UNIVERSIDADE UNICESUMAR
CENTRO DE CIÊNCIAS EXATAS TECNOLÓGICAS E AGRÁRIAS
CURSO DE GRADUAÇÃO EM **ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

APRENDIZAGEM PROFUNDA APLICADA NA CLASSIFICAÇÃO DE TAIKOS

Eduardo Haruo Fuzita Nagata

MARINGÁ – PR
2020

Eduardo Haruo Fuzita Nagata

APRENDIZAGEM PROFUNDA APLICADA NA CLASSIFICAÇÃO DE TAIKOS

Artigo apresentado ao Curso de Graduação em Engenharia de controle e automação da Universidade UNICESUMAR como requisito parcial para a obtenção do título de Bacharel(a) em Engenharia de Controle e Automação, sob a orientação do Prof. Msc Anderson Rodrigues.

MARINGÁ – PR

2020

FOLHA DE APROVAÇÃO

Eduardo Haruo Fuzita Nagata

APRENDIZAGEM PROFUNDA APLICADA NA CLASSIFICAÇÃO DE TAIKOS

Artigo apresentado ao Curso de Graduação em Engenharia De Controle E Automação da Universidade UNICESUMAR como requisito parcial para a obtenção do título de Bacharel(a) em Engenharia de Controle e Automação, sob a orientação do Prof. Msc Anderson Rodrigues.

Aprovado em: 12 de Novembro de 2020.

BANCA EXAMINADORA

Prof.(a): Msc. Anderson Rodrigues

Prof. (a): Msc. Murilo Magan

Prof. (a): Dra. Sonia Tomie Tanimoto

APRENDIZAGEM PROFUNDA APLICADA NA CLASSIFICAÇÃO DE TAIKOS

Eduardo Haruo Fuzita Nagata

RESUMO

O *deep learning* ou aprendizagem profunda é uma vertente da *machine learning* ou aprendizagem da máquina, por isso possui como função possibilitar a uma máquina, a partir de dados recebidos, a resolução de problemas designados sem a necessidade de constante supervisão e interferência humana. O diferencial da aprendizagem profunda é a sua capacidade de dividir um problema complexo em várias camadas que subdivididas de forma simples, facilitam a análise, classificação e solução. Levando-se em conta esse aspecto da aprendizagem profunda a proposta para este trabalho é aplicá-la no reconhecimento de instrumentos musicais de percussão, a partir das ondas sonoras emitidas. De forma simplificada, serão gravadas faixas de áudios dos instrumentos, para o treinamento do sistema, os algoritmos após treinamento serão capazes de fazer a identificação das particularidades de cada instrumento e batida e assim classificá-los. Os instrumentos a serem utilizados são as variantes dos taikos, os tambores japoneses, a princípio os tipos considerados são o shime daiko, okedo daiko, hira-daiko e nagado-daiko. O pretendido é a construção da rede neural e o tratamento do sinal utilizando a linguagem de programação *python*.

Palavras-chave: Aprendizado da Máquina. Redes Neurais. Taiko.

DEEP LEARNING APPLIED ON TAIKOS CLASSIFICATION

ABSTRACT

Deep learning is a part of machine learning, it has the function of enabling a machine from received data, the resolution of designated problems without the need of constant supervision and human intervention. The differential of deep learning is its ability to divide a complex problem into several layers that subdivided in a simple way, facilitate the analysis, classification and solution. Taking this aspect of deep learning into account, the proposal for this assignment is to apply it in the recognition of percussion musical instruments, the type and intensity of percussion being applied, from the sound waves emitted by them, in a simplified way, audio tracks of the instruments will be recorded, for system training, the algorithms after training will be able to identify the particularities of each instrument and beat and thus classify them. The instruments to be used are the variants of taikos, the Japanese drums, at first the types considered are the shime daiko, okedo daiko, hira-daiko and nagado-daiko. The aim is the construction of the neural network and the treatment of the signal using the python programming language.

Keyword: Machine Learning. Neural Network. Taiko.

1. INTRODUÇÃO

O tema principal deste trabalho é a aprendizagem profunda ou no original *deep learning* (DL), que consiste em uma forma de aprendizado da máquina, no original *machine learning* (ML), e portanto, uma área da inteligência artificial ou IA. Assim o DL carrega as premissas e objetivos da IA, "ramo da ciência da computação que se ocupa da automatização do comportamento inteligente" (LUGER, 2013, p.1), o ramo da IA busca projetar uma máquina dotada da capacidade de aprender e se adaptar de forma autônoma (LUGER, 2013). Possui ainda as particularidades do ML que combina a inteligência da máquina com a utilização de um extenso banco de dados a fim de desenvolver no sistema a capacidade associativa e comparativa, capaz de prever os resultados para futuras situações (ALPAYDIN, 2010); o fator negativo do ML é a sua necessidade de um especialista na área abordada para fazer o pré-processamento dos dados (PEREIRA, 2017), o que acaba por consumir mais tempo, requer a presença de um número maior de especialistas, que vai de acordo com as características relevantes para determinada função. O DL não necessita do pré-processamento de dados, além de possuir poder de processamento significativamente maior.

A DL utiliza para seu aprendizado um banco de dados e possui sua estrutura baseada em uma rede neural multicamadas. Esta subcategoria do ML necessita de um volume maior de dados, porém como foi dito, sua capacidade de processamento é muito maior, sendo capaz de lidar com mais facilidade com problemas complexos e com um grande volume de características a serem consideradas, o que vem a ser um problema para a ML convencional (SANTANA, 2018). Outra vantagem em relação a ML é a quantidade máxima de informações que consegue extrair dos dados, considerando que a DL é projetada para lidar com um volume maior de dados (SANTANA, 2018).

Warren McCulloch e Walter Pitts em 1943, formularam seu modelo de neurônio computacional cuja proposta era resolver quaisquer funções lógicas (LUGER, 2013). McCulloch e Pitts influenciaram o surgimento do modelo de neurônio *perceptron* posteriormente desenvolvido por Frank Rosenblatt nos anos de 1950 e 1960 com funcionalidade aprimorada e passível de ser estruturado a fim de aprender sem a intervenção de um programador, no entanto as saídas desses neurônios continuavam 0 ou 1, empecilho resolvido com o desenvolvimento do neurônio sigmoidal, o qual era capaz de gerar saídas cujos valores podem abranger todos os valores entre 0 e 1 (NIELSEN, 2015). Com relação à

estrutura das redes neurais, é feita a divisão em camada de entrada, onde são inseridos os dados; camada oculta, responsável pelo processamento dos dados e camada de saída, onde a resposta da máquina ao problema é concebida. Como já mencionado, o DL utiliza redes neurais multicamadas, ou seja, formadas por mais de uma camada oculta, isso ocorre por necessidade; o grau de complexidade das decisões tomadas se elevam a cada camada (NIELSEN, 2015). Sendo assim, para uma rede neural conseguir lidar com situações mais sofisticadas no nível daquelas em que são empregadas, o DL demanda um número elevado de camadas ocultas.

O DL vem ganhando o seu espaço, principalmente na atual realidade, tudo é digital; portanto, pode ser automatizado, já que a todo momento circulam volumes imensos de dados. O DL é comumente utilizado em sistemas para detecção de spam, recomendação de produtos a consumidores, compreensão da linguagem natural e reconhecimento de imagens e sons.

Este trabalho tem como objetivo utilizar os recursos do DL na identificação dos tipos de taikos, a partir do estímulo sonoro emitido pelo instrumento, a fim de auxiliar no aprendizado do instrumento, assim como auxiliar nos ensaios de grupos e na identificação de erros.

2. DESENVOLVIMENTO

2.1 Taikos:

Segundo Izumi (2010) o taiko ou na tradução tambor grande, possui profunda ligação com a história japonesa, possuindo uso de caráter religioso e ritualístico, foram usados inicialmente em festivais típicos do país, com a finalidade de agradecer boas colheitas ou afastar as coisas ruins. No entanto após a Segunda Guerra Mundial os instrumentos passaram a ser utilizados para músicas e performances populares sem o caráter espiritual.

De acordo com o site KaminaritaikoRJ (2012) os taikos são constituídos de um corpo e couros especiais; o corpo é formado por uma madeira nobre oca esculpida de forma artesanal.

Como já estabelecido os 3 instrumentos alvos do estudo foram Nagado-daiko, Okedo-daiko e Shime-daiko; a seguir algumas informações sobre os instrumentos segundo o site Ishindaiko (2012?):

- **Nagado-daiko:** Corpo feito na madeira escavada, o couro é preso a partir de tachas e sua afinação ocorre apenas durante sua fabricação.

Figura 1 – Nagado-daiko

Fonte: http://www.ishindaiko.com.br/index.php/o-que-e-taiko?limitstart=0

- **Okedo-daiko:** Formado por um corpo de madeira compensada, duas tiras de couro costuradas a aros de metal e cordas entrelaçadas nos aros que fixam as tiras de couro ao corpo e permitem afinar o instrumento.

Figura 2 – Okedo-daiko

Fonte: http://www.ishindaiko.com.br/index.php/o-que-e-taiko?limitstart=0

- **Shime-daiko:** Possui dimensões menores que os instrumentos anteriores e pode ser formado por duas tiras de couro especial, corpo e cordas ou pode utilizar um sistema de porcas ou para fusos no lugar das cordas.

