Documentação de Código: Sophie

Grupo:

João Pedro Viana,
Mikael da Silva,
Rebeca Rufino

Índice

Sobre o Projeto

Descrição do Projeto

A Sophie foi projetada para fazer uma musica baseado em outras músicas e ela também pode classifica ás música por estilo como sendo animada ou calma.

Participantes

João Pedro: Ficou encarregado pela parte do código com ajuda do Mikael
Mikael: Ajudou na programação e fez a documentação
Rebeca: Fez a escolha das músicas e ajudou na documentação

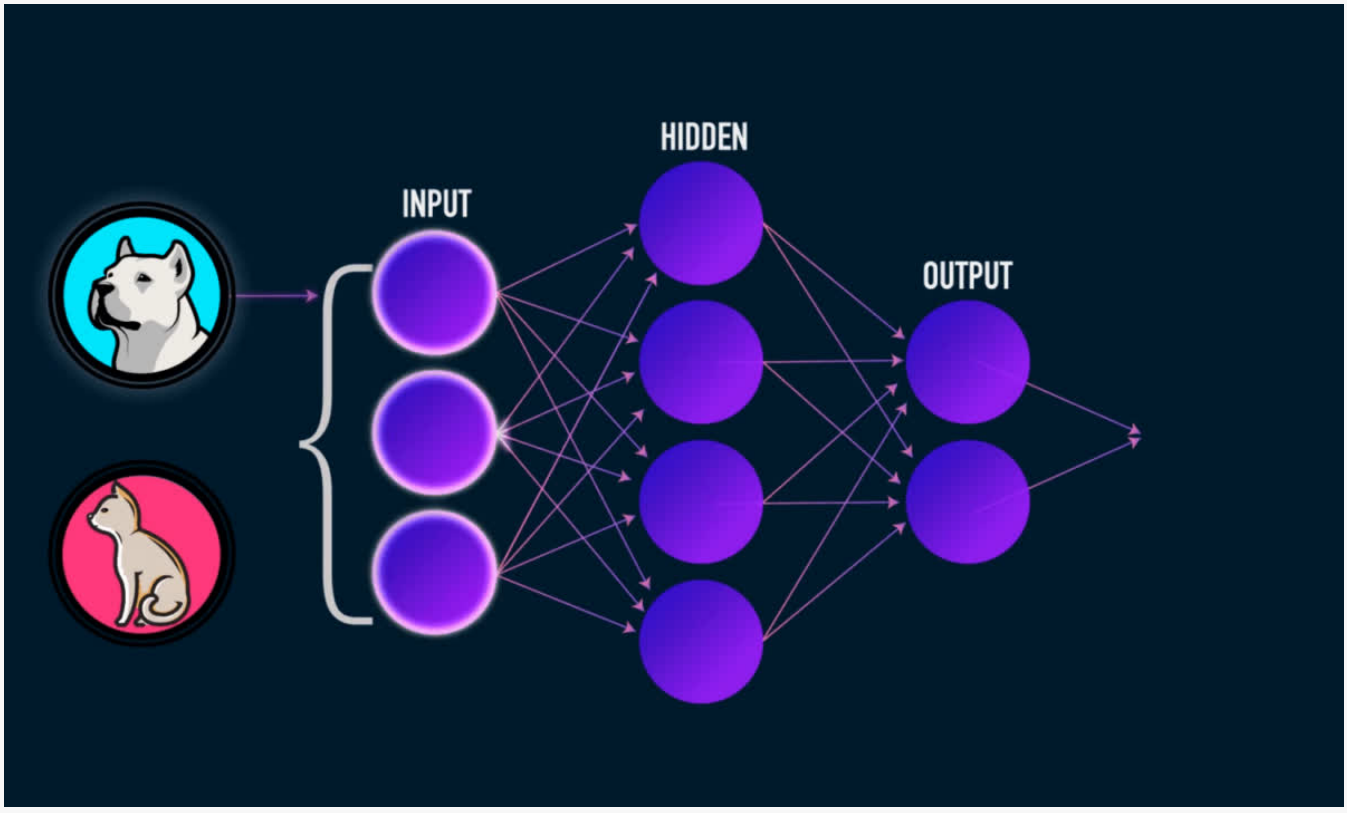
Motivação

A principal motivação do grupo ao escolher tal tema foi a junção das áreas de domínio de cada um, que resultou na junção de redes neurais com música.

