Relatório do Projeto de IA



Feito por: Miguel Salgado nº22002195 e Tiago Catano nº22002128

Docente: Francisco Melo Pereira e Tiago Candeias

Cadeira: Inteligência Artificial

Curso: Engenharia Informática

Ano: 2022/2023

Índice

[Introdução 3](#_Toc137975051)

[Bibliotecas 4](#_Toc137975052)

[DataSet 5](#_Toc137975053)

[Código 6](#_Toc137975054)

[Conclusão 18](#_Toc137975055)

# Introdução

Na cultura pop existem centenas senão milhares de histórias que rondam o assunto inteligência artificial, alguns exemplos são: Matrix, I Robot, Terminator e Avengers: Age of Ultron. No qual o ponto comum entre quase todas é que as máquinas estão no estagio tão avançado de inteligência que decidiram-se virar contra a sociedade humana, porém na vida real não chegamos a esse ponto, todavia podemos dizer que a inteligência artificial como a conhecemos tem evoluído muito rapidamente, desde a sua primeira menção nos anos 40, onde Warren McCulloch e Walter Pitts descreve redes neuronais e estruturas de raciocínios artificial, até os dias de hoje, no qual já existe um modelo muito conhecido a nivel global, o famoso “ChatGPT”, que vai na sua 4ª versão. Existe já quem diga que a nova revolução industrial virá devido ao boom da inteligência artificial.

Neste projeto não iremos criar nenhuma máquina que pretenda terminar com a raça humana, aquilo que propomos é a criação de um modelo que consiga distinguir músicas pelo seu genero musical. Para a realização deste projeto será então realizado o desenvolvimento e implementação de um sistema de classificação automatizado (sistema completamente autónomo) de géneros musicais, fazendo uso de CNN (Convolution Neural Network). É proposto a utilização de técnicas de Deep Learning para a analise e extração de dados importantes de músicas tendo por fim categorizá-las por genro musical.

# Bibliotecas

* Pandas
  + Biblioteca, que serve para manipulação e análise de dados, este aplica uma estrutura de dados chamada DataFrame (estrutura de dados tabular composta por colunas e linhas).
* Librosa
  + Biblioteca utilizada para análise de áudio e processamento de sinais de áudio. Esta poder ler vários tipos de ficheiro áudio, WAV, MP3 e FLAC. Após uma leitura destes, Librosa fornece vários métodos de extração de dados, tais como espectrogramas, cronogramas e MFCCS (Mel frequency cepstral coefficients). O Librosa é frequentemente utilizado em conjunto com outras bibliotecas processamento de dados em Python, tais como NumPy, SciPy e Matplotlib.
* Keras
  + Utilizada para criação de redes neurais de maneira percetível. Este permite classificar imagens, o processar de linguagem, áudio e também permite fazer a previsão de tempo.
* Sklearn
  + Biblioteca usada para a mineração de dados e analise destes. Esta possui vários algoritmos, como por exemplo regressão logística, árvore de decisão, random forest e redes neurais.
* Seaborn
  + Biblioteca usada para visualização de dados em python, esta cria uma interface onde é mostrador vários tipos de gráficos estatísticos. Tipos de Gráficos:
    - Gráficos de distribuição;
    - Gráficos de dispersão;
    - Gráficos de barras.
* Matplotlib
  + Biblioteca que serve para visualização de dados em python, podendo também criar gráficos, mapas de calor, histogramas.

# DataSet

GTZAN Dataset - Music Genre Classification, apresenta informação sobre uma variedade de gêneros musicais, oferecendo uma visão geral de suas principais características e elementos musicais característicos. No entanto o elemento que utilizaremos existente neste dataset são as próprias músicas.

Este apresenta um total de 1000 músicas, sendo elas dividas em categorias, tais como Blues, Classical, country, disco, Rock e ETC… .

Uma das razões pela escolha deste dataset foi a sua múltipla utilização em outros projetos similares ao nosso, permitindo ter um dataset fidedigno, para adquisição de dados, necessários.

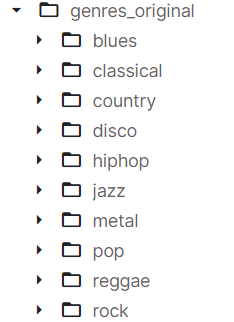
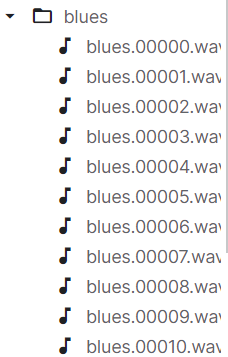


Figura 3 - Músicas

Figura 1 - Géneros De Música

# Código

Neste capítulo será explicado como foi a implementação da leitura dos dados, o seu pré-processamente, a sua separação dos dados de treino e validação e por fim o treino e teste do modelo CNN.