Figura 3 – Shime-daiko
Fonte: http://www.ishindaiko.com.br/index.php/o-que-e-taiko?limitstart=0

2.2 Inteligência artificial:

Inteligência artificial ou IA, conforme Luger (2013, p 01), "pode ser definida como o ramo da ciência da computação que se ocupa da automação do comportamento inteligente". Segundo Ciriaco (2008), "Inteligência Artificial (IA) é um ramo da ciência da computação que se propõe a elaborar dispositivos que simulem a capacidade humana de raciocinar, perceber, tomar decisões e resolver problemas, enfim, a capacidade de ser inteligente."

A origem do termo inteligência artificial é atribuída a John McCarthy, que em conjunto com Marvin L. Minsky, Nathaniel Rochester e Claude E. Shannon, elaboraram uma proposta para a realização de um estudo sobre a inteligência artificial em 1956 na universidade de Darthmouth. O grupo propunha para o evento a discussão de temas diversos pertinentes à inteligência artificial, dentre eles havia sete que se destacavam, que se tornaram a base para a inteligência artificial: a automação de computadores, como um computador poderia ser programado para utilizar uma linguagem, redes neurais, teoria do tamanho do cálculo, auto aprimoramento, abstração e aleatoriedade e criatividade (LUGER, 2013).

A inteligência artificial possui um campo de atuação muito vasto e diversificado, Luger (2013) destacou 10 campos de atuação como sendo os mais relevantes:

- Área de jogos - foi utilizada durante as etapas iniciais para busca de espaço de estados, por possuírem regras bem estruturadas que limitam o número de possibilidades. Essa área também se mostrava interessante devido a facilidade de se representar os tabuleiros.
- A área de raciocínio automatizado e prova de teoremas – mostra-se bastante produtiva uma vez que contribuiu para a elaboração de algoritmo de buscas e a linguagem de programação prolog, isso se deve ao fato dos teoremas terem embasamento lógico e, portanto, serem formalizados, não oferecendo barreiras para o processo de sua atualização. Em contrapartida, para situações lógicas complexas, tem-se a formação de uma grande amostra de teoremas, muito irrelevantes para a resolução, os provadores iniciais realizavam a prova um a um dos teoremas, sendo pouco eficientes para essas situações, porém o avanço nessa área fez surgir provadores automáticos capazes de reduzir a complexidade desses sistemas.
- A área de sistemas especialistas - fruto da colaboração entre um especialista na área abordada que fornece seus conhecimentos e experiências e um especialista na área da inteligência artificial que faz a implementação em um programa inteligente, o foco para situações onde é necessário um nível de conhecimento específico é justificado por possuir passos ou estratégias para a sua resolução bem definidos, evitando, assim, que o espaço de estados seja limitado, evitando também ambiguidades.
- Compreensão da linguagem natural e modelagem semântica - possui como principal foco a compreensão e geração da linguagem humana; objetiva o desenvolvimento de um programa que além do domínio das propriedades gramaticais e o significado das palavras seja capaz de compreender o contexto e as ambiguidades.
- Modelando o desempenho humano - objetiva a elaboração de inteligências artificiais que utilizem a mente humana como base, emulando o processo humano de pensar, muitas vezes mostra-se menos eficiente que as inteligências que não seguem a formatação humana, porém possui grandes contribuições para a área de estudo da cognição humana.

- Planejamento e robótica - tem como objetivo projetar robôs capazes de realizar suas ações adaptando-se ao meio em que estão inseridos, sendo assim capazes de lidar com obstáculos. O maior entrave é a extensa possibilidade de movimentos e trajetórias possíveis de serem executados.
- Linguagem e ambientes para IA - conforme o grau de complexidade ascendia proporcionalmente aos avanços, os programadores desenvolveram recursos mais específicos e voltados para a organização e simplificação dos algoritmos, como a programação orientada à objetos e às linguagens de programação lisp e prolog.
- Aprendizado de máquina - objetiva o desenvolvimento de máquinas capazes de aprender a partir de treinamento por exemplo, associação ou a partir de experiência própria.
- Representações alternativas - redes neurais e algoritmos genéticos – Faz um paralelo com conceitos de neurônios e genética. As redes neurais consistem em unidades individuais que desempenham diferentes funções, porém estão interconectadas, e cujos finais das trajetórias resultam nas possíveis soluções para a situação aplicada, compondo uma relação que muito se assemelha aos neurônios. Os algoritmos genéticos utilizam os resultados anteriores para, a partir deles, realizar mutações ou recombinações das características prévias para assim construir novas gerações cada vez mais próximas de alcançar o resultado ideal.
- IA e filosofia - aborda o desenvolvimento de um sistema provido de inteligência assim como discussões filosóficas a respeito deste assunto e das demais áreas presentes na inteligência artificial.

2.3 Redes Neurais:

O surgimento das redes neurais artificiais ocorreu nos anos de 1943 a 1958 com as contribuições de McCulloch e Pitts (1943), que trouxeram a ideia de redes neurais para sistemas computacionais. Hebb (1949) formulou as primeiras regras para aprendizagem auto-organizada, além de Resenblatt (1958) que propôs o perceptron, (HAYKIN, 2009).

Para a engenharia, a área das redes neurais artificiais é motivo de um crescente interesse não por possibilitar uma maior compreensão do cérebro humano, mas por possibilitar a implementação de funções tais como reconhecimento de fala, visão e aprendizado em máquinas (ALPAYDIN, 2010).

O Neurônio Perceptron É a forma mais simples de rede neural utilizada em classificação de padrões possíveis de serem separados de forma linear. O perceptron em sua forma mais básica consiste em um neurônio com pesos sinápticos e bias ajustáveis e possível de receber um número de entradas de acordo com a aplicação. Quando construído em torno de um neurônio é limitado a realizar a classificação de padrões envolvendo apenas duas classes (HAYKIN, 2009).

De acordo com Chandra (2018) o perceptron é representado pela equação:

$$y(u)\ onde\ u = \sum_{i=1}^{n} \omega_i * x_i \quad (1)$$

Na qual x são as entradas da rede, wi é o peso associado à i-ésima entrada, b representa a bias, g(u) é a função de ativação e u é o potencial de ativação

Os valores da função de ativação podem ser do tipo degrau ou degrau bipolar, sendo que no primeiro o valor é 0 ou 1 enquanto no segundo é -1 ou 1, de acordo com Silva, Spatti e Flauzino(2016).

O neurônio perceptron possui como desvantagem o fato de que pequenas alterações nos pesos sinápticos ou nas bias podem alterar erroneamente a saída, causando empecilhos na aprendizagem. O neurônio sigmoidal por sua vez não possui este entrave, apesar de possuir estrutura semelhante com entradas, saída, pesos sinápticos e bias, sua representação matemática o torna capaz de assumir todos os valores entre 0 a 1, fazendo com que pequenas alterações nos pesos ou na bias acarretem pequenas alterações na saída (NIELSEN, 2015).

De acordo com Nielsen (2015) o neurônio sigmoidal é descrito pela equação:

$$\frac{1}{1 + e(-\sum_j w_j x_j - b)} \quad (2)$$

2.4 Estrutura de Rede Neural:

A seguir apresento brevemente as principais estruturas de redes neurais.

- **Single-Layer Feedforward:** Composta por uma camada de entrada e uma camada de saída de neurônios que efetuam a classificação das entradas, as saídas dos neurônios alimentam os neurônios a sua direta, assim sucessivamente até encontrar a saída, sem nenhuma forma de realimentação, isso representa o feedforward, (HAYKIN, 2009).

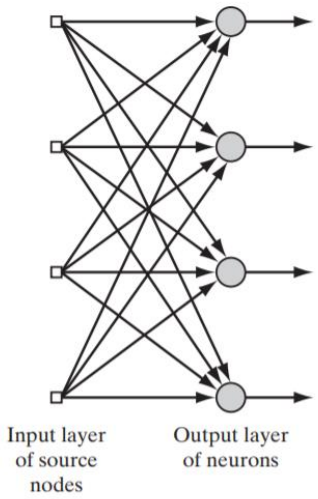

Figura 4 - Single-Layer Feedforward
Fonte: Haykin (2009)

- **Multilayer FeedForward:** É também conhecido popularmente como Perceptron multicamadas (Multilayer perceptron), apesar de serem compostos por neurônios sigmoidais, conforme explica Nielsen (2015). Este modelo possui as camadas ocultas (***hidden layers***), aquelas que se localizam entre as camadas de entrada e saída, podendo conter uma ou mais delas. A presença dessas camadas possibilita a rede extrair estatísticas de maior ordem a partir das entradas (HAYKIN, 2009).

  Esta configuração não possui a limitação do modelo anterior ou do perceptron isolados, sendo capaz de efetuar a classificação de padrões não lineares e regressão de funções não lineares (ALPAYDIN, 2009).

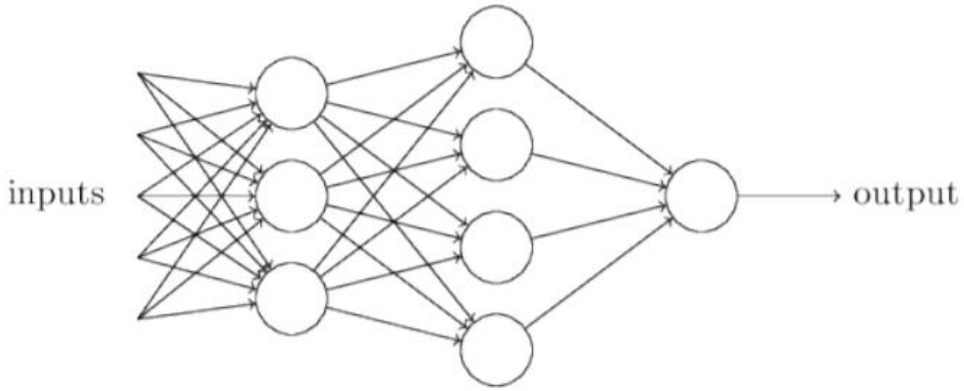

Figura 5 – Estrutura Multilayer FeedForward

Fonte: Nielsen (2015, p.4)

- **Recurrent Networks:** se diferencia dos modelos anteriores por possibilitar realimentação, que pode ocorrer de diversas formas; a saída pode realimentar o próprio neurônio da qual se originou ou não, assim como pode realimentar todos os outros neurônios. Este modelo pode ser aplicado para modelos de uma camada ou de múltiplas camadas, sendo que neste último a realimentação pode se originar tanto nas camadas ocultas quanto na camada de saída (HAYKIN, 2009).