1.Introdução

Redes neurais são sistemas de computação com nós interconectados que funcionam como os neurônios do cérebro humano. Usando algoritmos, elas podem reconhecer padrões escondidos e correlações em dados brutos, agrupá-los e classificá-los, e – com o tempo – aprender e melhorar continuamente.

Pela sua habilidade em aprender, uma rede neural deve ser treinada antes de ser aplicada ao seu fim, sendo assim, para a elaboração da Sophie foram necessários dois códigos, um para treinamento (Somente Devs tem acesso a esse código) e outro para aplicação da Rede Neural em si (Esse sim será usado pelos usuários).



Desenvolvimento do Código Interno

2.1 Introdução

O Código Interno (ou Código de Treinamento) esta encarregado da parte de configuração, elaboração e refinamento da Rede Neural, nele serão apresentados os tipos de músicas e respectivos Bancos de dados sobre cada um, assim será possível à Sophie "Entender seu papel". Após todo esse passos a RN será salva em um arquivo .h5

Todos os scripts do projeto foram desenvolvidos na linguagem Python 3.7, foram usadas as bibliotecas: Librosa, TensorFlow, Numpy.
Assim como toda a programação foi feita pela ferramenta Google Collab, da fase de prototipagem até o projeto final.

Código Interno
2.2 Importação de Bibliotecas

Importação De Bibliotecas

```
[ ]  %matplotlib inline

     from tensorflow import keras
     from tensorflow.keras.models import load_model
     import tensorflow as tf
     import numpy as np
     import librosa

     MusicasAnimadas = '/content/drive/MyDrive/ProjetoL/Animadas/Musica'
     MusicasCalmas = '/content/drive/MyDrive/ProjetoL/Calmas/Musica'
```

Nesse Trecho, todas as bibliotecas necessárias foram importadas para seus devidos fins:

TensorFlow: Auxilia na construção da estrutura da RN, na escolha de funções de ativações e na escolha da quantia de neurônios por camada.

Numpy: Auxilia no manuseio de dados, possui arrays otimizados oque consede uma maior facilidade de processamento.

Librosa: Utilizada principalmente no manuseio das músicas.

Todas são bibliotecas necessárias para o funcionamento do código, logo abaixo estão identificados os locais da memória que se encontram as músicas usadas no treinamento da rede.

Codigo Interno

2.3 Configuração dos DataSets

Nesse trecho serão armazenados os DataSets contendo os dados das músicas escolhidas para o treinamento, vale salientar as variáveis "Intervalo" e "Numusicas" que consistem no intervalo de tempo que a música será saçva para treino e a quantidade de músicas usadas.

O intervalo de 220500 equivale a 10 segundos de uma música na frequência 22050hz.
O número de músicas é 14, serão usadas 14 amostras de músicas Animadas e Calmas.

As listas presentes no código (Os parâmetros que terminam com "[ ]") serão usados para armazenamento das informações das músicas, sendo divididos por tipo de música e função, os de Teste serão usados para avaliar o modelo final, enquanto os de Treino serão utilizados para o treino do modelo.

