual = {
    "C4": {"length": 0, "notascorretas": 0, "acuracia" : 0},
    "C5": {"length": 0, "notascorretas": 0, "acuracia" : 0},
    "D4": {"length": 0, "notascorretas": 0, "acuracia" : 0},
    "D5": {"length": 0, "notascorretas": 0, "acuracia" : 0},
    "E4": {"length": 0, "notascorretas": 0, "acuracia" : 0},
    "E5": {"length": 0, "notascorretas": 0, "acuracia" : 0}
};

for dirpath, dirnames, filenames in os.walk(DATASET_PATH):
  if dirpath is not DATASET_PATH and "MP3" not in dirpath:
    dirpath_components = dirpath.split('/')
    label = dirpath_components[-1]
    for filename in filenames:
        snd = parselmouth.Sound(dirpath+'/'+filename)
        pitch = snd.to_pitch()
        average = average_pitch(pitch)
        detected = detect_note(average)
        correta = 1 if detected == label else 0
        acuraciaIndividual[label]["length"] = acuraciaIndividual[label]["length"] + 1
        acuraciaIndividual[label]["notascorretas"] = acuraciaIndividual[label]["notascorretas"] + correta
        print("pitch: {}".format(detect_note(average)))
```

Fonte: O autor (2022)

A iteração apresentada após a criação do objeto mostra o sistema percorrendo cada pasta criada e executando a função de detectar a nota apresentada na seção 6.1.3, logo depois o código atualiza o respectivo objeto com o tamanho e se a nota identificada foi correta ou não.

Ao final dessa execução foi necessário calcular a acurácia de cada nota musical em porcentagem, para isso foi necessário multiplicar as notas identificadas por 100 e dividir pela quantidade total, como mostra a seguinte equação:

$$AcuraciaIndividual = \frac{Notas_{corretas} \cdot 100}{Notas_{total}} \tag{3}$$

O código que representa essa equação se encontra abaixo:

Código 6 — Cálculo da Acurácia Individual

```
# Equação 3
acuraciaIndividual["C4"]["acuracia"] = (acuraciaIndividual["C4"]["notascorretas"] * 100)/acuraciaIndividual["C4"]["length"]
acuraciaIndividual["C5"]["acuracia"] = (acuraciaIndividual["C5"]["notascorretas"] * 100)/acuraciaIndividual["C5"]["length"]
acuraciaIndividual["D4"]["acuracia"] = (acuraciaIndividual["D4"]["notascorretas"] * 100)/acuraciaIndividual["D4"]["length"]
acuraciaIndividual["D5"]["acuracia"] = (acuraciaIndividual["D5"]["notascorretas"] * 100)/acuraciaIndividual["D5"]["length"]
acuraciaIndividual["E4"]["acuracia"] = (acuraciaIndividual["E4"]["notascorretas"] * 100)/acuraciaIndividual["E4"]["length"]
acuraciaIndividual["E5"]["acuracia"] = (acuraciaIndividual["E5"]["notascorretas"] * 100)/acuraciaIndividual["E5"]["length"]
```

Fonte: O autor (2022)

Ao final da execução desse cálculo, temos o seguinte objeto:

Código 7 — Resultado Acurácia Individual

```
acuracia: {'C4': {'length': 43, 'notascorretas': 32, 'acuracia': 74.4186046511628}, 'C5': {'length': 111, 'notascorretas': 105,
 'acuracia': 94.5945945945946}, 'D4': {'length': 70, 'notascorretas': 51, 'acuracia': 72.85714285714286}, 'D5': {'length': 111,
 'notascorretas': 92, 'acuracia': 82.88288288288288}, 'E4': {'length': 94, 'notascorretas': 73, 'acuracia': 77.65957446808511},
 'E5': {'length': 54, 'notascorretas': 41, 'acuracia': 75.92592592592592}}
```

Fonte: O autor (2022)

Esse resultado mostra a porcentagem de acerto do algoritmo criado para cada nota, a nota C4 por exemplo possui 43 arquivos porém apenas 32 notas foram identificadas corretamente, resultando numa acurácia de aproximadamente 74,42%.

Após esse resultado se fez necessário capturar uma acurácia de todo o modelo, para isso foi necessário somar todas as acurácias das notas individuais e dividir pela quantidade de notas identificadas, como é mostrado na equação abaixo:

$$AcuraciaTotal = \frac{\sum AcuraciaIndividual}{QuantidadeNotas} \tag{4}$$

O código que representa essa equação se encontra abaixo:

Código 8 — Cálculo Acurácia Total

```
118  # Equação 4
119  somaAcuracias = acuraciaIndividual["C4"]["acuracia"]
120  + acuraciaIndividual["C5"]["acuracia"]
121  + acuraciaIndividual["D4"]["acuracia"]
122  + acuraciaIndividual["D5"]["acuracia"]
123  + acuraciaIndividual["E4"]["acuracia"]
124  + acuraciaIndividual["E5"]["acuracia"]
125  print("total: {}".format(somaAcuracias/6))
```

Fonte: O autor (2022)

O resultado desse cálculo foi um total aproximado de 79.72%. Essa acurácia um pouco baixa pode se dar por diversos fatores, um deles é a quantidade de ruídos presente nos arquivos que alteram a frequência dos arquivos.

6.2 MÉTODO 2: REDE NEURAL

6.2.1 **Pré-processamento**

O objetivo dessa etapa é capturar e digitalizar todas as características dos áudios a serem utilizados pela rede neural, para isso todos os áudios tem que estar no formato *.wav.* Para isso foi necessário converter os áudios que não estavam nesse formato.

Código 9 — Pré-processamento: Definição de função

```
13  def save_mfcc(dataset_path, json_path, n_mfcc= 40, n_fft=2048, hop_length=512, num_segments=5):
14    data = {
15      "mapping": [],
16      "mfcc": [],
17      "labels": []
18    }
19    num_samples_per_segment = int(SAMPLES_PER_TRACK / num_segments)
```

Fonte: O autor (2022)

O trecho de código acima mostra a definição da função que exporta o arquivo que vai ser utilizado pela rede neural, alguns dos parâmetros de entrada são explicados na seção 5.4, os outros são:

- *dataset_path:* o caminho onde ficam salvos os arquivos de áudio.
- *json_path:* representa o caminho que o resultado desta função vai ser salvo.

- *SAMPLE_PER_TRACK*: é o resultado da multiplicação da taxa de amostragem com a duração do áudio, para esses valores foram utilizados o um padrão, resultando em um valor de 2205.

O arquivo exportado é do tipo *.json* com os seguintes *arrays*: *mapping*, *mfcc* e *labels*. Mapping representa quais são as categorias existentes nos áudios, nesse caso são eles: "C4", "C5", "D4", "D5", "E4" e "E5". MFCC representa os valores encontrados no arquivo, é na verdade uma matriz onde representa uma lista de valores fracionários, positiva ou negativamente. Label representa em que categoria se espera que cada arquivo de música se encaixe, ou seja, se o arquivo é um C4 é esperado que a categoria que ele se encaixe seja a primeira, representado por 0 nesse arquivo.

Código 10 — Pré-processamento: Estruturando o arquivo