  Após os treinos esta configuração produz um modelo de dados dinâmico que se molda ao longo do tempo e é capaz de realizar classificações com maior precisão, dependendo do contexto dos exemplos fornecidos, de acordo com Deep Learning Book (2015?)

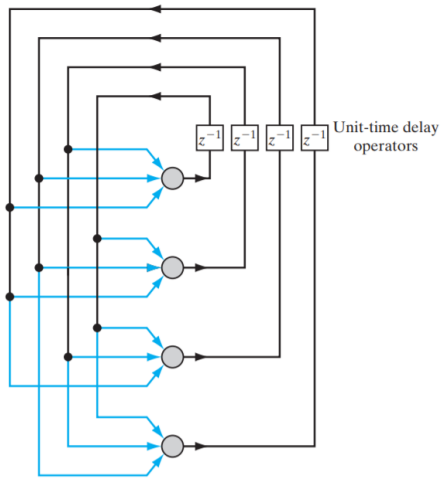

Figura 6 – Exemplo de Recurrent Network em rede com uma camada, sem auto realimentação e realimentação das saídas nas entradas dos demais neurônios

Fonte: Haykin (2009, p.23)

- **Redes Neurais Convolucionais:** Utilizada para classificação e detecção de objetos, nesta arquitetura cada camada é tridimensional, possuindo assim extensão espacial e profunda que corresponde ao número de características. Na camada de entrada essas características correspondem aos canais de cores, enquanto nas camadas ocultas correspondem aos mapas de recursos ocultos que codificam diversos tipos de formas na imagem. Quanto às camadas ocultas, as de mais baixo nível capturam linhas ou outras formas mais básicas enquanto as camadas de mais alto nível capturam formas mais complexas (AGGARWAL, 2018).

  Conforme Nielsen (2015) as redes convolucionais utilizam três ideias principais:

  Local receptive fields*:* imagens ou objetos captados são divididos em pixels, cada pixel corresponde a uma entrada para a rede, essa região da imagem na entrada é o *local receptive field* para os neurônios ocultos. Os neurônios ocultos analisam e capturam as características de pequenas porções do campo receptivo, quando outros neurônios terminam, o campo sofre um pequeno deslocamento e assim outro neurônio inicia analisando um novo espaço e parte do espaço analisado pelo neurônio anterior.

  Shared weights and biases: todos os neurônios ocultos de uma mesma camada compartilham os pesos sinápticos e bias, por isso todos eles analisam a mesma

característica, porém uma camada oculta pode analisar múltiplas características possuindo vários mapas de recursos, cada um possui seus pesos e bias compartilhados e nesses casos essa camada passa a ser chamada de camada convolucional.

Pooling layers: Camadas que também podem estar presentes nesta estrutura e possuem a função de simplificar as saídas das camadas convolucionais.

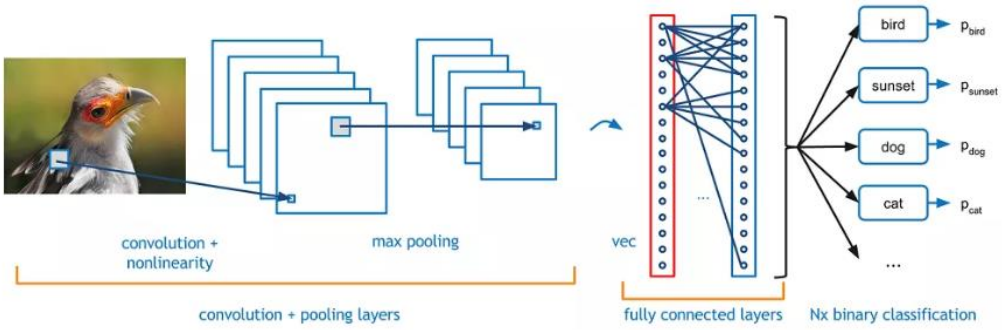

Figura 7 – Rede neural convolucional

Fonte: http://deeplearningbook.com.br/as-10-principais-arquiteturas-de-redes-neurais/

2.5 Aprendizado da máquina:

Iniciando com a definição, “algoritmo é uma sequência de instruções que devem ser executadas para se transformar as entradas em saídas” (ALPAYDIN, 2010, p. 1, tradução nossa)[1]. Para Alpaydin(2010) a aprendizagem da máquina corresponde a todo o processo para se encontrar o algoritmo capaz de resolver o problema, afirmando a necessidade de um banco de dados e uma inteligência por parte do sistema, de modo que ele deve ser capaz de aprender e se adaptar a mudanças. Shalev-Shwartz e Ben-David(2014) firmam a necessidade de que o aprendizado deve ir além da memorização e atuação em situações idênticas, é necessário o desenvolvimento de um raciocínio indutivo, ou seja, saber identificar padrões e prever o resultado.

[1]“An algorithm is a sequence of instructions that should be carried out to transform the input to output”

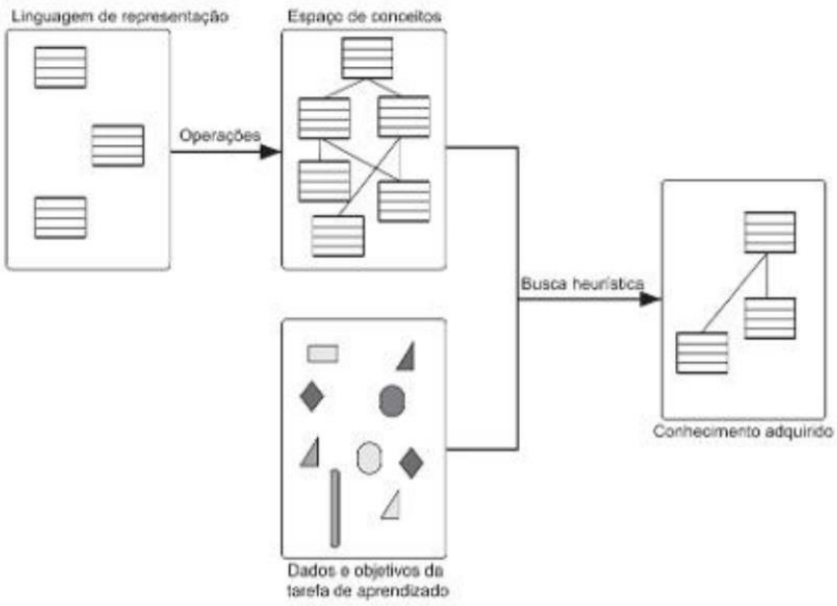

Figura 8 – Modelo geral do processo de aprendizagem

Fonte: Luger (2013, p.324)

Outro aspecto levantado por Luger (2013) é a relevância que o conhecimento prévio tem no aprendizado da máquina, são apresentados dois argumentos concretizando essa ideia.

> Um argumento a favor da importância do conhecimento no aprendizado é que técnicas de aprendizado baseadas em similaridade dependem de quantidades relativamente grandes de dados de treinamento. [...] Outro argumento para a importância de conhecimento prévio é o reconhecimento de que qualquer conjunto de exemplos de treinamento pode dar suporte a um número ilimitado de generalizações, a maioria das quais é irrelevante ou não faz sentido. (LUGER, 2013, p.365)

A forma mais comum de classificar o aprendizado da máquina é em supervisionado, não supervisionado e aprendizagem por reforço.

- **Aprendizagem Supervisionada:** Necessitam de um vetor de entrada e um vetor de saída, sendo este último tido como referência. Após aplicar o vetor de entrada e rodar a aplicação, o resultado obtido é comparado com o resultado referência, assim é calculado o erro, que é alimentado na rede neural e os pesos de cada neurônio são alterados, de modo que a cada aplicação a rede aprende e chega cada vez mais próximo do valor desejado (PEREZ, 2017).
- **Aprendizagem Não Supervisionada:** Não possui um vetor de saída para ter como referência de dados então o próprio algoritmo define o valor de qualidade de representação que deverá aprender (HAYKIN, 2009). Os parâmetros livres da rede

se ajustam em volta do valor objetivando fornecer resultados concisos com relação à entrada (PEREZ, 2017).

Para Honda, Facure, Yaohao (2017) o funcionamento desta metodologia se dá a partir da identificação e agrupamento de padrões dos elementos envolvidos.

- **Aprendizagem Por Reforço:** Assim como o modelo anterior, não possui um parâmetro de saída definido, o sistema aprende a partir da interação com o ambiente que fornece estímulos à rede (HAYKIN, 2009). A aprendizagem ocorre a partir de tentativa e erro, cabe ao ambiente observar o rumo do processo de aprendizagem e fornecer eventuais recompensas conforme o progresso (MAINI, 2017).