A estrutura "For" na linha 10 será usada para repetição do trecho abaixo para cada música das 14 escolhidas

Será usada a biblioteca librosa para adquirir os dados da música, e separa os intervalos das músicas nas listas de Teste e Treino diferenciando o tipo de música, sempre usando o mesmo intervalo de 22050 Hz para as duas

E por último algumas variáveis de backup com intuito de prevenir futuros erros

Configurando DataSets

```
[ ]  listaCalmaTreino=[]
     listaAnimadaTreino=[]

     listaCalmaTeste=[]
     listaAnimadaTeste=[]

     intervalo = 220500
     tempo = 5
     Numusicas = 5

     for x in range(1,Numusicas+1):
       MscAnim = MusicasAnimadas+str(x) + '.wav'
       MscCalm = MusicasCalmas+str(x)   + '.wav'
       a,b = librosa.load(MscCalm)
       listaCalmaTreino.append(a[0:intervalo])
       listaCalmaTeste.append(a[intervalo:((intervalo)+(tempo*22050))])
       print('.',end='')
       a,b = librosa.load(MscAnim)
       listaAnimadaTreino.append(a[0:intervalo])
       listaAnimadaTeste.append(a[intervalo:((intervalo)+(tempo*22050))])
       print('.',end='')
       print(x)


     BDATR = listaAnimadaTreino.copy()
     BDCTR = listaCalmaTreino.copy()
     BDCT = listaAnimadaTeste.copy()
     BDAT = listaCalmaTeste.copy()
```

Codigo Interno

2.4 Normalização de Dados

Nesse trecho, será feita a normalização dos dados, O objetivo da normalização é alterar os valores das colunas numéricas no conjunto de dados para uma escala comum, sem distorcer as diferenças nos intervalos de valores. Para o aprendizado de máquina isso é essencial pois ajuda também na otimização e maior precisão do modelo

Normalização de Dados

```
[ ]  def normalize(v):
         norm = np.linalg.norm(v)
         if norm == 0:
            return v
         return v / norm
```

```
listaAnimadaTeste = [[] for x in range(len(listaAnimadaTeste))]
listaCalmaTeste = [[] for x in range(len(listaCalmaTeste))]

listaAnimadaTreino = [[] for x in range(len(listaAnimadaTreino))]
listaCalmaTreino = [[] for x in range(len(listaCalmaTreino))]

cont = 0
for x in BDCTR:
  listaCalmaTreino[cont] = normalize(np.asarray(x))
  cont+=1
cont = 0
for x in BDATR:
  listaAnimadaTreino[cont] = normalize(np.asarray(x))
  cont+=1
cont = 0
for x in BDCT:
  listaCalmaTeste[cont] = normalize(np.asarray(x))
  cont+=1
cont = 0
for x in BDAT:
  listaAnimadaTeste[cont] = normalize(np.asarray(x))
  cont+=1
```

Para normalizar os dados primeiro foi feita uma "def" com o objetivo de facilitar um pouco o processo, com isso o resto do processo se repete de acordo com cada DataSet a ser configurado, mas de forma resumida esse trecho vai simplesmente pegar cada espaço reservado nas listas, normalizar, e colocar de volta.

Codigo Interno

2.5 Formação de DataSet X e Y de Treino

Nesse trecho, será feita a formação do DataSet X e Y de treino do modelo, para facilitar a explicação Toma-se de exemplo um aluno estudando um livro, ele lê as perguntas e depois analisa as respostas, o mesmo se repete em RN, as "Perguntas" são o DataSet X, as "Resposta" são o DataSet Y

Formação DataSet X de Treino

```
DataSetTreinox = []
DataSetTreinoy = []

auxA = []
auxC = []

auyA = []
auyC = []


for a in listaCalmaTreino:
        lista = []
        lista.append(a)
        lista = np.array(lista)
        auxC.append(lista)
for a in listaAnimadaTreino:
        lista = []
        lista.append(a)
        lista = np.array(lista)
        auxA.append(lista)

for b in listaCalmaTreinoR:
        auyC.append(b)
for b in listaAnimadaTreinoR:
        auyA.append(b)

for x in range(len(auyA)):
    DataSetTreinoy = list(DataSetTreinoy) +

for x in range(len(auxA)):
    DataSetTreinox = list(DataSetTreinox) +

DataSetTreinoy = np.array(DataSetTreinoy)
DataSetTreinox = np.array(DataSetTreinox)

print("Formação DataSet de Treino Feito")
```