```
21  #loop the folders.
22  for i, (dirpath, dirnames, filenames) in enumerate(os.walk(dataset_path)):
23
24    # ensure that we're not at the root level
25    if dirpath is not dataset_path and "MP3" not in dirpath :
26
27      #save the semantic label
28      dirpath_components = dirpath.split("/")
29      semantic_label = dirpath_components[-1]
30      data["mapping"].append(semantic_label)
31      print("\nProcessing {}".format(semantic_label))
32      # process files for a specific genre
33      for f in filenames:
34
35        # load audio file
36        file_path = os.path.join(dirpath, f)
37        signal, sr = librosa.load(file_path, sr=SAMPLE_RATE)
38
39        #process segments extracting mfcc and storing data
40        for s in range(num_segments):
41          start_sample = num_samples_per_segment * s # s=0 -> 0
42          finish_sample = start_sample + num_samples_per_segment #s=0 -> num_samples_per_segment
43
44          mfcc = librosa.feature.mfcc(y=signal[start_sample:finish_sample], sr=sr, n_fft=n_fft, n_mfcc=n_mfcc,hop_length=hop_length)
45          mfcc = mfcc.T
46
47          data["mfcc"].append(mfcc.tolist())
48          data["labels"].append(i-1)
49          print("{}, segment: {}".format(file_path, s+1))
50
51  with open(json_path, "w") as fp:
52    json.dump(data, fp, indent=4);
```

Fonte: O autor (2022)

Continuando a função, foi criado um *loop* (laço de iteração) por todos os arquivos dentro do diretório, excluindo aqueles que estão dentro da pasta MP3.

Cada iteração representa uma pasta predefinida com os nomes das categorias, então o trecho da linha 28 até a 30 está apenas salvando no arquivo qual é a categoria, logo depois entra em um outro *loop* que itera sobre cada arquivo dentro desta pasta.

Utilizando o librosa, é capturado o sinal e o *sample rate* do arquivo de áudio, sinal este que logo é separado em pedaços de acordo com o número de segmentos

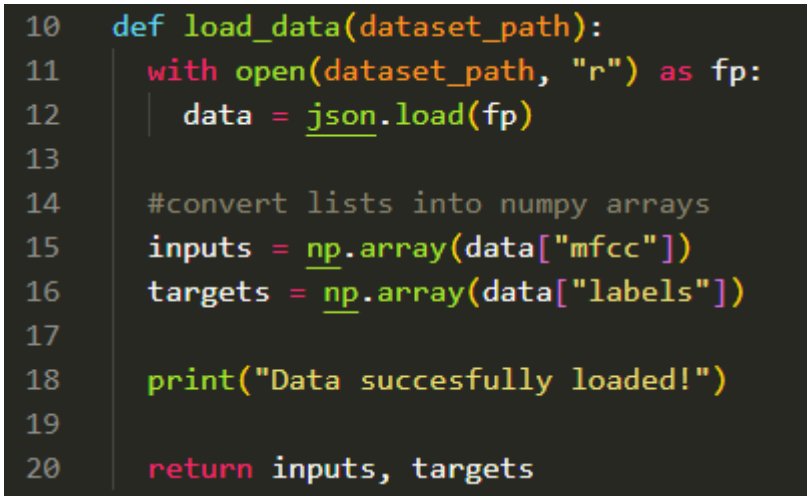
como podemos ver a partir da linha 40. Dentro desse último *loop,* é extraído o *mfcc* de cada segmento usando o librosa (linha 44) e logo depois é salvo dentro do arquivo *.json* que está sendo montado.

As últimas estão apenas salvando o arquivo *.json* pronto para ser utilizado pela rede neural.

6.2.2 **Arquitetura da rede neural**

Antes de mostrar as configurações exatas da arquitetura da rede neural, é necessário pegar os dados pré-processados pela função descrita na seção 6.2.1.

Código 11 — Rede Neural: Carregar arquivo



```
10  def load_data(dataset_path):
11      with open(dataset_path, "r") as fp:
12          data = json.load(fp)
13
14      #convert lists into numpy arrays
15      inputs = np.array(data["mfcc"])
16      targets = np.array(data["labels"])
17
18      print("Data succesfully loaded!")
19
20      return inputs, targets
```

Fonte: O autor (2022)

Para isso foi necessário carregar o arquivo *.json* e transformá-los em arrays do tipo *numpy* para que podemos manusear com mais facilidade, note que essa função tem como objetivo retornar os dados musicais *mfcc* salvos e as categorias em que cada dado se coloca.

Código 12 — Rede Neural: Arquitetura

```
if __name__ == "__main__":
    #load data
    inputs, targets = load_data(DATASET_PATH)
    #split data into train and test sets
    inputs_train, inputs_test, targets_train, targets_test = train_test_split(inputs,
                                                                              targets,
                                                                              test_size=0.3)
    #build the network architecture

    model = keras.Sequential([
        # input layer
        keras.layers.Flatten(input_shape=(1, 40)),
        #1st hidden layer, relu = rectified linear unit. Relu - Better convergence, reduced likelihood of vanishing gradient
        keras.layers.Dense(1024, activation="relu",  kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.3),
        #2st hidden layer
        keras.layers.Dense(512, activation="relu",  kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.3),
        #3st hidden layer
        keras.layers.Dense(256, activation="relu", kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.3),
        #4st hidden layer
        keras.layers.Dense(64, activation="relu", kernel_regularizer=keras.regularizers.l2(0.001)),
        keras.layers.Dropout(0.3),
        # output layer, 12 neurons because we have 6 outputs ("C4", "C5", "D4", "D5", "E4", "E5")
        # softmax its a activation function, that normalize the values for us.
        keras.layers.Dense(12, activation="softmax"),

    ])
```

Fonte: O autor (2022)

No começo do arquivo vemos que os arquivos de dados estão sendo separados entre testes e treino conforme descrito na seção 5.5.2.1 e, logo em seguida começa a construção da arquitetura de rede, os detalhes da rede foram descritos na seção 5.5.2 e cada parte foi configurada para apresentar a melhor acurácia.

6.2.3 **Compilação da rede neural**

Nessa etapa é apresentado a compilação e o treino do modelo.

Código 13 — Rede Neural: Compilação e treino