2.6 Linguagem De Programação Python:

A linguagem de programação python surgiu em 1989 e possui como principal idealizador Guido Van Rossum e é reconhecida como sucessora da linguagem de programação ABC. Nos dias atuais é gerenciada pela Python Software Foundation, que mantém a licença de código aberto, tornando a linguagem acessível (VAZ, 2018), além de que a linguagem python possui compatibilidade com uma vasta gama de sistemas operacionais, tais como Linux, FreeBSD, Microsoft Windows e Mac OS X. Outros motivos que levaram à adoção do python para o projeto são o fato de ser uma linguagem de fácil legibilidade o que torna tranquila a compreensão de códigos úteis e do próprio projeto, além de simplificar o processo de identificação dos erros e refinamento do código; possuir uma vasta biblioteca aberta criada pela própria comunidade eliminando a necessidade de escrever determinadas funções do zero, segundo Menezes (2014).

Segundo Menezes (2014), a linguagem python vem ganhando bastante destaque atualmente, algumas das áreas em que se mostra presente são a inteligência artificial, banco de dados, animação 3D, a área de aplicativos de celulares, biotecnologias, jogos e desenvolvimento web.

```python
def jogo_da_velha1():
    g1=0
    g2=0
    c=0
    T = [[' ',' ',' '],[' ',' ',' '],[' ',' ',' ']]

    for linha in T:
        print(linha)
    jogador = 1

    while g1==0 and g2==0:
        print('vez do jogador', jogador)
        jogador=1 if jogador==2 else 2
        peça = 'x' if jogador==2 else 'o'
        Q=int(input('digite um quadrado:'))
        if T[int((Q-1)/3)][(Q+2)%3] == ' ':
            T[int((Q-1)/3)][(Q+2)%3] = peça

        else:
            print('a celula ja esta em uso')

        for linha in T:
            print(linha)

        for linha in T:
            for celula in linha:
                if all(celula == 'x' for celula in linha):
                    g1+=1
                elif all(celula == 'o' for celula in linha):
                    g2+=1
                elif T[0][0] == 'x' and T[1][0] == 'x'and T[2][0] =='x':
                    g1+=1
                elif T[0][1] == 'x' and T[1][1] == 'x'and T[2][2] =='x':
                    g1+=1
                elif T[0][2] == 'x' and T[1][2] == 'x'and T[2][2] =='x':
```

Figura 9 – Exemplo de código em python

Fonte: Próprio autor

2.7 Trabalhos Semelhantes:

Alguns trabalhos com assuntos similares encontrados:

- **RECONHECEDOR DE VOZ VIA REDES NEURAIS**, desenvolvido por Daniel Richetti Lemos, Gabriel Junqueira Rodrigues e Emílio Del Moral Hernandez;
- **Reconhecendo Instrumentos Musicais Através de Redes Neurais Artificiais**, desenvolvido por Carlos Roberto Ferreira de Menezes Júnior, Eustáquio São José de Faria e Keiji Yamaha.

Para o desenvolvimento dos códigos, como python é uma linguagem aberta não foi difícil encontrar inúmeros projetos que tratavam de assuntos semelhantes; analisando vários deles foi possível de se observar que muitas estruturas se tornaram quase como universais na área de redes neurais envolvendo analise de sons, por isso neste projeto, muitas estruturas foram utilizadas para se alcançar o resultado final, com adaptações para se adequarem à proposta e as condições estabelecidas neste trabalho.

Os trabalhos a seguir foram utilizados como base para estruturar os códigos:

- ***Sound Classification Using Deep Learning***, desenvolvido por Mike Smales;
- ***Music Genre Classification with Python***, desenvolvido por Parul Pandey;
- ***Audio Data Analysis Using Deep Learning (Part 2)***, desenvolvido por Nagesh Singh Chauhan.

2.8 Desenvolvimento:

Antes de falar sobre as etapas do desenvolvimento, seguem os fluxogramas das etapas:

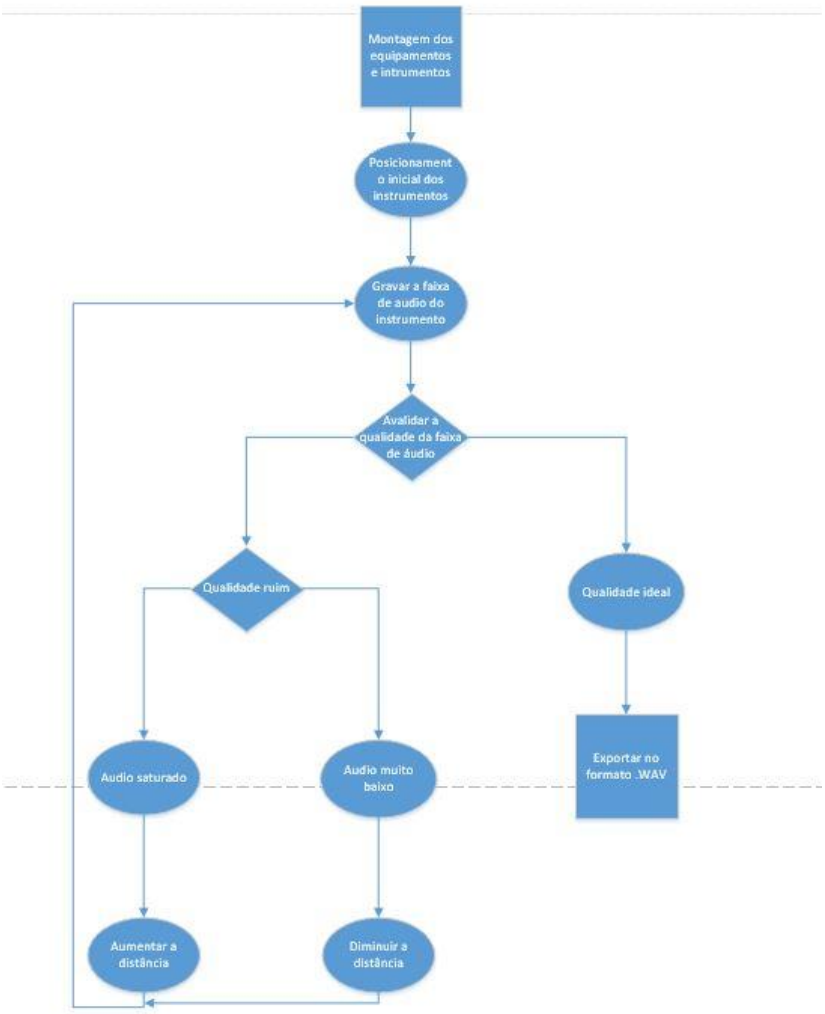

Figura 10 – Fluxograma da captação das faixas de áudio

Fonte: Próprio autor

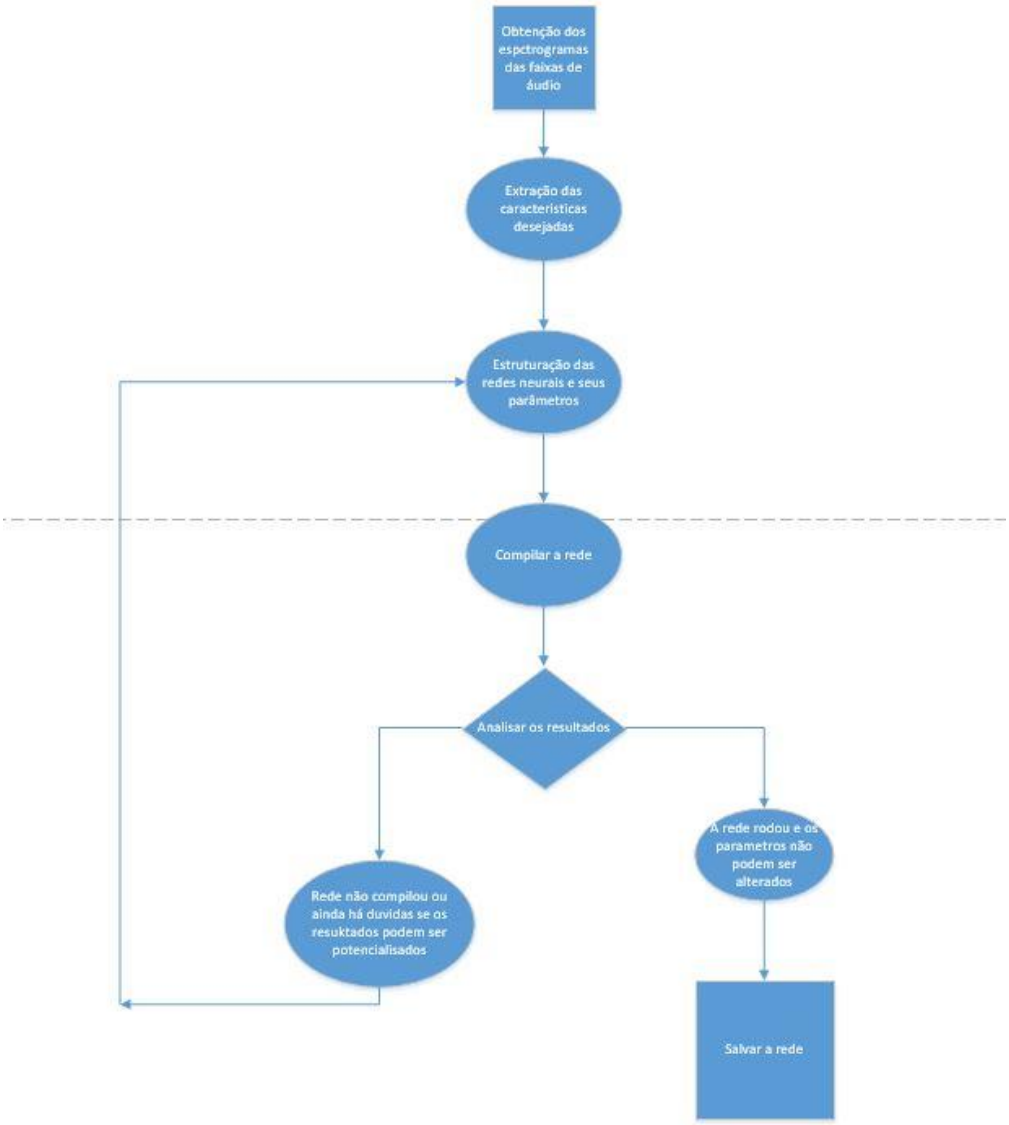

Figura 11 – Fluxograma a partir da obtenção de espectrograma até a conclusão

Fonte: Próprio autor

Coleta De Dados: Para essa etapa foram utilizados os equipamentos que estavam à disposição, equipamentos profissionais de captação de som se mostraram inviáveis em função do preço e tempo para obtenção.