Formação de DataSet Y de Treino

```
for x in range(len(listaAnimadaTeste)):
    listaCalmaTeste[x] = np.asarray(listaCalmaTeste[x]).astype("float32")
    listaAnimadaTeste[x] = np.asarray(listaAnimadaTeste[x]).astype("float32")
    listaCalmaTreino[x] = np.asarray(listaCalmaTreino[x]).astype("float32")
    listaAnimadaTreino[x] = np.asarray(listaAnimadaTreino[x]).astype("float32")

print("Processamento de Dados Feito")

listaAnimadaTreinoR=[np.ones((1,1)) for x in listaCalmaTreino]
listaCalmaTreinoR=[np.zeros((1,1)) for x in listaAnimadaTreino]
listaAnimadaTesteR=[np.ones((1,1)) for x in listaCalmaTeste]
listaCalmaTesteR=[np.zeros((1,1)) for x in listaAnimadaTeste]

print("Configuração de Labels de Resposta Feita")
```

No DataSet X foram organizadas as músicas em "zig-zag", sendo colocada uma musica animada após uma calma e assim por diante

No DataSet Y só foram organizadas as respostas esperadas, sendo assim, por ser uma rede classificadora, so foi preciso colocar 1 e 0, sendo 1 para músicas animadas e 0 para músicas calmas

Codigo Interno

2.6 Classificação e Transfer learning

Nesse trecho, será feita a formação da RN em si, usando todos os recursos disponíveis pelo tensorflow.

```
[ ]  activationFunction='relu'
     model = keras.Sequential([
         keras.Input(shape=(intervalo,)),
         keras.layers.Dense(256,activation=activationFunction),
         keras.layers.Dense(256,activation=activationFunction),
         keras.layers.Dense(256,activation=activationFunction),
         keras.layers.Dense(1,activation=activationFunction)
     ])
     model.compile(optimizer='adam',loss='mse',metrics=['accuracy'])
     model.summary()

     model.fit(DataSetTreinox, DataSetTreinoy, epochs = 25)
```

Primeiro foi escolhida a Função de Ativação, o. A escolha das funções de ativação de uma rede neural são uma consideração importante uma vez que definem como devem ser seus dados de entrada, no caso a escolhida foi a 'relu'.

Após, o modelo foi configurado como sequencial. O modelo sequencial permite inserir camadas de uma rede neural em série, onde o output da primeira camada serve como input da segunda, e assim por diante, a camada de entrada é a o intervalo de tempo a ser analisado em cada música (10 segs ou 220500 Hz), 3 Camadas escondidas de 256 neurônios, todos usando a função relu para ativação, e 1 camada de saída apenas (Pois espera-se apenas um número para saída, 1 ou 0).

Daí, para compilar o modelo foi utilizado o otimizador Adam, e só então definido o numero de épocas para treinamento, épocas se classifica por quantas vezes o algoritmo vai ler e reler o livro até entender, no caso, 25 vezes.