```
76  #compile network
77  optimiser = keras.optimizers.Adam(learning_rate=0.0001)
78  model.compile(optimizer=optimiser,
79                loss="sparse_categorical_crossentropy",
80                metrics=["accuracy"])
81  model.summary()
82
83  #train network
84  history = model.fit(inputs_train, targets_train,
85                      validation_data=(inputs_test, targets_test),
86                      batch_size=32,
87                      epochs=100)
88
89  #plot the accuracy and error over the epochs
90  plot_history(history)
```

Fonte: O autor (2022)

No primeiro trecho do código vemos como é feito para adicionar um algoritmo de otimização, um algoritmo de perda e qual é a métrica que queremos analisar, que é a acurácia.

O modelo é treinado então na linha 84, onde possui como parâmetro de entrada os dados separados, e por último é apresentado um gráfico com o resultado do treino. Todos parâmetros apresentados pela compilação e treino estão descritos na seção 5.5.2.2

6.2.4 **Apresentação dos resultados**

Essa função tem como objetivo preparar e apresentar o gráfico com as respectivas acurácias e perdas.

Código 14 — Rede Neural: Apresentação dos resultados

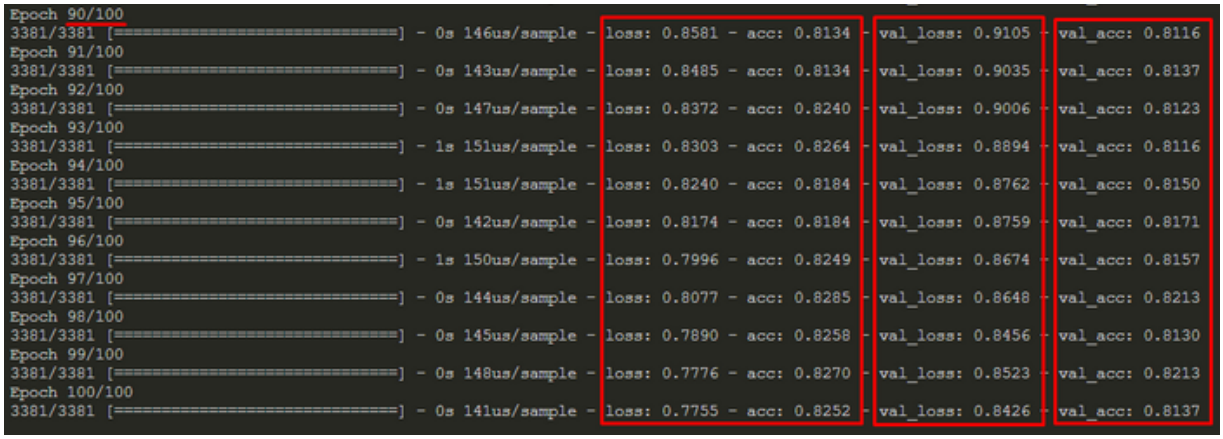
```python
def plot_history(history):

    fig, axs = plt.subplots(2)

    # create the accuracy subplots
    axs[0].plot(history.history["acc"], label="Treino (acurácia)")
    axs[0].plot(history.history["val_acc"], label="Teste (acurácia)")
    axs[0].set_ylabel("Acurácia")
    axs[0].legend(loc="lower right")
    axs[0].set_title("Acurácia")

    # create the error subplots
    axs[1].plot(history.history["loss"], label="Treino (erro)")
    axs[1].plot(history.history["val_loss"], label="Teste (erro)")
    axs[1].set_ylabel("Erro")
    axs[1].set_xlabel("Epoch")
    axs[1].legend(loc="upper right")
    axs[1].set_title("Erro")