Sendo assim, utilizou-se: notebook Asus com processador i5 e 8GB RAM, microfone de uso casual, cabo y com ponta P2 e mesa de som disponível no local onde os instrumentos se encontravam.

O posicionamento dos instrumentos em relação ao microfone foi ajustado após algumas tentativas a fim de encontrar uma posição em que se pudesse captar as particularidades dos instrumentos, evitando entretanto, a saturação do som.

Foram coletadas 75 amostras para cada taiko contendo faixas com diversos níveis de intensidade de percussão e variações na forma de percussão.

A gravação das faixas de áudio e conversão para o formato .wav, formato necessário para o projeto, foram feitas no Audacity, um software gratuito para gravação e edição de áudio.

As faixas foram separadas em pastas de acordo com o instrumento e salvas na plataforma Google Driver.

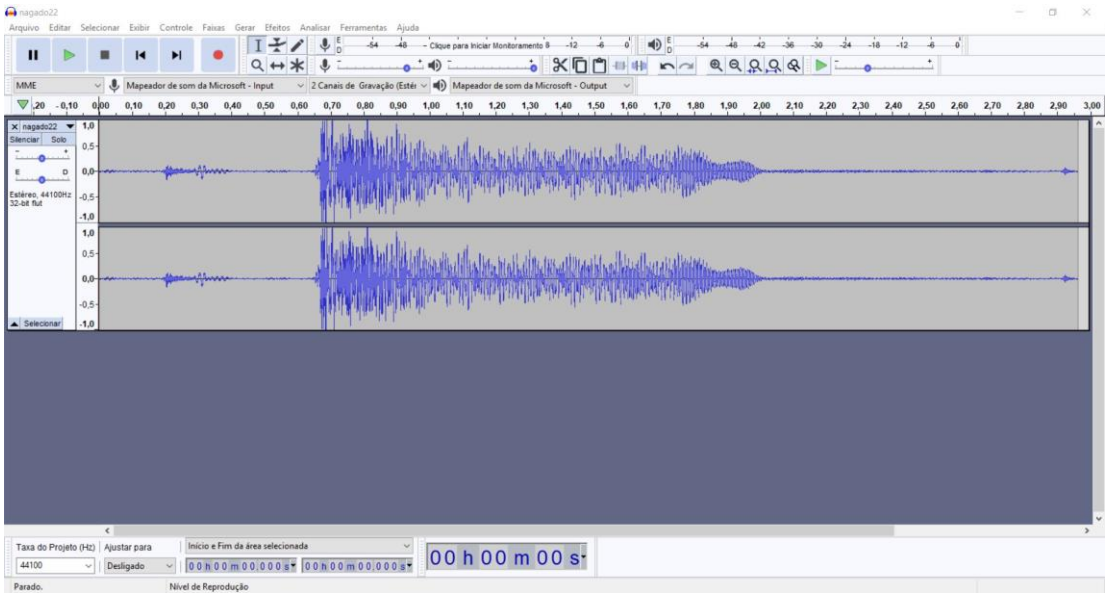

Figura 12 - Interface do Audacity com faixa de som do nagado

Fonte: Próprio autor

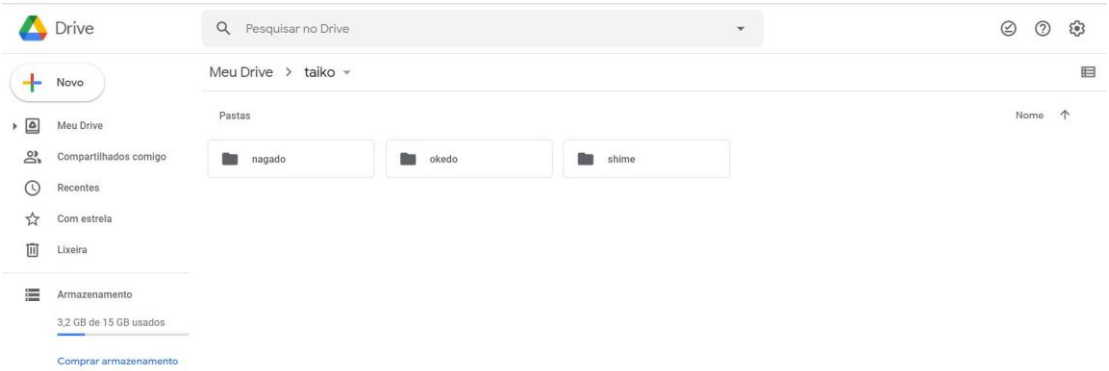

Figura 13 – Armazenamento do Google Drive

Fonte: Próprio autor

Tratamento Dos Dados: A partir desta etapa o desenvolvimento envolverá apenas a programação em linguagem python, realizada na plataforma Google Colab.

Figura 14 – Página inicial do Google Colab

Fonte: https://colab.research.google.com/notebooks/intro.ipynb

A plataforma Google Colab, como explicado por Santos (2020), é um serviço que permite o desenvolvimento e hospedagem de códigos em python na nuvem, além de disponibilizar o acesso ao processamento do Google.

Esta etapa consiste em: 1. obter o espectrograma de todas as faixas de audio previamente armazenadas no Google Driver e 2. extrair as características necessárias para a distinção dos instrumentos, a partir das faixas de áudio.

Das bibliotecas prontas em python a principal para o projeto certamente foi a librosa, uma biblioteca desenvolvida para analise sonora e musical, capaz de fornecer recursos para todas as etapas seguintes.

Para o projeto, a melhor escolha foi a obtenção de espectrogramas em esquema de cores, para facilitar uma melhor visualização da natureza do som do instrumento, ou seja, se ele tende mais ao grave ou agudo.

O código a seguir gera uma pasta taikos, e estabelece o esquema de cores do espectrograma, em seguida, obtém o espectrograma de cada faixa de audio, salvando-os em formato .JPEG e por fim salva os espectrogramas na pasta desejada, que por sua vez possui uma pasta para cada instrumento.

```
[ ] cmap = plt.get_cmap('inferno')
    plt.figure(figsize=(8,8))
    taiko = 'nagado okedo shime'.split()
    for g in taiko:
        pathlib.Path(f'taikos/{g}').mkdir(parents=True, exist_ok=True)
        for filename in os.listdir(f'./drive/My Drive/taiko/{g}'):
            instrument = f'./drive/My Drive/taiko/{g}/{filename}'
            y, sr = librosa.load(instrument, mono=True, duration=5)
            plt.specgram(y, NFFT=2048, Fs=2, Fc=0, noverlap=128, cmap=cmap, sides='default', mode='default', scale='dB');
            plt.axis('off');
            plt.savefig(f'taikos/{g}/{filename[:-3].replace(".", "")}.png')
            plt.clf()
```

Figura 15 – Código para gerar os espectrogramas.

Fonte: Próprio autor

A características extraídas previamente mencionadas foram: spectral centroid, zero-crossing rate e chroma feature.

- **Spectral Centroid:** Indica em qual frequência a energia do espectro está centrada (Chauhan, 2020).

  De acordo com Chauhan (2020) é descrita pela formula:

  $$Spectral\ centroid = \frac{\sum_{k=1}^{N} k.F[k]}{\sum_{k=1}^{N} F[k]} \quad (3)$$

  A razão para a escolha desta característica foram as diferentes intensidades do som, em consequência da energia que cada instrumento emite, além da duração do som que varia para cada instrumento; portanto, foi concluído que cada espectro possuía centros de massa distintos.

- **Zero-Crossing Rate:** Mede o número de vezes que o sinal cruza com o valor 0 (Chauhan, 2020).

De acordo com Chauhan (2020) é descrita pela fórmula:

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} II\{s_t s_{t-1} < 0\} \quad (4)$$

Essa característica foi escolhida em razão da natureza oscilatória bem evidente do som dos instrumentos de percussão.

- **Chroma feature:** Divide o espectro em 12 classes de tons diferentes (C, C#, D, D#, E, F, F#, G, G#, A, A#, B), mostrando a intensidade de cada tom ao longo do espectro (Chauhan, 2020).

De acordo com Tralie (2017?) é descrita pela equação:

$$F\tau[k] = \sum_{n=0}^{N} X[h\tau + n]W[n]e^{\frac{-j2\pi kn}{N}} \quad (5)$$

Sua escolhe se deve ao seu foco em aplicações musicais, identificando com facilidade variações de instrumentos musicais e timbres.