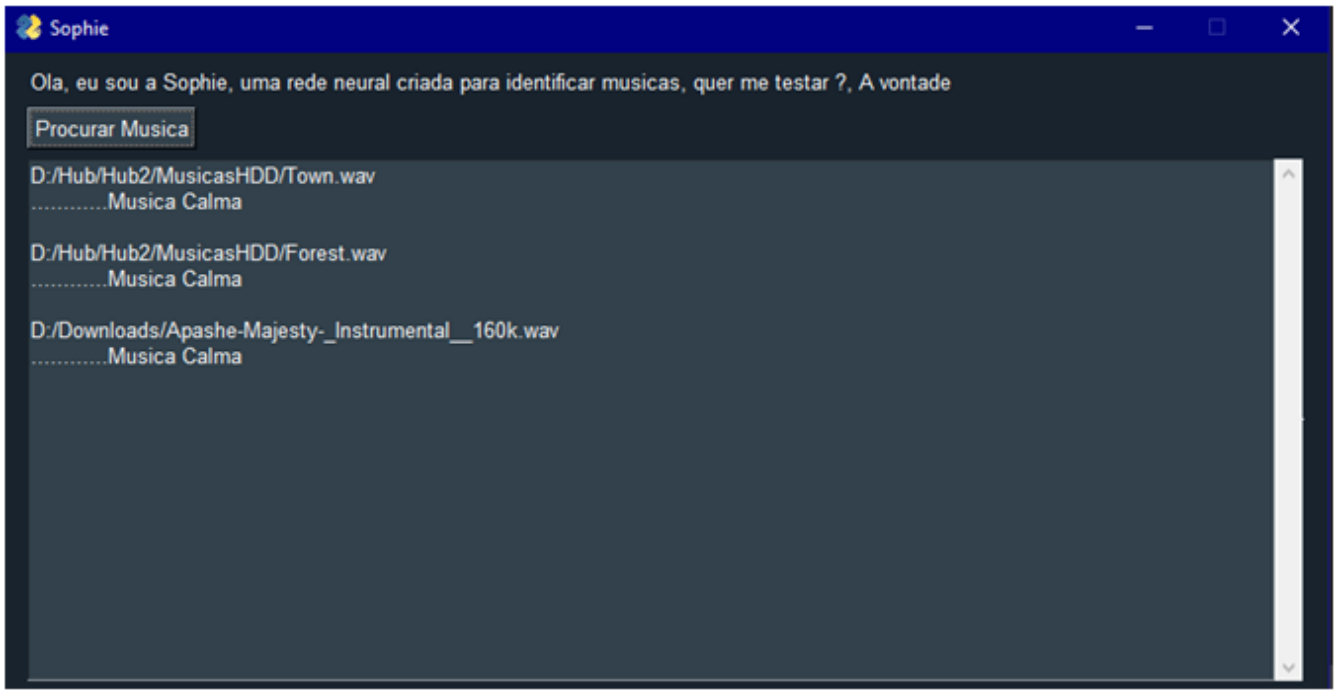
```
model.save('/content/drive/MyDrive/ProjetoL/Treinos/TreinoFinal/ModeloFinal2.h5');
```

Após isso, a Rede Neural é testada e então salva num lugar adequado.

Codigo Externo

3.1 Introdução

O Código Externo esta encarregado de ajudar o usuário a interagir com a RN Sophie sem precisar necessáriamente treina-la antes de casa uso graças ao arquivo salvo na última etapa do código Interno, esse script será utilizado diretamente pelo usuário e por isso deve ser bem fácil de entender, esse código foi desenvolvido na api "Spyder" do Mini conda



Codigo Externo

3.2 Visão Geral

Esse script tem basicamente uma versão resumida do código externo com a ausencia do treinamento.

```
import PySimpleGUI as sg
from IPython.display import Audio
import matplotlib.pyplot as plt
import librosa.display as ld
from tensorflow import keras
import tensorflow as tf
#from PySimpleGUI import PySimpleGUI as sg
from tensorflow.keras.models import load_model
import numpy as np
import sklearn
import librosa
import os.path
import time
from time import sleep as ts

def println(a,t=0.1):
    for b in a:
        ts(t)
        print(b, end='')
    print('\n')

def normalize(v):
    norm = np.linalg.norm(v)
    if norm == 0:
       return v
    return v / norm

intervalo = 220500
```

Importação de bibliotecas e defs necessárias e definição da variável "Intervalo"

```
model = load_model('D:\Downloads\ModeloFinal2.h5')
sg.theme('Dark Grey 8')
layout = [
         [sg.Text('Ola, eu sou a Sophie, uma rede neural criada para identificar musicas, quer me testar ?, A vontade')],
         [sg.Button('Procurar Musica')],
         [sg.Output(size=(110, 20), font=('Helvetica 10'))]
]

janela = sg.Window('Sophie',layout)

while True:
  eventos, valores = janela.read()
  if eventos == sg.WINDOW_CLOSED:
    break

  if eventos == "Procurar Musica":
#      sg.popup_get_file('filename to open', no_window=True)
      file = sg.popup_get_file('Escolha a Musica')
      cmds = file
      print(f'{file}',flush=True)
```

Configuração de Interface gráfica, a instauração de um botão com nome "Procurar Música" e alguns eventos não muito relevantes, ênfase no primerio comando, ele esta importando a RN salva anteriormente.