    plt.show()
```

Fonte: O autor (2022)

No caso desse código é utilizado o *pyplot* da biblioteca do *matplotlib* que é uma biblioteca bastante utilizada no python, para essa função é necessário um histórico como parâmetro, onde ele cria dois gráficos, um apresentando a acurácia e o outro o erro.

Ao executar o código é mostrado nos logs cada *Epoch* executado com os valores salvos:

Código 15 — Rede Neural: Epochs



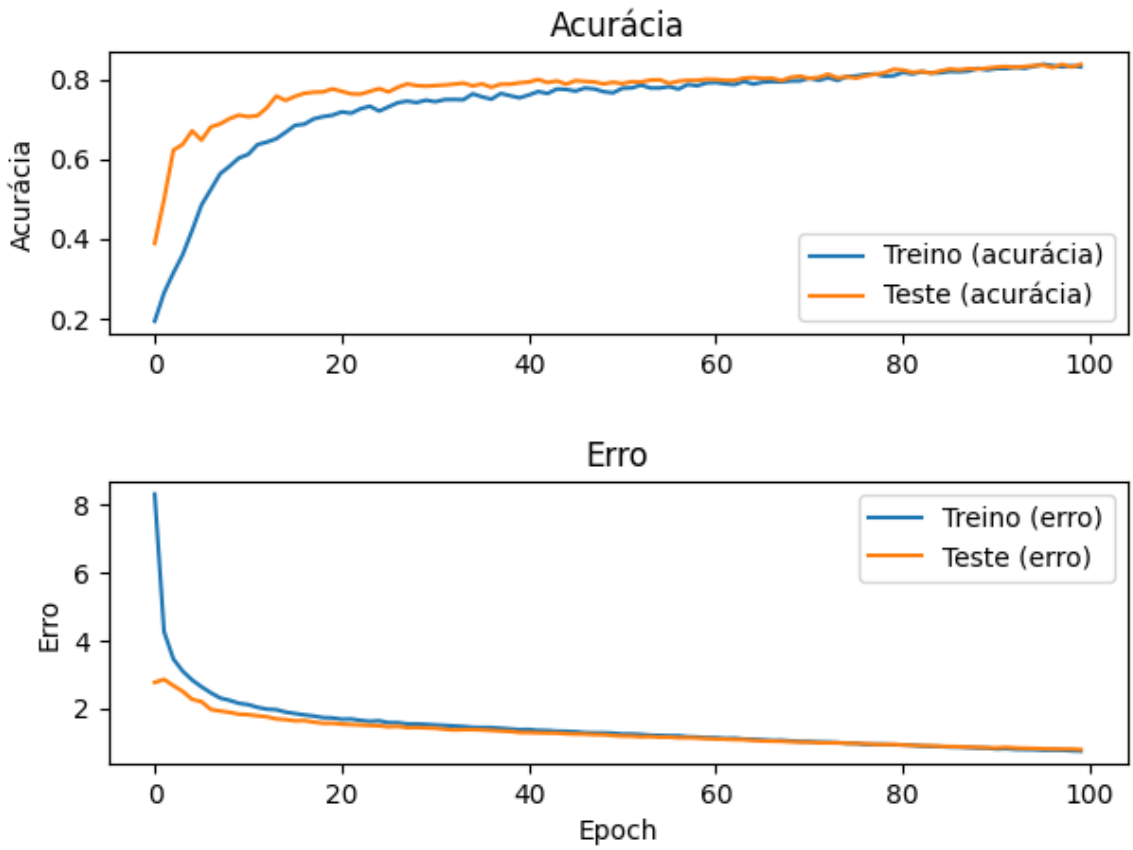
Fonte: O autor (2022)

Nesse log, vemos algumas variáveis: *loss, acc, val_loss* e *val_acc.* Esses dados representam os valores separados para treinar a rede neural e os valores que validam a rede (valores de teste), sendo *loss* a perda do valor de treino, *acc* a acurácia do valor de treino, *val_loss* a validação da perda e *val_acc* o valor de teste da acurácia do modelo.

Ou seja, no *Epoch* 90 por exemplo, vemos que os dados de treino possuem uma perda de 85% e uma acurácia de 81%, já os valores de teste da rede possuem o valor de perda 91% e o de acurácia 81%.

Ao final da execução do programa temos como resultado o seguinte gráfico:

Gráfico 1 — Rede Neural: Resultado



Fonte: O autor (2022)

Na horizontal vemos os *Epochs* que representam cada iteração que os dados passaram pela rede neural, na vertical vemos o número de porcentagem de cada situação, vemos que a acurácia da rede neural chegou acima dos 80%.

6.3 RESULTADOS

Após todos os experimentos, observou-se que a diferença entre os modelos foi significativa, ressaltando a qualidade da análise de notas através da frequência que alcançou um desempenho melhor se comparado com a rede neural.

Apesar do resultado, podemos observar que a rede neural alcançou uma acurácia bastante significativa se comparado com os trabalhos correlatos, a tabela abaixo mostra os trabalhos correlatos em relação com o modelo de rede neural apresentado no presente trabalho.

Todos os trabalhos na tabela abaixo tiveram a avaliação quantitativa e todas exceto trabalho de (TAMBOLI; KOKATE, 2019, p. 3) apresentaram como métrica a acurácia, nesse projeto de Tamboli e Kokate (2019, p. 3) foi apresentado como

métrica: Precisão, recall/sensibilidade e especificidade.

Tabela 3 — Tabela de comparação de desempenho com outros modelos

Referência	Abordagem	Data Set	Tamanho do Data Set	Resultado
(TAMBOLI; KOKATE, 2019)	Rede Neural Baseada em Otimização (OBNN)	Um banco de dados de notas musicais com vários gêneros de músicas como blues, eletrônica, folk, country, jazz, pop e rock.	Não informado	OBNN: 75%; NN: 65%
(DE JESUS GUTERRO-TURRUBIATES et al., 2014)	Rede Neural Artificial, usando MLP (Perceptron multicamada)	Um banco de dados de diversos instrumentos como fonte de áudio	16.384 de um banco maior de 44.100	97,5%
(BARROS; LUZ; BAIA, 2019)	Rede Neural Artificial (Rede Treinada)	Amostra de notas musicais de um violão	12 notas (Escala cromática)	94,44%
(SLIZOVSKAIA; GÓMEZ; HARO, 2017)	Redes Neurais Convolucionais	Vários vídeos extraídos do FCVID e YouTube-8M	5.154 do FCVID e 60.862 do YouTube-8M	30% até 89% dependendo do instrumento
(SOUZA, 2020)	Rede Neural Recorrente e Gated Recorrent Unit	Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS)	1440 arquivos de áudio	79,69%
Presente Trabalho: Método 1	Análise de frequência	Retirado de áudios produzidos pelo autor e vídeos/ áudios publicados online.	483 arquivos wav	79%
Presente Trabalho: Método 2	Rede Neural Artificial	Retirado de áudios produzidos pelo autor e vídeos/ áudios publicados online.	483 arquivos wav	82%

Fonte: O autor (2022)

7 **CONCLUSÃO**

O presente trabalho buscou apresentar um modelo de análise e reconhecimento musical através de redes neurais, com uma pesquisa na literatura com foco nas diversas maneiras de identificação de som através de diferentes redes neurais.

Por meio da revisão bibliográfica, foram identificados diferentes tipos de modelos com capacidade para análise musical. Diferente do tipo de rede neural, a técnica MFCC para extração de características é utilizada para captar não apenas os sons musicais mas também a intenção por trás dos sons.

No começo do trabalho tinha-se o objetivo de analisar e reconhecer partituras completas de exercícios musicais de diferentes instrumentos, identificar afinação, tempo e dinâmica. Porém, devido a falta de dados para fazer essa análise com mais precisão, foi decidido analisar notas musicais separadas como afinação, o trompete como instrumento melódico e foi decidido fazer uma comparação com o método tradicional de afinação.

Tendo em vista a acurácia como ponto principal da avaliação, ao decorrer do trabalho foi identificado a importância do tamanho do conjunto de dados para treinamento e validação, e como influenciavam na precisão do modelo. Em testes iniciais foi verificado que os dados de treino eram superiores e distantes dos dados de teste, foram então aplicadas técnicas para resolução de *overfitting,* diminuindo a distância entre teste e treino. Além disso, o desempenho da rede neural melhorou significativamente após o aumento de características através do MFCC.

Com os resultados apresentados nesse projeto, percebe-se que mesmo executando a rede neural com um aumento significativo de *Epochs,* a tendência é que a acurácia se mantenha por volta dos resultados obtidos, provavelmente isso se dá pela quantidade de dados. Com uma quantidade maior de dados há a possibilidade de identificar com mais precisão as notas executadas porém, caso haja uma variedade de instrumentos musicais de diferentes tipos, existe o risco de diminuir a acurácia obtida.

7.1 TRABALHOS FUTUROS

Para a realização de trabalhos futuros é importante explorar todas as notas da escala cromática (Dó, Dó#/Réb, Ré, Ré#/Réb, Mi, Fá, Fá#/Solb, Sol, Sol#/Láb, Lá, Lá#/Sib, Si.). Também é importante utilizar algumas técnicas diferentes de aumento de precisão, testar novos parâmetros de rede e do método de extração MFCC, visando a continuidade do trabalho.

Uma das sugestões é executar a mesma rede com mais de um instrumento, não necessariamente sendo do mesmo naipe ou categoria. Isso criaria modelos de análise de notas musicais mais eficientes, porém limitaria a rede neural para identificação de notas.

Visando identificar a duração das notas e a dinâmica seria necessário desenvolver uma rede neural com rótulos mais específicos envolvendo todas as três características, como por exemplo: "C4 4t piano", "C4 4t forte", "C4 2t piano", e assim por diante, evidenciando a diferença entre tempo e dinâmica.

REFERÊNCIAS

BARROS, Anderson dos Santos de; LUZ, Heverton Pires da; BAIA, Joas Weslei. Inteligência Artificial na Educação Musical. **Revista de Informática Aplicada**, v. 15, n. 2, 18 09 2019. Disponível em: https://seer.uscs.edu.br/index.php/revista_informatica_aplicada/article/view/6979/3110. Acesso em: 17 nov. 2020.

BRAGA, Antônio de Pádua; CARVALHO, André Ponce de Leon F. de; LUDEMIR, Teresa Bernarda. **Redes Neurais Artificiais**: Teoria e aplicações. LTC, 2000.

DATA SCIENCE ACADEMY. **Função de Ativação**. Deep Learning Book. 2022. Disponível em: https://www.deeplearningbook.com.br/funcao-de-ativacao/. Acesso em: 15 jul. 2022.

DE JESUS GUTERRO-TURRUBIATES, Jose *et al*. Pitch estimation for musical note recognition using Artificial Neural Networks. **IEEE - International Conference on Electronics, Communications and Computers (CONIELECOMP)**. 5 p, 2014. Disponível em: https://ieeexplore.ieee.org/document/6808567. Acesso em: 11 set. 2019.

DUNNETT, Ben. **Transposing Instruments**. Music Theory Academy. Disponível em: https://www.musictheoryacademy.com/how-to-read-sheet-music/transposing-instruments/. Acesso em: 8 jul. 2022.

FACELI, Katti *et al*. **Inteligência Artificial**: Uma Abordagem de Aprendizado de Máquina. Rio de Janeiro: Grupo Gen - LTC, 2011. 396 p. Disponível em: https://books.google.com.br/books?id=4DwelAEACAAJ. Acesso em: 8 jul. 2022.

FREIRE, Ricardo Dourado. Como será que eu afino? A relação entre sistemas de afinação e parâmetros de afinação na performance musical. **Revista Música Hodia**, Goiânia, v. 16, n. 2, p. 133-144, 2016. Disponível em: https://revistas.ufg.br/musica/article/view/45333. Acesso em: 8 jul. 2022.

GEORGE, Susan E.. Online Pen-Based Recognition of Music Notation with Artificial Neural Networks. **Computer Music Journal**, v. 27, n. 2, p. 70 - 79, jun. 2003.

HALLAM, Susan. **Music Psychology in Education**. London: Institute of Education, University of London, v. 25, 2006. 281 p.

HAYKIN, Simon. **Redes Neurais**: Princípios e Prática. Bookman Editora, v. 1, f. 449, 2006. 898 p.

HOMENDA, Wladyslaw; LUCKNER, Marcin. Automatic Recogniton Of Music Notation Using Neural Network. **International Conference on AI and Systems**, Divnormorkoye, Russia. 6 p, jan. 2004.

HSIEH, S. *et al*. Brain correlates of musical and facial emotion recognition: Evidence from the dementias. **Neuropsychologia**. 9 p. 2012.

JAIN, Anil K.; MAO, Jianchang; MOHIUDDIN, K. M.. Artificial Neural Networks: A Tutorial. **IEEE Computer**, v. 29, p. 31-44, 1996.

JÚNIOR, Carlos Roberto Ferreira de Menezes; FARIA, Eustáquio São José de ; YAMANAKA, Keiji. Reconhecendo Instrumentos Musicais Através de Redes Neurais Artificiais. **Hífen**, Uruguaiana, v. 31, n. 59/60, 2007. I/II Semestre.

KINGA, Diederik P.; BA, Jimmy. Adam: A Method for Stochastic Optimization. *In:* INTERNATIONAL CONFERENCE FOR LEARNING REPRESENTATIONS, n. 3. 2015. 9 ed, San Diego, 2017. Disponível em: https://arxiv.org/abs/1412.6980. Acesso em: 3 jul. 2022.

KLAPURI, Anssi. Pattern Induction and Matching in Music Signals. *In:* 7TH INTERNATIONAL SYMPOSIUM, n. 6684. 2011, Málaga, Spain: Springer, 2011, p. 188 - 204. Disponível em: https://link.springer.com/chapter/10.1007/978-3-642-23126-1_13. Acesso em: 26 jan. 2020.

LALITHA, S *et al*. Emotion Detection using MFCC and Cepstrum Features. *In:* INTERNATIONAL CONFERENCE ON ECO-FRIENDLY COMPUTING AND COMMUNICATION SYSTEMS. 2015, Bangalore, Karnataka: Procedia Computer Science 70, 2015, p. 29-35.

MAAS, Andrew L; HANNUN, Awni Y; NG, Andrew Y. **Rectifier Nonlinearities Improve Neural Network Acoustic Models**. Stanford University. CA, 2013. 6 p. Disponível em: http://robotics.stanford.edu/. Acesso em: 15 jul. 2022.

MCFEE, Brian *et al*. librosa: Audio and Music Signal Analysis in Python. *In:* PYTHON IN SCIENCE, n. 14. 2015. Proceedings [...] www.academia.edu, 2015, p. 18 - 25. Disponível em: https://www.academia.edu/18862537/librosa_Audio_and_Music_Signal_Analysis_in_Python. Acesso em: 29 jun. 2022.

MCPHERSON, Gary E.; GABRIELSSON, Alf. **From Sound to Sign**. Nova York: Oxford university Press, f. 200, 2002, p. 99-116. (The Science & Psychology of Music Performance: Creative Strategies for Teaching and Learning).

MED, Bohumil. **Teoria da música**. 4 ed. Brasília, DF: Musimed, f. 210, 1996. 420 p.

MILETTO, E. M. *et al*. Introdução à Computação Musical. *In:* IV CONGRESSO BRASILEIRO DE CIÊNCIA DA COMPUTAÇÃO. 2004, Itajaí, SC, 2004. 21 p. Disponível em: https://www.academia.edu/31155836/Introdução_à_computação_musical. Acesso em: 30 nov. 2020.

MIYAO, Hidetoshi; NAKANO, Yasuaki. Head and stem extraction from printed music scores using a neural network approach. **Proceedings of 3rd International Conference on Document Analysis and Recognition**, p. 1074 - 1079, 1995.

PONS, Jordi *et al*. Timbre Analysis of Music Audio Signals with Convolutional Neural Networks. **25th European Signal Processing Conference (EUSIPCO)**, ago 2017. Disponível em:

https://www.researchgate.net/publication/320823288_Timbre_analysis_of_music_audio_signals_with_convolutional_neural_networks. Acesso em: 11 set. 2019.

RAPACCIUOLO, Antonio. **Arban**. Trumpet Exercises. 2006. Disponível em: http://www.trumpetexercises.com/SITE/Mp3/Arban/1/index.html. Acesso em: 7 jul. 2022.

SAZAKI, Yoppy; AYUNI, Rosda; KOM, S. Musical Note Recognition Using Minimum Spanning Tree Algorithm. **IEEE - 2014 8th International Conference on Telecommunication Systems Services and Applications (TSSA)**, 10 2014.

SHAH, Hardik. **Full Overview of Artificial Neural Networks (ANN)**. Learn Hub. 2020. Disponível em: https://learn.g2.com/artificial-neural-network. Acesso em: 15 jul. 2022.

SLIZOVSKAIA, Olga; GÓMEZ, Emilia; HARO, Gloria. Musical Instrument Recognition in User-generated Videos using a Multimodal Convolutional Neural Network Architecture. **Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval - ICMR '17**, 2017.

SOUZA, Renato Manoel de. **Reconhecimento de emoções através da fala utilizando redes neurais**. Florianópolis, 2020. 78 p Trabalho de Conclusão de Curso (Ciências da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2020.

SUITS, B. H.. Physics of Music: Notes*. In:* MICHIGAN TECHNOLOGICAL UNIVERSITY. 1998, Michigan, 1998. Disponível em: https://pages.mtu.edu/~suits/notefreqs.html. Acesso em: 3 jul. 2022.