O código a seguir cria um arquivo em formato .CSV que armazena o tipo de taiko além das características extraídas. Para a extração das características foram utilizadas funções da biblioteca librosa que permitiram de forma direta a extração delas a partir das faixas de audio.

```
file = open('dataset.csv', 'w', newline='')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
taiko = 'nagado okedo shime'.split()
for g in taiko:
    for filename in os.listdir(f'./drive/My Drive/taiko/{g}'):
        instrument = f'./drive/My Drive/taiko/{g}/{filename}'
        y, sr = librosa.load(instrument, mono=True, duration=30)
        chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
        spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
        zcr = librosa.feature.zero_crossing_rate(y)
        to_append = f'{filename} {np.mean(chroma_stft)} {np.mean(spec_cent)} {np.mean(zcr)}'
        to_append += f' {g}'
        file = open('dataset.csv', 'a', newline='')
        with file:
            writer = csv.writer(file)
            writer.writerow(to_append.split())
```

Figura 16 – Código para extrair spectral centroid, zero-crossing rate e chroma feature

Fonte: Próprio autor

Elaboração Da Rede Neural: A princípio a intenção era apenas utilizar a convolucional, uma vez que os primeiros trabalhos encontrados trabalhavam com espectrogramas ou a própria rede convolucional; posteriormente, entretanto, nos estudos de trabalhos que envolviam a aplicação de rede neural na área musical, foi possível observar que, convencionalmente são empregados dois modelos de redes neurais: a feedforward e a convolucional. Ainda que não houvesse a certeza de que ela seria capaz de ser aplicada na classificação de instrumentos com a mesma função, optou-se por usar também a rede feedforward.

As estruturas de códigos a baixo são usadas de forma recorrente em projetos e práticas envolvendo aplicação de rede neural em classificação e reconhecimento de sons ou gêneros musicais e representam a arquitetura de rede feedforward e convolucional respectivamente.

```
model = Sequential()
model.add(layers.Dense(512, activation='relu', input_shape=(X_train.shape[1],)))
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(64, activation='softmax'))
model.add(layers.Dense(10, activation='softmax'))
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Figura 17 – Código de rede feedforward

Fonte: Próprio autor

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(128, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(10, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

Figura 18 – Código de rede convolucional

Fonte: Próprio autor

Treinamento Da Rede: Esta etapa consiste em aplicar o banco de dados às redes previamente elaboradas, bem como configurar as características pelas quais as aplicações devem compilar. Para a aplicação foi configurado que a rede deveria passar por 1000 épocas a 2 passos por época, para cada uma das redes.

O valor de épocas utilizado foi o máximo alcançado, mesmo com a plataforma Google Colab, não foi possível de lidar com números maiores de épocas para o treinamento da rede convolucional. Para o numero de épocas novamente o fator limitante foi a rede convolucional que não permitia valores maiores.

```
history = model.fit(X_train, y_train, epochs=1000, batch_size = 2, validation_data=(X_test, y_test))
```

Figura 19 – Código de treinamento da rede feedforward

Fonte: Próprio autor

```
model.summary()