Codigo Externo

3.3 Classificação

Essa é a parte chave do código, pois ela será a responsável pela classificação das músicas, e funciona de uma forma diferente do Código Interno

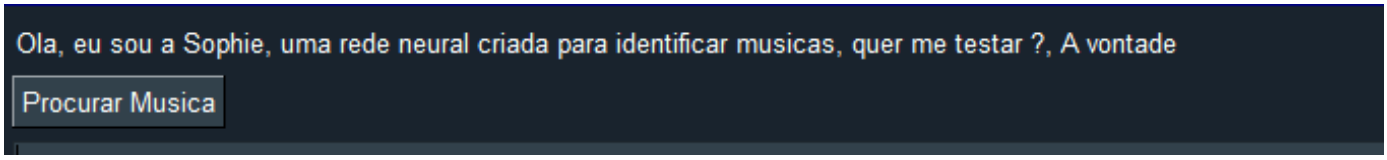
```
dtx,a = librosa.load(file)
DataSetTestex=[]
for y in range(1,13):
    DataSetTestex.append(normalize(dtx[intervalo*(y-1):intervalo*y]).reshape([1,intervalo]))

cont1 = 0
soma = 0
div = 0
for y in DataSetTestex:
    pp = model.predict(y)
    print('.', end = '')
    soma+=pp
    div+=1
total = soma/div
if total >=0.6:
    print("Musica Animada\n")
else:
    print("Musica Calma\n")
```

Nota-se de cara alguns comando gigantes, o 4° comando de cima a baixo vai importar a música, reorganizar suas tabelas e já normalizar os dados de uma vez só, ao final, terá uma lista com toda a musica separada em 10 segundos, Sophie analisará cada um e fazer média entre todos, pra de acordo com eu resultado classificar como Animada ou Calma

Produto/Projeto

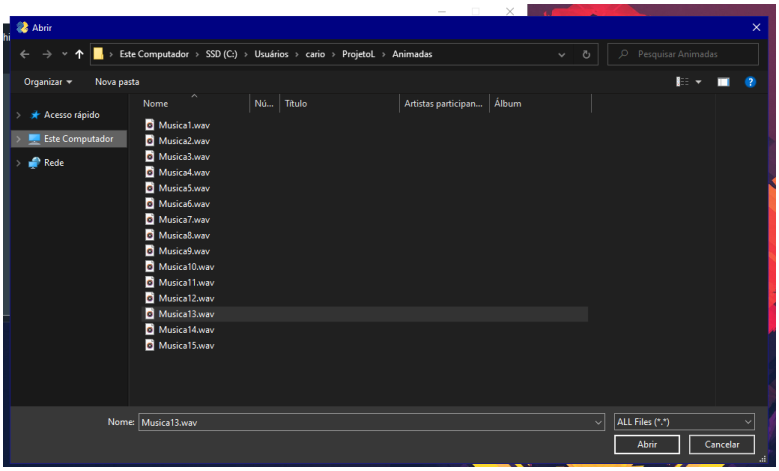
4. Como Usar

Ao iniciar o programa nota-se uma janela, nela só um botão de escolher música.



O usuário deve pressionar o botão para que abra a janela de seleção de música.

Ao apertar em "Browse" abrirá o gerenciador de arquivo, nesse momento espera-se que o usuário selecione a música que deseja classificar.



Daí só resta selecionar a opção "Abrir" e esperar umpouco e logo aparecerá a classificação da música.