TAMBOLI, Allabakash Isak; KOKATE, Rajendra D.. An Effective Optimization-Based Neural Network for Musical Note Recognition. **Journal of Intelligent Systems**, v. 28, n. 1, p. 173 - 183, 28 jan. 2019. Disponível em: https://philpapers.org/rec/TAMAEO. Acesso em: 22 ago. 2019.

APÊNDICE A — CÓDIGO FONTE DESENVOLVIDO

Url do repositório do código fonte produzido e base de dados:

https://github.com/matheeeusl/reconhecimentoMusical

APÊNDICE B — ARTIGO SBC SOBRE O TCC

Reconhecimento e Análise Musical de Instrumentos Melódicos com Redes Neurais

Matheus Laureano [1]

[1] Sistema de Informação - Departamento de Informática e Estatística (INE) - Universidade Federal de Santa Catarina (UFSC) - Florianópolis, SC - Brazil

matheuslaureanol1@gmail.com

Abstract. This project presents how neural network technology can help in the process of music recognition and shows the details of a neural network implementation developed with the help of several tools such as the library librosa, TensorFlow, and scikit-learn. The Mel Frequency Cepstral Coefficients (MFCC) extraction method was also used to extract the audio spectrum coefficients creating a feature vector. Using a dataset developed by the author, the project achieved 82% accuracy in recognizing the trained musical notes. This project also presents a comparison with the traditional method of note recognition, which brought a lower hit rate. Despite little difference, the neural network proved to be efficient to perform this knowledge.

Resumo. Esse projeto apresenta como a tecnologia de redes neurais pode auxiliar no processo de reconhecimento musical, mostra os detalhes de uma implementação de rede neural desenvolvida com auxílio de diversas ferramentas como a biblioteca librosa, tensorflow e scikit-learn. Também foi utilizado o método de extração Mel Frequency Cepstral Coefficients (MFCC) para extrair os coeficientes de espectro de áudio criando um vetor de características. Utilizando um conjunto de dados desenvolvido pelo autor, o projeto obteve 82% de acerto no reconhecimento das notas musicais treinadas. Esse projeto também apresenta uma comparação com o método tradicional de reconhecimento de notas, o qual trouxe uma taxa de acerto inferior. Apesar de pouca diferença, a rede neural se demonstrou eficiente para realizar esse conhecimento.

1. Introdução

A música é capaz de comunicar emoções básicas que são reconhecidas sem esforço em adultos, independentemente da formação musical e universalmente em diferentes culturas (HSIEH et al., 2012). O sistema auditivo humano nos dá a capacidade extraordinária de identificar instrumentos sendo tocados (afinados e desafinados) a partir de uma peça musical

e também ouvir o ritmo/melodia do instrumento individual sendo tocado. Esta tarefa parece "automática" para nós, mas provou ser muito difícil de replicar em sistemas computacionais .

Uma maneira para reproduzir essa capacidade analítica em sistemas computacionais é por meio das Redes Neurais Artificiais (RNAs) que conforme Jain, Mao e Mohiuddin (1996), as RNAs são inspiradas por redes neurais biológicas e são sistemas de computação massivamente paralelos que consistem em um número extremamente grande de processadores simples com muitas interconexões. Os modelos de RNA tentam usar alguns princípios “organizacionais” que se acredita serem usados no cérebro humano. A identificação do ritmo/melodia de uma peça musical é parte essencial para o projeto e se espera que utilizando modelos de RNA consiga completar essa tarefa.

A proposta deste trabalho é desenvolver uma comparação entre modelos de análise musical para auxiliar o iniciante a atingir a maneira de execução correta apresentada no repertório por meio de feedbacks na prática individual utilizando RNAs que contribuam na identificação dos padrões musicais comparando com o áudio identificado e retornando informações dos resultados da comparação, para isso é necessário analisar a fundamentação teórica e trabalhos correlatos que se relacionem computação musical, definir experimentos com modelos de redes neurais para o reconhecimento e análise de notas musicais, a partir de uma entrada de áudio, que produzam um *feedback* com erros e acertos.

2. Análise Musical

O som é a sensação produzida no ouvido pelas vibrações de corpos elásticos. Uma vibração põe em movimento o ar na forma de ondas sonoras que se propagam em todas as direções simultaneamente. Estas atingem a membrana do tímpano fazendo-a vibrar. Transformadas em impulsos nervosos, as vibrações são transmitidas ao cérebro que as identifica como tipos diferentes de sons (MED, 1996, p. 11).

A afinação é uma questão central para a prática musical de vários instrumentistas e cantores. É com base nela que todos os instrumentos são padronizados, Freire (2016) explica que a estrutura de um sistema de afinação depende da descrição exata das frequências de cada uma das notas de uma escala e de quais são os critérios e os parâmetros para a construção desta escala. A escolha de uma afinação é essencial para padronizar as notas identificadas, para esse projeto vai ser utilizado como referência A = 440 Hz.

Segundo (MILETTO et al., 2004) a digitalização é o processo de representar numericamente grandezas analógicas (áudio, vídeo e imagem) em computadores, que são máquinas que trabalham na notação binária.

O processo de digitalização do som, é realizado em quatro etapas:

- Filtragem: Limita uma faixa de frequência, onde valores acima da frequência máxima são descartados.
- Amostragem: Faz com que o sinal analógico passe por um circuito amostrador, e deste circuito distribua taxas de amostragens do som a ser digitalizado

- Quantização: Arredonda os valores de nível de amplitude para o valor mais próximo quando ficam entre dois valores digitais, os pulsos gerados na amostragem passam a ser convertidos em números.
- Gravação dos dados obtidos em arquivo.

Mel Frequency Cepstrum (MFC) é a representação linear do cosseno transformado de um espectro de potência logarítmica de curto prazo do sinal de voz em uma escala de frequência Mel não linear. Souza (2020) explica que a extração do MFCC é um método para extrair os coeficientes de um espectro de áudio, permitindo criar um vetor de características que fornecem uma melhor representação do sinal, já que exploram a frequência que os humanos ouvem. Essa etapa de extração para gerar vetores de características é fundamental para que se obtenha sucesso nas outras etapas.