model.compile(loss = 'categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

history = model.fit(train_generator, epochs=1000, steps_per_epoch=2, validation_data = validation_generator, verbose = 1, validation_steps=3)
```

Figura 20 – Código de treinamento da rede convolucional

Fonte: Próprio autor

3.7 RESULTADOS E DISCUSSÃO

Espectrogramas: Após a efetiva transformação da amostragem, foram gerados espectrogramas e salvos no formato .PNG.

Os espectrogramas nem sempre serão iguais uma vez que há vários fatores que influenciam, tais como a intensidade ou a velocidade da percussão, sons de fundo, a qualidade do instrumento de captação, entre outros, porém elas seguem alguns padrões dos espectrogramas apresentados:

- **Nagado:** instrumento referente a figura 1, possui coloração mais clara voltada para o amarelado ou alaranjado, indicando sua natureza mais grave;

  Seus espectrogramas possuem uma explosão inicial seguida de um decaimento acelerado de sua energia, porém seus espectros possuem um caráter mais duradouro que os demais.

- **Okedo:** representado pela figura 2, possui principalmente em seu pico uma coloração mais escura em relação ao nagado, indicando sua tendência menos grave;

  Seus espectros possuem uma perda de energia menos acentuada e duração mais curta em relação aos demais.

- **Shime:** representado pela figura 3, principalmente em seu pico possui a coloração mais escura que os demais, tendendo ao roxo, o que indica sua natureza mais aguda.

  Seus espectros possuem perda de energia e duração intermediaria em relação aos demais.

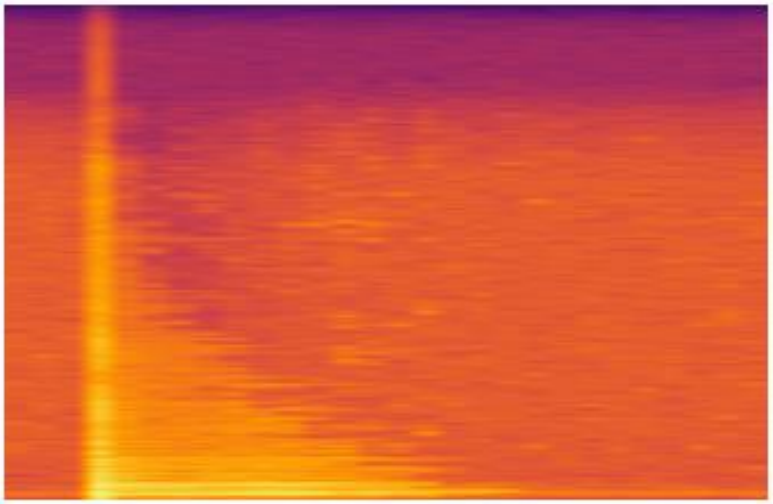

Figura 21 – Espectrograma de nagado com batida alta

Fonte: Próprio autor

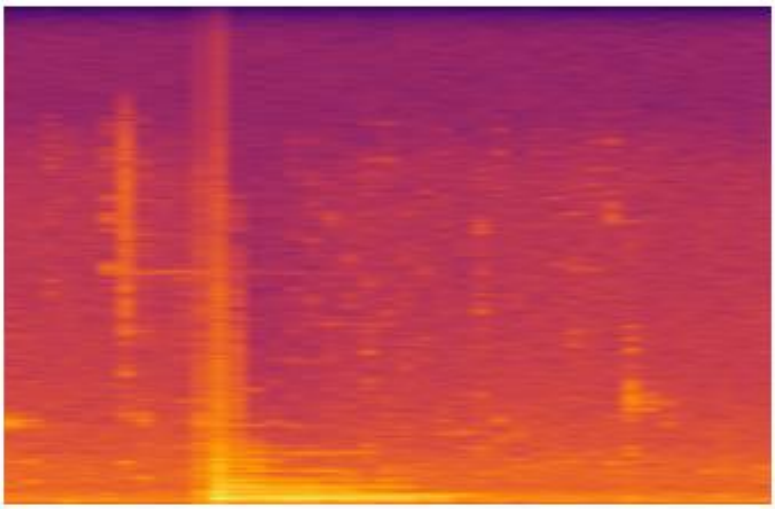

Figura 22 – Espectrograma de nagado com batida baixa

Fonte: Próprio autor

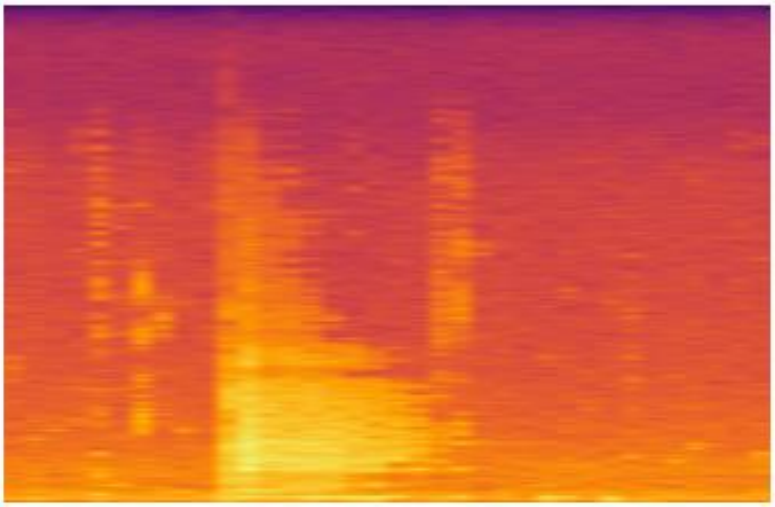

Figura 23 – Espectrograma de nagado com batida na lateral

Fonte: Próprio autor

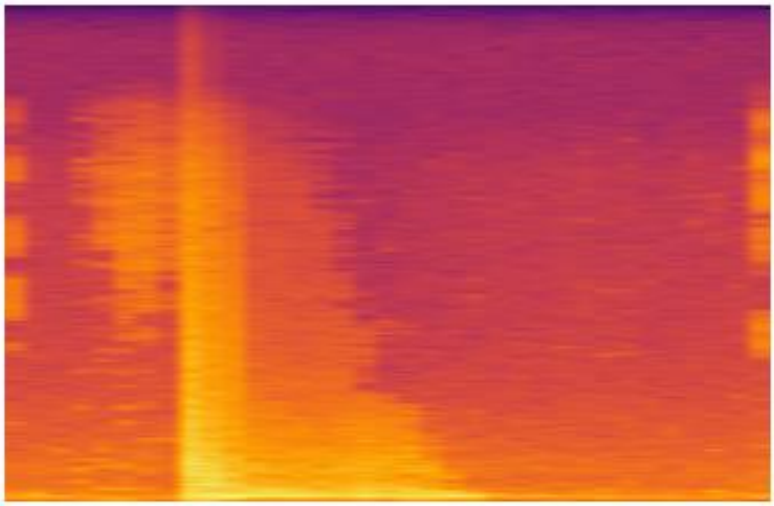

Figura 24 – Espectrograma do okedo com batida alta

Fonte: Próprio autor

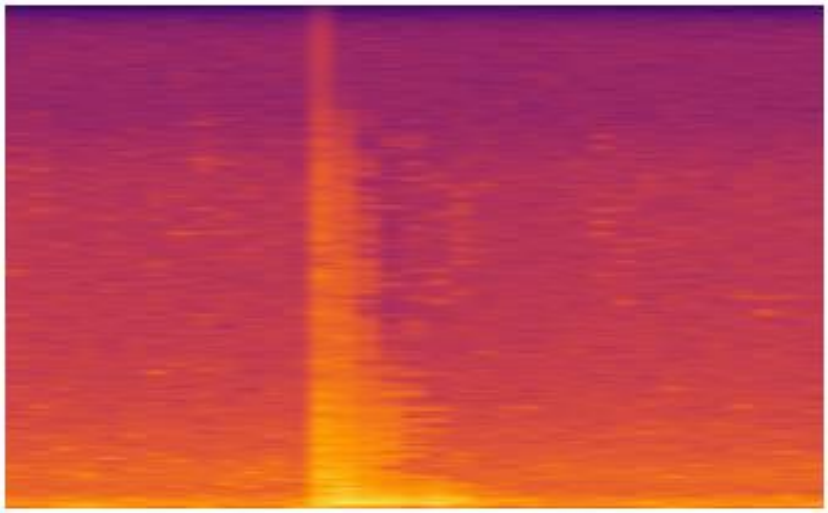

Figura 25 – Espectrograma do okedo com batida baixa

Fonte: Próprio autor

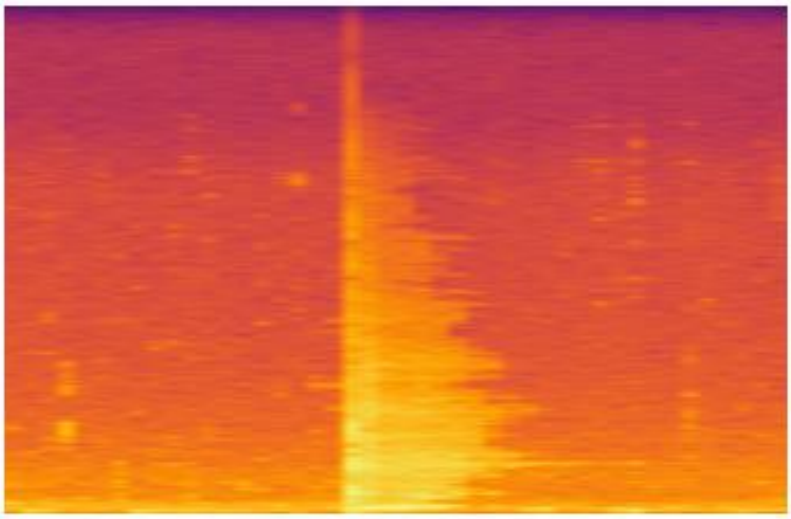

Figura 26 – Espectrograma de okedo com batida na lateral

Fonte: Próprio autor

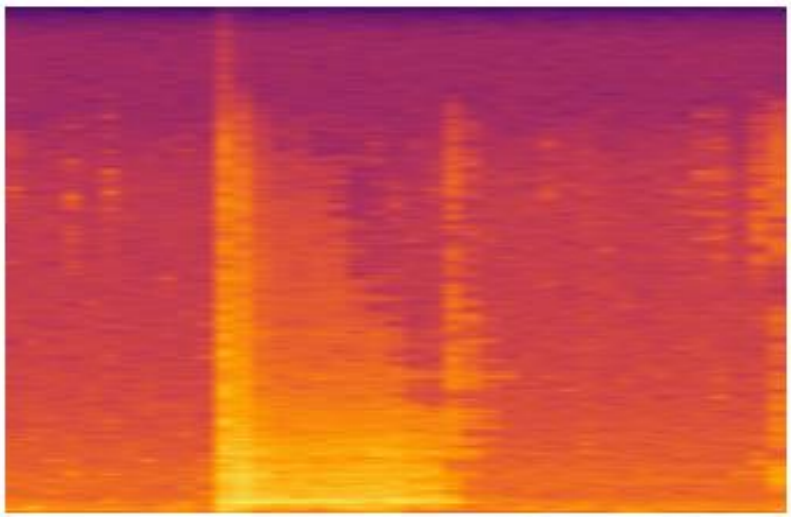

Figura 27 – Espectrograma do shime com batida alta

Fonte: Próprio autor

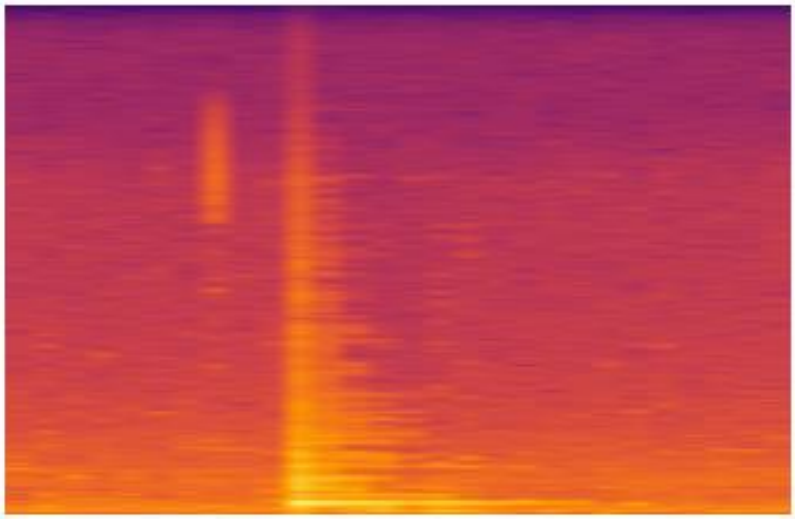

Figura 28 – Espectrograma do shime com batida baixa

Fonte: Próprio autor

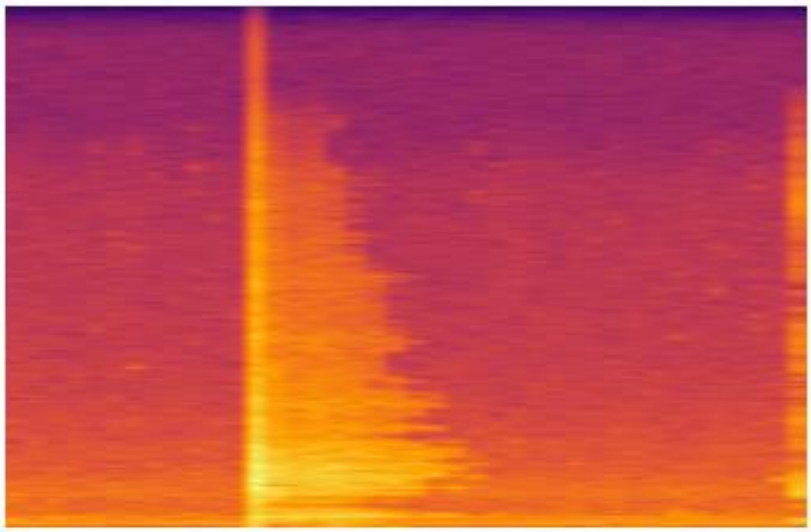

Figura 29 – Espectrograma do shime com batida na lateral

Fonte: Próprio autor

Características extraídas: As três características (zero-crossing rate, spectral centroid e chroma feature) se mostraram o suficiente para a aplicação e foram utilizados para o treinamento da rede neural multilayer feedforward.

Os dados extraídos foram armazenados em um arquivo .CVS, que após tratado no Excel assim se apresentam:

	A	B	C	D	E
1	filename	chroma_stft	spectral_centroid	zero_crossing_rate	label
2	nagado1.wav	0.3550942833298219	20.128.732.070.914.000	0.117899386682243	nagado
3	nagado2.wav	0.3465236674816675	20.200.114.513.874.600	0.121043079304245	nagado
4	nagado3.wav	0.4137423184186362	2.372.900.832.170.940	0.162503756009615	nagado
5	nagado4.wav	0.35163750092517926	21.451.445.985.572.200	0.136555989583333	nagado
6	nagado5.wav	0.35164397508242407	2.145.129.382.645.030	0.136742001488095	nagado
7	nagado6.wav	0.34224407183329475	1.981.076.560.441.700	0.115035445601851	nagado
8	nagado7.wav	0.3833404213318679	21.251.812.145.469.800	0.137529861828512	nagado
9	nagado8.wav	0.35195734124197825	20.604.212.433.774.400	0.125094229714912	nagado
10	nagado9.wav	0.4216519419040759	2.191.500.206.541.060	0.1383056640625	nagado
11	nagado10.wav	0.4662408052628865	2.308.739.845.708.820	0.167447916666666	nagado
12	nagado11.wav	0.3994257172528845	22.869.304.195.911.300	0.150204613095238	nagado
13	nagado12.wav	0.48747689311918324	26.007.741.009.175.700	0.188308071082746	nagado
14	nagado13.wav	0.3736541317873051	12.001.901.305.238.300	0.086412702287946	nagado
15	nagado14.wav	0.3914718733737488	6.195.420.495.200.150	0.024444855011261	nagado
16	nagado15.wav	0.3764621064575901	710.679.677.912.366	0.030317034040178	nagado
17	nagado16.wav	0.36325120515180304	8.429.938.405.380.830	0.034122721354166	nagado
18	nagado17.wav	0.4196945022514232	8.236.717.301.955.620	0.035542659622302	nagado
19	nagado18.wav	0.37151430378363576	8.285.268.365.830.420	0.034701936141304	nagado
20	nagado19.wav	0.39502724457529353	7.460.277.611.075.060	0.027350217301324	nagado
21	nagado20.wav	0.3950247479656962	7.461.508.287.189.220	0.027266142384105	nagado
22	nagado21.wav	0.3767333415607126	7.302.987.441.154.750	0.027560355968045	nagado
23	nagado22.wav	0.38664929715689206	801.759.868.571.153	0.032848358154296	nagado
24	nagado23.wav	0.34532833817975195	7.135.475.752.682.780	0.031750288165983	nagado
25	nagado24.wav	0.3963311705958828	6.923.155.499.765.140	0.025915075231481	nagado
26	nagado25.wav	0.3907270817146592	6.730.588.919.794.200	0.026950588474025	nagado
27	nagado26.wav	0.3309093394284365	6.606.947.036.407.890	0.032536204268292	nagado
28	nagado27.wav	0.44980689155168885	741.495.160.905.786	0.027380022321428	nagado
29	nagado28.wav	0.3789568914586276	6.170.128.829.842.420	0.028274536132812	nagado

Figura 30 – Exemplo de espectrograma do shime

Fonte: Próprio autor

3.8 Redes Neurais:

A multilayer foi elaborada de forma a encontrar padrões nas características obtidas das faixas de áudio enquanto a convolucional encontra padrões nas imagens dos espectrogramas.

Após a realização dos treinos e testes a precisão alcançada utilizando 1000 épocas e 1 passo por época, foi de 99,46% para a multilayer com um período de treino de 3 minutos enquanto a convolucional obteve 76,79% com 1 hora e meia de treino.

A rede convolucional opera com extensos bancos de dados, portanto, com mais faixas de áudio a rede conseguiria obter melhores resultados em termos de precisão, porém, considerando o tempo necessário para o seu treinamento e a capacidade de processamento necessária, é evidente que em relação a rede feedforward ela é a menos recomendada.

Foi uma surpresa a confirmação da eficácia da rede feedforward para o projeto, que se mostrou muitos mais versátil para esse tipo de aplicação, uma vez que foi capaz de atingir

precisão próxima da máxima com um número limitado de amostragens, exigindo menos da máquina e em um tempo consideravelmente menor para o treino que a rede convolucional.

4 CONCLUSÃO

Mesmo com instrumentos de percussão com uma distinção sonora menor que instrumentos de funções diversas foi possível o desenvolvimento de uma rede neural capaz de distingui-los e classificá-los com uma taxa alta de acerto, mesmo com uma quantidade pequena de amostragens de áudio, principalmente para a rede convolucional que exige um banco de dados mais robusto para conseguir aprender e ser capaz de classificar elementos não antes vistos.

O pretendido no início do desenvolvimento do trabalho era, além da classificação dos taikos, a detecção de um ou múltiplos instrumentos em tempo real e a interação com a rede neural para devidamente classificá-los, porém além de necessitar de um banco de dados muito mais vasto, o que não foi permitido devido a complicações decorrentes da pandemia, esta funcionalidade exigiria um estudo mais complexo, uma vez que a rede deveria aprender a ignorar demais sons e situações de silêncio, interagindo apenas com os taikos, portanto exigiria uma grande gama de amostragens de sons corriqueiros dos mais diversos para assim aprender a identificá-los. Por fim, essa etapa ainda exigiria equipamentos de nível profissional para a captação de áudio dos instrumentos, a razão desta necessidade é a qualidade das faixas de áudio captadas que teriam pouca interferência e vazamento de áudio não desejado, assim fornecendo amostras mais precisas para o aprendizado da rede.

O principal, ou seja, o estudo da possibilidade da elaboração de uma rede neural capaz de classificar instrumentos de percussão mostrou-se possível ainda com duas possibilidades exploradas pelo projeto, ficando assim para trabalhos futuros o acréscimo da função de reconhecimento de instrumentos e classificação em tempo real.

REFERÊNCIAS

AGGARWAL, C. C. **Neural Networks and Deep Learning**. 1. ed. Nova York: Springer, 2018.

ALPAYDıN, E. **Introduction to Machine Learning**. 2. ed. London: The MIT Press, 2010.

CAPÍTULO 10 – As 10 Principais Arquiteturas de Redes Neurais. **Deep Learning Book**, 2015? Disponivel em: <http://deeplearningbook.com.br/as-10-principais-arquiteturas-de-redes-neurais/>. Acesso em: 28 Julho 2020.

CHANDRA, A. L. Perceptron Learning Algorithm: A Graphical Explanation Of Why It Works. **towards data science**, 2018. Disponivel em: <https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975>. Acesso em: 17 Maio 2020.

CHAUHAN, N. S. Audio Data Analysis Using Deep Learning with Python (Part 1). **kdnuggets**, 2020. Disponivel em: <https://www.kdnuggets.com/2020/02/audio-data-analysis-deep-learning-python-part-1.html>. Acesso em: 15 Agosto 2020.

CHAUHAN, N. S. Audio Data Analysis Using Deep Learning with Python (Part 2). **KDnuggets**, 2020. Disponivel em: <https://www.kdnuggets.com/2020/02/audio-data-analysis-deep-learning-python-part-2.html>. Acesso em: 15 Agosto 2020.

CIRIACO, D. O que é Inteligencia Artificial? **CollBusinessNews**, 2008. Disponivel em: <https://collbusinessnews.com.br/o-que-e-inteligencia-artificial/>. Acesso em: 17 Abril 2020.

HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. [S.l.]: Pearson, 2009.

HONDA, H.; FACURE, M.; YAOHAO, P. Os Três Tipos de Aprendizado de Máquina. **LAMFO**, 2017. Disponivel em: <https://lamfo-unb.github.io/2017/07/27/tres-tipos-am/>. Acesso em: 18 Novembro 2020.

IZUMI, M. Um Resumo Histórico do Taiko - parte 1. **Descubra Nikkei**, 2010. Disponivel em: <http://www.discovernikkei.org/pt/journal/2010/3/15/taiko/>. Acesso em: 16 Novembro 2020.

LUGER, G. F. **Artificial Inteligence**. 6. ed. [S.l.]: Pearson, 2013.

MAINI, V. Machine Learning for Humans. **Medium**, 2017. Disponivel em: <https://medium.com/machine-learning-for-humans/reinforcement-learning-6eacf258b265>. Acesso em: 14 jul. 2020.

MENEZES, N. N. C. **Introdução à Programação com Python**. 2. ed. São Paulo, SP: Novatec, 2014.

NIELSEN, M. **Neural Networks and Deep Learning**. Nova York: Determination Press, 2015. Acesso em: 20 Maio 2020.

O QUE É TAIKO? **ISHINDAIKO**, 2012? Disponivel em: <http://www.ishindaiko.com.br/index.php/o-que-e-taiko?limitstart=0>. Acesso em: 16 Novembro 2020.

PANDEY, P. Music Genre Classification with Python. **toward data science**, 2018. Disponivel em: <https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8>. Acesso em: 12 Agosto 2020.

PEREIRA, T. Qual a diferença entre Machine Learning e Deep Learning? **computerworld**, 2017. Disponivel em: <https://computerworld.com.br/brandpost/qual-diferenca-entre-machine-learning-e-deep-learning/>. Acesso em: 22 maio 2020.

PEREZ, R. Algoritmo BackPropagation. **PROGRAMAR**, n. 57, p. 16-19, Julho 2017. ISSN ISSN 1647-0710.

SANTANA, M. Deep Learning: do Conceito às Aplicações. **medium**, 2018. Disponivel em: <https://medium.com/data-hackers/deep-learning-do-conceito-%C3%A0s-aplica%C3%A7%C3%B5es-e8e91a7c7eaf>. Acesso em: 24 maio 2020.

SANTOS, T. G. Google Colab: o que é e como usar? **alura**, 2020. Disponivel em: <https://www.alura.com.br/artigos/google-colab-o-que-e-e-como-usar#:~:text=O%20Google%20Colaboratory%2C%20carinhosamente%20chamado,vamos%20ver%20como%20us%C3%A1%2Dlo.>. Acesso em: 19 Novembro 2020.

SHALEV-SHWARTZ, ; BEN-DAVID, S. **Understanding Machine Learning:** From Theory to Algorithms. 1. ed. New York: CAMBRIDGE UNIVERSITY PRESS, 2014.

SILVA, I. N. D.; SPATTI, D. H.; FLAUZINO, R.. **Redes Neurais Artificiais para engenharia e ciências aplicadas**. 2. ed. São Paulo: Artliber, 2016.

SMALES, M. Sound Classification using Deep Learning. **medium**, 2019. Disponivel em: <https://mikesmales.medium.com/sound-classification-using-deep-learning-8bc2aa1990b7>. Acesso em: 05 Julho 2020.

TIPOS de Taiko (instrumentos). **kaminaritaikoRJ**, 2012. Disponivel em: <http://kaminaritaikorj.blogspot.com/2012/09/tipos-de-taiko-instrumentos.html>. Acesso em: 16 Novembro 2020.

TRALIE, C. ECE 381 Data Expeditions Lab 1. **ctralie**, 2017? Disponivel em: <http://www.ctralie.com/Teaching/ECE381_DataExpeditions_Lab1/>. Acesso em: 24 Outubro 2020.

VAZ, W. Saiba mais como o python surgiu e qual o seu cenário atual. **EU SOU DEV**, 2018. Disponivel em: <https://www.eusoudev.com.br/python-como-surgiu/>. Acesso em: 14 jul. 2020.