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Figura 4 - Música e Pré Processar do Melspectogram

Nesta parte do código é feito o processamento das músicas adquiridas no dataset e a extração de recursos na espectrograma para cada música. Esta usa várias bibliotecas, tais como OS, Librosa e Numpy. No código é inicializado duas listas vazias que depois vão guardar os dados pretendidos. O código inicializa duas listas vazias, **genderlabel** para armazenar os gêneros das músicas e a **musicData** para armazenar os dados da espectrograma mel de cada música.

Após a inicialização das listas é executado a função **os.listdir()** para obter as musicas por cada género. Dentro do loop, o código repete sobre os arquivos de música em cada género, para cada música esta é dada a sua categoria/género.

Em seguida, o código executa a função **mimosa.load()** para obter a amplitude e a taxa de amostragem, para todas as musicas, sendo também indicado que a leitura da musica começará nos 15 segundos finais, é depois calculado o espetrograma para cada música usando o **mimosa.feature.melspectrogram()**. A **espectrograma** **mel** é uma representação visual do espectro de frequência da música ao longo do tempo.

Após a aquisição do espectrograma mel, este é convertido para decibéis usando **mimosa.power\_to\_db()** , normalizando os valores do espectrograma. É usado o **np.resize()**, para o redimensionamento do espetrograma dando a ele uma forma fixa, pois cada música tem um formato próprio, desta forma ficam todas com um formato standard.

Por fim, a espectrograma mel redimensionado é adicionado à lista **musicData**, criando uma lista de espectrogramas mel correspondentes às músicas processadas.

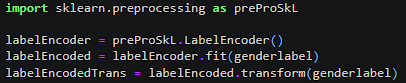


Figura5- Enconding das Categorias

No código a cima é feito a codificação dos géneros, usando como fonte de tal codificação o **sklearn.preprocessing**.

Na primeira linha é demonstrado o responsável pelo transformar dos textos em valores numéricos. Na segunda linha é apresentado **labelEncoder.fit(),** que vai ter como argumento a nossa lista de géneros, **genderlabel,** este método ira preparar os géneros para a codificação, na terceira linha onde ira acontecer a transformação dos géneros em valores numéricos que irão corresponder ao diferentes tipos de género utilizado.

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

Figura 6 - Normalização de Valores

Na imagem de código demonstrada em cima é criado duas variáveis que são **indexMin** e **indexMax** que vão servir para gerir o número de faixas de músicas a serem normalizadas, juntamente com esta é criado uma lista chamada **normalized\_data,** que como o nome indica vai servir para guardar os dados normalizados. O **numCategories** irá calcular, a quantidade de categorias lidas pelo programa, através da quantidade de labels codificados dividindo eles por 100, depois irá ser realizado um loop, no qual será adquirido a valor mínimo e o valor máximo de cada categoria, com base no intervalo de dados obtidos pelas variáveis **indexMin** e **indexMax**, com base nesta informação dentro do loop seguinte, realizado para cada musica da categoria atual, é realizada a normalização das mesmas utilizando a formula (X – Xmin) / (Xmax – Xmin).

Por fim estes dados iriam ser armazenado na lista mencionada anteriormente, **normalized\_data**. As variáveis usadas **indexMin** e **indexMax**, são aumentadas em 100 pois cada categoria neste projeto contem 100 músicas.

Anteriormente para normalização dos valores tinha sido usado outro método que implicava o uso da média do dataset para a normalização dos valores pretendidos, este normalizava todos os dados independentemente do seu género, já a normalização min max, verifica para cada genro o seu ponto máximo e mínimo e normaliza somente os dados desse género. Por fim provou--se que a normalização min max é mais adequado aos dados que procuramos.