3. Desenvolvimento

O projeto apresenta dois métodos, sendo o primeiro desenvolvido para identificar as notas com base na frequência e o segundo desenvolvido para analisar as notas através de uma rede neural. Os dois projetos utilizaram a mesma base de dados.

3.1 Construção do conjunto de dados

A construção do conjunto de dados consta em recolher as notas musicais gravadas pelo autor e de exercícios musicais publicados online através do site desenvolvido por Rapacciuolo (2006).

Como pré-processamento esses exercícios foram exportados para o Audacity e foram cortadas apenas as notas C, D e E. Após exportar essas notas o pré-processamento dos dois métodos se difere.

A maioria das gravações são dos mesmos exercícios extraídos do livro de método para trompete Arban. Foram selecionados nove exercícios cada um com sua característica específica:

- Staccato: Página 22, exercício 48
- Legato: Página 39, exercício 01
- Notas curtas (0,5 tempo): Página 17, exercício 29
- Notas curtas (0,25 tempo): Página 26, exercício 15 e Página 59, exercício 03
- Notas longas (4 tempos): Página 12, final do exercício 07
- Notas longas (2 tempos): Página 12, exercício 09
- Notas longas (1 tempo): Página 14, exercício 17
- Compasso 6/8: Página 34, exercício 32

Desses exercícios foram extraídos os seguintes rótulos:

- Notas Dó 4 (C4): 43 notas
- Notas Dó 5 (C5): 111 notas
- Notas Ré 4 (D4): 70 notas

- Notas Ré 5 (D5): 111 notas
- Notas Mi 4 (E4): 94 notas
- Notas Mi 5 (E5): 54 notas

Todas essas notas possuem duração diferentes, estão em ritmos diferentes e a única coisa em comum é a frequência. Para separar as notas dos exercícios foi utilizado o programa Audacity. A separação de notas foi feita manualmente selecionando o trecho correto da nota executada no exercício.

3.2 Ferramentas

As ferramentas utilizadas nesse projeto foram o Audacity que foi selecionado para o projeto devido a sua facilidade em cortar e exportar trechos de áudios. Esse programa foi utilizado para isolar as notas musicais.

Librosa que fornece uma interface fácil para extrair certas características e, para utilizá-la, é necessário importar dentro do código e acessar o arquivo musical. Com o Librosa é possível acessar recursos como a extração do MFCC de cada nota musical.

TensorFlow que possui um ecossistema abrangente e flexível de ferramentas, bibliotecas e recursos da comunidade que permite aos pesquisadores levar adiante machine learning e aos desenvolvedores criar e implantar aplicativos com tecnologia de machine learning. Nesse projeto usamos a API do Keras que já implementa alguns dos modelos comumente utilizados, agilizando assim o desenvolvimento.

O PyAudio é uma biblioteca python que permite o acesso do python ao microfone independente do sistema operacional, no projeto ele é utilizado no primeiro método, onde ele grava alguns segundos de música para determinar se qual a nota que o usuário tocou.

Também foi utilizado o Scikit-learn, uma biblioteca python que é desenvolvida especificamente para aplicação prática de machine learning. No projeto é utilizado para separar os dados entre dados de treino e de teste.

Por último foi utilizado a biblioteca python Parselmouth, foi utilizado a ferramenta para sintetizar o som digitalmente e capturar o tom da gravação.

3.3 Extração de características

A etapa da extração de características é fundamental para que se obtenha sucesso em desenvolvimentos que visam fazer uma classificação, principalmente os que utilizam redes neurais. Esse projeto conta com a utilização do método MFCC, que fornece um vetor dos atributos presentes no espectro.

Os valores de entrada para o algoritmo foram definidos conforme documentação da biblioteca e testes realizados:

- **n_mfcc**: número de características retornadas do áudio de entrada
- **n_fft**: tamanho do sinal numa janela de tempo
- **hop_length**: representa o número de amostras entre quadros sucessivos, ex: as colunas de um espectrograma.

Foram testados alguns valores para o tamanho do vetor de características e estes apresentaram uma diferença significativa nos testes realizados, por isso foi optado por utilizar o número de 40 características. Alguns valores foram testados também no tamanho da janela de tempo, e o valor cujo testes foram melhores foi de 2048. No caso das amostras entre quadros sucessivos, não foram apresentadas diferenças significativas nos experimentos, mantendo assim o valor de 512 definido como padrão pela documentação.

3.4 Arquitetura e Resultados do Modelo 1: Análise de Frequência

O primeiro modelo utiliza um método muito similar aos afinadores digitais hoje em dia, o objetivo dele é detectar a frequência do som e com base nela informar qual é a nota tocada.

A tabela a seguir mostra quais foram as frequências usadas para representar cada nota, foi utilizado como base a tabela de notas publicada por Suits (1998).

Tabela 1 — Frequências

Nota Real	Nota Trompete	Frequência Ideal	Largura de banda
Bb3	C4	233.08 Hz	226.54 Hz - 240.01 Hz
Bb4	C5	466.16 Hz	453.08 Hz - 480.02 Hz
C4	D4	261.63 Hz	254.285 Hz - 269.405 Hz
C5	D5	523.25 Hz	508.565 Hz - 538.81 Hz
D4	E4	293.66 Hz	285.42 Hz - 302.395 Hz
D5	E5	587.33 Hz	570.85 Hz - 604.79 Hz

A tabela mostra por exemplo que se o som apresentar uma frequência de 230Hz ele deve retornar um texto visual mostrando que a nota é Bb3 (ou C4 para o trompete).

Nesse modelo foi utilizado também um retorno textual que apresenta se a nota está afinada ou não, para isso é verificado o quão próximo da frequência ideal a nota tocada se encontra, sendo que o mais próximo representa mais afinado.

Sendo assim, foi necessário fazer um calcular a acurácia de cada nota seguindo o seguinte cálculo:

$$AcuraciaIndividual = \frac{Notas_{corretas} \cdot 100}{Notas_{total}}$$

Após o cálculo de cada acurácia individual, para avaliar a acurácia do modelo completo foi-se utilizado da seguinte fórmula:

$$AcuraciaTotal = \frac{\sum AcuraciaIndividual}{QuantidadeNotas}$$

O resultado desse cálculo foi um total aproximado de 79.72%. Essa acurácia um pouco baixa pode se dar por diversos fatores, um deles é a quantidade de ruídos presente nos arquivos que alteram a frequência dos arquivos.

3.5 Arquitetura e Resultados do Modelo 2: Rede Neural

A tabela abaixo apresenta as camadas da rede neural desenvolvida. A escolha de cada uma foi com base em testes realizados e obtiveram a melhor precisão.

Tabela 2 - Arquitetura da Rede Neural

Camada	Tipo	Neurônios	Ativação	Regularizador	Dropout
Entrada	Flatten	-	-	-	-
1	Dense	1024	ReLu	L2 - 0.001	0.3
2	Dense	512	ReLu	L2 - 0.001	0.3
3	Dense	256	ReLu	L2 - 0.001	0.3
4	Dense	64	ReLu	L2 - 0.001	0.3
Saída	Dense	12	Softmax	-	-

Os modelos foram construídos utilizando camadas do tipo Dense, que é um tipo padrão para camadas densamente conectadas. A camada de entrada é do tipo Flatten e tem como argumento o (input_shape) que é do uma tupla de inteiros representando o formato de dado analisado.

As outras camadas utilizam ReLU como função de ativação que é o Rectified Linear Unit, essa função de ativação vai retornar diretamente o valor se for positivo e caso não for, vai retornar zero. Geralmente ela possui melhor desempenho, possui uma boa convergência e reduz a probabilidade de gradiente de fuga.

Também foram adicionados regularizadores nas camadas com o intuito de aplicar penalidades para reduzir a complexidade da rede neural e reduzir o overfitting (ou sobreajuste). O regularizador escolhido foi o L2 (ou queda de peso), que resumidamente transforma os pesos um pouco menores a cada vez que uma atualização ocorre.

Com o intuito de reduzir o overfitting foi adicionado dropout em cada camada, este transforma um input aleatório em 0 a cada atualização durante a fase de treino, diminuindo a complexidade da rede e reduzindo o overfitting.

A última camada possui softmax como função de ativação, essa função tem como objetivo normalizar a saída dos valores da rede neural.

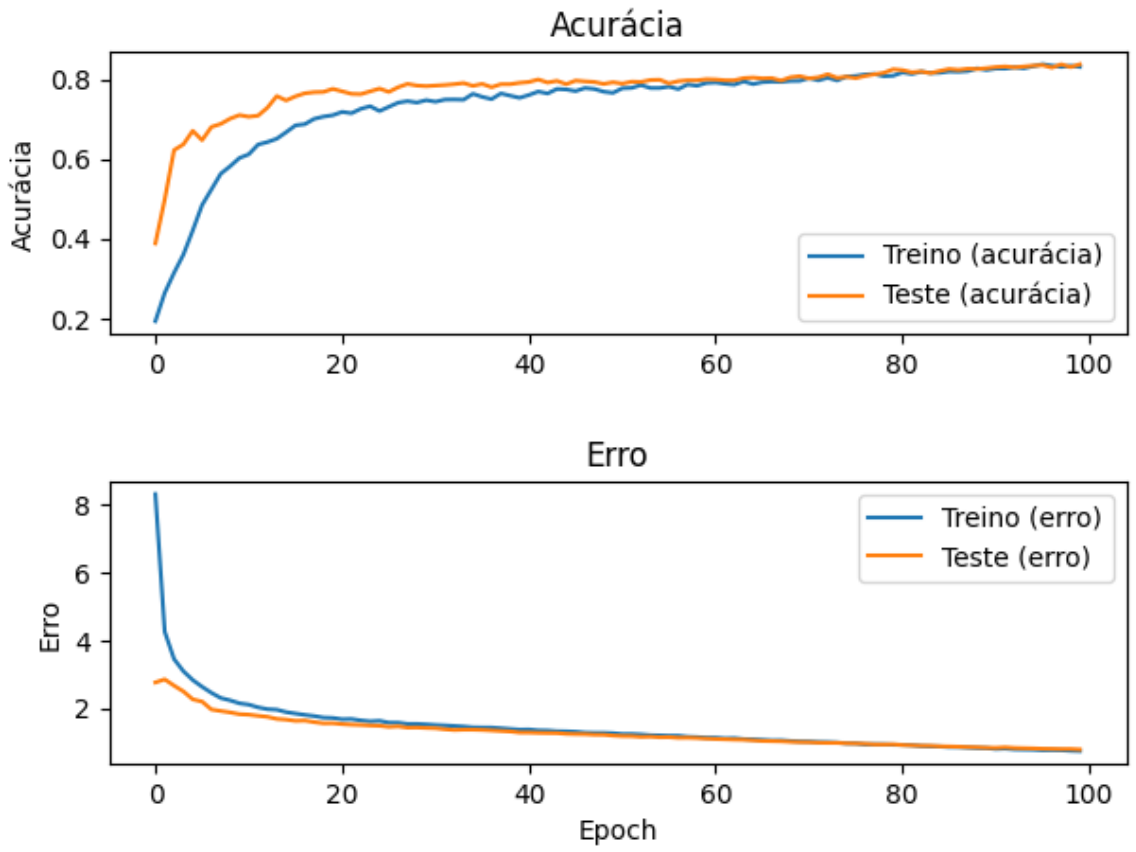
Existem alguns algoritmos que otimizam a conversão do modelo. O modelo foi otimizado usando o algoritmo chamado Adam que vai minimizar a função de perda em relação aos parâmetros da rede, de acordo com Kinga e Ba (2017) Adam é um método computacionalmente eficiente, tem pouca necessidade de memória, invariante ao reescalonamento diagonal de gradientes e é adequado para problemas grandes em termos de dados/parâmetros.

As métricas de perda tem como objetivo calcular a quantidade que um modelo deve procurar minimizar durante o treinamento, para o modelo foi utilizado a métrica probabilística chamada Sparse Categorical Crossentropy que calcula a perda de entropia cruzada entre os rótulos e as previsões, é recomendado utilizar essa função de perda de entropia cruzada quando houver duas ou mais classes de rótulos.

Epoch é o número de vezes que os dados passam por todas as camadas de rede. O número atribuído para o modelo foi de 100.

Ao final da execução do programa temos como resultado o seguinte gráfico:

Gráfico 1 — Rede Neural: Resultado



Na horizontal vemos os Epochs que representam cada iteração que os dados passaram pela rede neural, na vertical vemos o número de porcentagem de cada situação, vemos que a acurácia da rede neural chegou em torno dos 80%.

4. Conclusão

O presente trabalho buscou apresentar um modelo de análise e reconhecimento musical através de redes neurais, com uma pesquisa na literatura com foco nas diversas maneiras de identificação de som através de diferentes redes neurais.

Por meio da revisão bibliográfica, foram identificados diferentes tipos de modelos com capacidade para análise musical. Diferente do tipo de rede neural, a técnica MFCC para extração de características é utilizada para captar não apenas os sons musicais mas também a intenção por trás dos sons.

No começo do trabalho tinha-se o objetivo de analisar e reconhecer partituras completas de exercícios musicais de diferentes instrumentos, identificar afinação, tempo e dinâmica. Porém, devido a falta de dados para fazer essa análise com mais precisão, foi decidido analisar notas musicais separadas como afinação, o trompete como instrumento melódico e foi decidido fazer uma comparação com o método tradicional de afinação.

Tendo em vista a acurácia como ponto principal da avaliação, ao decorrer do trabalho foi identificado a importância do tamanho do conjunto de dados para treinamento e validação, e como influenciavam na precisão do modelo. Em testes iniciais foi verificado que os dados de treino eram superiores e distantes dos dados de teste, foram então aplicadas técnicas para resolução de overfitting, diminuindo a distância entre teste e treino. Além disso, o desempenho da rede neural melhorou significativamente após o aumento de características através do MFCC.

Com os resultados apresentados nesse projeto, percebe-se que mesmo executando a rede neural com um aumento significativo de Epochs, a tendência é que a acurácia se mantenha por volta dos resultados obtidos, isso se dá pela quantidade de dados. Com uma quantidade maior de dados há a possibilidade de identificar com mais precisão as notas executadas porém, caso haja uma variedade de instrumentos musicais de diferentes tipos, existe o risco de diminuir a acurácia obtida.

Referências

HSIEH, S. *et al*. Brain correlates of musical and facial emotion recognition: Evidence from the dementias. Neuropsychologia. 9 p. 2012.

JAIN, Anil K.; MAO, Jianchang; MOHIUDDIN, K. M.. Artificial Neural Networks: A Tutorial. IEEE Computer, v. 29, p. 31-44, 1996.

MED, Bohumil. Teoria da música. 4 ed. Brasília, DF: Musimed, f. 210, 1996. 420 p.

FREIRE, Ricardo Dourado. Como será que eu afino? A relação entre sistemas de afinação e parâmetros de afinação na performance musical. Revista Música Hodia, Goiânia, v. 16, n. 2, p. 133-144, 2016. Disponível em: https://revistas.ufg.br/musica/article/view/45333. Acesso em: 8 jul. 2022.

MILETTO, E. M. et al. Introdução à Computação Musical. In: IV CONGRESSO BRASILEIRO DE CIÊNCIA DA COMPUTAÇÃO. 2004, Itajaí, SC, 2004. 21 p. Disponível em: https://www.academia.edu/31155836/Introdução_à_computação_musical. Acesso em: 30 nov. 2020.

SOUZA, Renato Manoel de. Reconhecimento de emoções através da fala utilizando redes neurais. Florianópolis, 2020. 78 p Trabalho de Conclusão de Curso (Ciências da Computação) - Universidade Federal de Santa Catarina, Florianópolis, 2020.

RAPACCIUOLO, Antonio. Arban. Trumpet Exercises. 2006. Disponível em: http://www.trumpetexercises.com/SITE/Mp3/Arban/1/index.html. Acesso em: 7 jul. 2022.

SUITS, B. H.. Physics of Music: Notes. *In:* MICHIGAN TECHNOLOGICAL UNIVERSITY. 1998, Michigan, 1998. Disponível em: https://pages.mtu.edu/~suits/notefreqs.html. Acesso em: 3 jul. 2022.