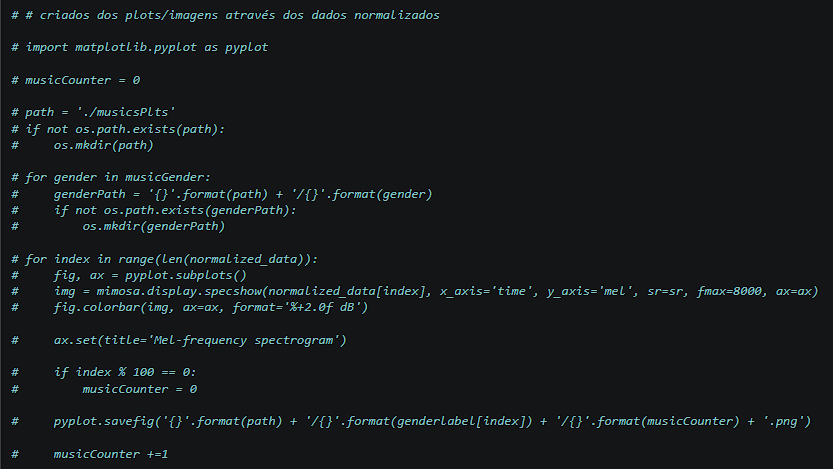


Figura 6.1 - Gráficos de Imagem Espectral

O código referido na figura em cima é utilizado para gerar os gráficos de imagem espectral utilizando os dados normalizados para cada música, estes depois de gerados são guardados no diretório musicsPlots, organizados em pastas por género musical.

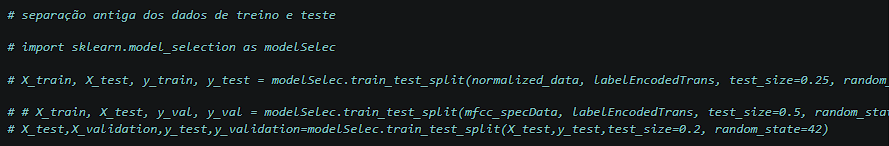


Figura 6.2 - Antigo treino

Inicialmente esta era a forma utilizada para a divisão de dados do treino e do teste, através do método **train\_test\_split** e a **validation**.

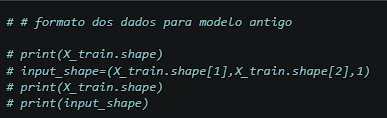


Figura 6.3 - Formatação de Dados

Neste módulo, era realizada a formatação dos dados para o do primeiro modelo usado.



Figura 7 - K-FOLDS Versão Anterior

Na figura 7 é mostrada a versão anterior dos K-FOLDS, decidimos não utilizar esta devido as funcionalidades limitadas, sendo uma delas a limitação de categorias selecionadas nesta função. Sendo assim optamos por utilizar outro método que permite adicionar categorias extras, filtras os dados, como se pode ver na figura 9.

Anterior a este modelo era usado um similar, no qual a separação realizada, é explicada na figura 6.2, não sendo necessário a realização do ciclo for no K-Fold, pois ambas tinham como função a separação dos dados. Esse modelo deixou-se de ser usado, devido a problemas de overfitting.

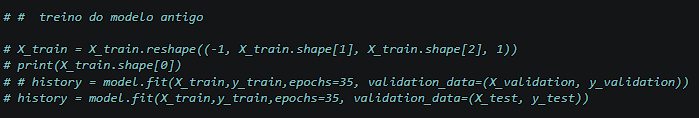


Figura 7.1 - Treino do Modelo

Neste modulo os dados eram novamente formatados e de seguida era realizado o treino do modelo e a sua validação.

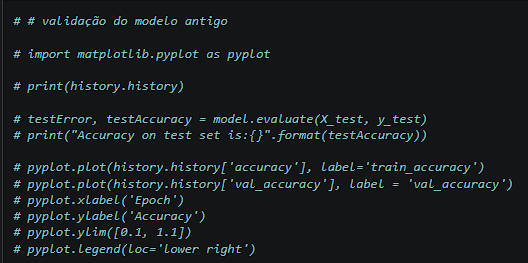


Figura 7.2 - Teste do modelo

Na figura em cima é mostrado a avaliação do modelo, gerando um gráfico da accuracy ao longo das épocas, com base na accuracy do treino e validação.

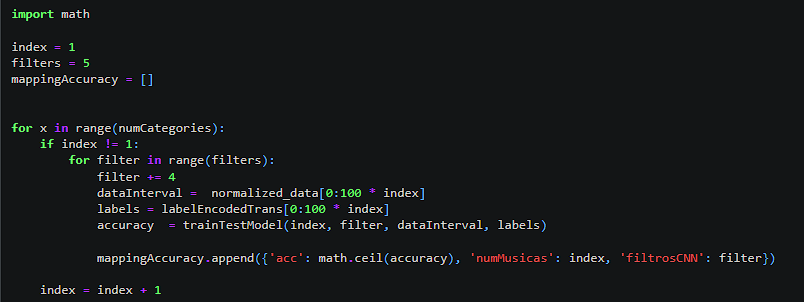


Figura 8 – Accuracy de Acordo com o número de músicas e filtros

Na figura em cima é corrido um ciclo que irá correr o número de vezes equivalente ao número de categorias lidas pelo código referido na figura 4.

Dentro do ciclo é efetuada uma verificação onde o **index**, que equivale ao numero de categorias, atual é diferente de 1, para que o modelo não treine só com uma categoria, dentro dessa validação é realizado um novo ciclo que irá correr o número de vezes equivalente aos filtros a serem testados no modelo, sendo testados 4 variantes de filtros. São definidos 2 arrays que vão receber o intervalo dos dados normalizados e labels sendo que o index máximo é equivalente ao número de músicas por categoria a multiplicar pelo número de categorias a serem testadas.

De seguida é recebida a accuracy de cada modelo testado, sendo que são enviados como argumentos, index (número de categorias), filter (número de filtros), dataInterval (intervalo dos dados normalizados) e labels (intervalo dos labels normalizados), por fim são adicionados a lista **mappingAccuracy** a accuracy recebida, os filtros usados e do número de géneros.

Uma imagem com texto, captura de ecrã, software

Descrição gerada automaticamente

Figura 9 - KFOLD E Training

Na figura 9 são feitos os splits pelo número de géneros com **random\_state** de 42, é feito um ciclo de split dos dados de treino e teste.

Dentro deste ciclo é inicializado o modelo CNN, para que cada ciclo tenha um modelo único, a partir do método **theModel**, utilizando 3 argumentos neste, que são filters (filtros), splitNumb (categoria), data (dados). Faz-se o treino do modelo com os dados de treino e a sua validação com os dados de teste, dentro do método **fit**, para este treino e validação são usadas 25 épocas (iterações do modelo). É registada a accuracy do modelo através do seu teste, a partir do método **evaluate**. A accuracy é guardada em **acc\_per\_fold** em percentagem e a média desta é enviada no return deste método.



Figura 10 - Criação do Modelo Neural Convolucional(CNN)

Na figura 10 é apresentado o modelo CNN, utilizado neste projeto este foi desenvolvido utilizando a biblioteca **Keras** vindo do **TensorFlow**. Este modelo necessita de 3 parâmetros que são **filters**, **neurons** e **data.**

Este possui várias camadas tais como:

* Input Layer
  + Ele recebe o formato de entrada **(data.shape[1], data.shape[2],1)**. A entrada passa por uma camada convolucional 2D com filters tendo como tamanho (3, 3). A função de ativação **relu** é aplicada à saída da camada convolucional. A saída passa por uma camada de max pooling com um tamanho (3, 3) e um passo de (2, 2). A normalização é aplicada para normalizar as ativações da camada anterior.
* Flatten Layer
  + Transforma os dados em um vetor unidimensional;
* Dense Layer
  + Conecta o layer com o **filters,** áfunção **relu;**
* Output Layer
  + Conecta o layer onde se encontra os **neurons**, que por defeito são o número de categorias atuais e utiliza uma nova função de ativação chamada de softmax.

Por fim o modelo é compilado com as seguintes configurações, otimização com uma taxa de aprendizagem de 0.01%, é indicada a função de **loss** para gestão das categorias e indicada a métrica de performance do modelo, neste caso **accuracy**, por fim é retornado o modelo criado neste metodo.

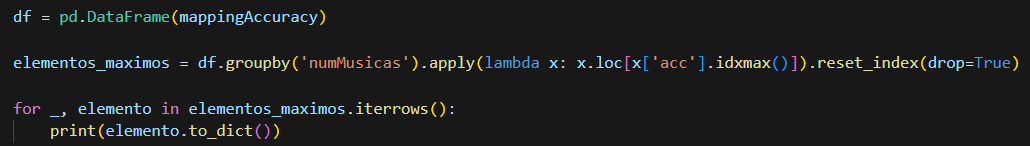


Figura 11 - Accuracy máxima de cada modelo com base na quantidade

Na figura em cima, é convertida para dataframe a lista **mappingAccuracy**, de seguida são agrupados em **elementos**\_**maximos** somente as linhas do dataframe no qual a **accuracy** por quantidade de música é máxima.

Figura 12 - Gráficos 2D e 3D

Na figura 12 são gerados dois tipos de gráfico um em 2D e outro em 3D.

No gráfico 2D os dados são dispostos na seguinte forma o eixo do Y é mostrado o **Número de Géneros** e no eixo do X a **Accuracy**.

No 3D é formatado na seguinte forma, eixo Z, contem **Accuracy**, eixo do X, **Filtros** e o eixo do Y, contem os **Géneros**, para além deste são criados pontos no gráfico através do scatter3D, pois o plot3D só faz uma linha.

# Conclusão

Falar sobre os problemas overffiting normalização achar máximos

Bibliografia

<https://observador.pt/opiniao/ia-a-catalisadora-de-uma-nova-revolucao-industrial/>

<https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification>