Classificação de Imagens com Deep Learning

Bot Discord Classificador de Imagens

Daniel McCarthy, Lorenzo Canelhas, Lucas Henrique, Thaysa Zille.

[1]Centro Universitário Senac, São Paulo, SP, Brasil.

07 de Junho de 2022

Resumo: A utilização de uma classificação de imagens com Deep Learning mostra-se muito útil e eficiente, classificações como raça de cachorro podem ser modeladas e realizadas por algorítimos eficientes. Por meio de um Data Set tornou-se possível realizar um aprendizado onde pode-se classificar a raça de um cachorro, dentre 120 raças. Para interação com o usuário, o Bot de classificação foi disponibilizado na plataforma Discord.

***Palavras-chave*:** Deep Learning, Data Set, classificação,

***Keywords*:** *key;words;english*

1 Introdução

O presente trabalho tem o objetivo de mostrar de forma pratica, uma modelagem e implementação de um código que utiliza da API Keras[**Keras**], para realizar um aprendizado de máquina, onde atravéz do mesmo é capaz de classificar imagens e a partir desse vetor imagem, relacionar a uma raça.

Foi utilizado um Data Set disponibilizado pelo Kaggle, onde por meio do mesmo, podemos criar um modelo passível de aprendizado baseando-se nas relações coletadas dentro do Data Set. Realizado a captura das informações, faz-se então necessário o treinamento do modelo para que o processo se torne completo e por fim obtenha-se um classificador eficiente.

Observa-se que há nesse resultado diversas variáveis que influenciam o fluxo do sistema, como o tempo de treinamento, as funções usadas, que serão apresentadas no desenvolvimento do trabalho.

2 imports

Fizemos imports de algumas bibliotecas, sendo essas: numpy, pandas, tensorflow, os, matplotlib.pyplot, gc, seaborn e cv2. Também pegamos algumas funções específicas de bibiliotecas, como por exemplo: do random importamos o randint, do tqdm importamos o próprio tqdm, do subprocess importamos check$_output, assimcomooutrosqueestarolistadoslogoabaixojuntosdestes$.

```
    import numpy as np
import pandas as pd
import numpy as np
import tensorflow
import os
import matplotlib.pyplot as plt
import gc
```

import seaborn as sns
import cv2
from matplotlib.image import imread
import seaborn as sns
import random
from PIL import Image
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout, Conv2D, MaxPool2D
from tensorflow.keras.utils import to$_categorical$
$fromsklearn.model_selectionimporttrain_test_split$

3 Modelo e treinamento

Primeiramente, para podermos efetivamente classificar as imagens, pegamos o data set escolhido para que o modelo pudesse ser treinado e calibrado. Para que o modelo conseguisse generalizar a classificação, foi necessário um enorme conjunto de dados de cada uma das categorias do modelo.

Seguindo para o Treinamento, ainda temos diversas funções para que ele possa ocorrer corretamente. Que são as seguintes:

4 Implementação Bot do Discord

Carregar modelo e classes de placas
model = load-model('ModelDAv.h5')

boardClasses = ['Speed limit (20km/h)', 'Speed limit (30km/h)', 'Speed limit (50km/h)', 'Speed limit (60km/h)', 'Speed limit (70km/h)', 'Speed limit (80km/h)', 'End of speed limit (80km/h)', 'Speed limit (100km/h)', 'Speed limit (120km/h)', 'No passing', 'No passing for vechiles over 3.5 metric tons', 'Right-of-way at the next intersection', 'Priority road', 'Yield', 'Stop', 'No vechiles', 'Vechiles over 3.5 metric tons prohibited', 'No entry', 'General caution', 'Dangerous curve to the left', 'Dangerous curve to the right', 'Double curve', 'Bumpy road', 'Slippery road', 'Road narrows on the right', 'Road work', 'Traffic signals', 'Pedestrians', 'Children crossing', 'Bicycles crossing', 'Beware of ice/snow', 'Wild animals crossing', 'End of all speed and passing limits', 'Turn right ahead', 'Turn left ahead', 'Ahead only', 'Go straight or right', 'Go straight or left', 'Keep right', 'Keep left', 'Roundabout mandatory', 'End of no passing', 'End of no passing by vechiles over 3.5 metric tons']

Criação do cliente

client = discord.Client() client = commands.Bot(command-prefix='.')

newline Registro de evento e preparação do bot
registra o evento
@client.event
Bot pronto para começar a ser usado
async def on-ready():
print('Conectado 0.user'.format(client))

Comandos no discord e análise da imagem dispara sempre que uma mensagem não autoral é recebida
@client.event
async def on-message(message):
if message.author == client.user:
return
if message.content.startswith('oi') :
$awaitmessage.channel.send('Eaemeuconsagrado'$)

image$_s hape = (50, 50)$
$img = Image.open(dir_p ath +' to.jpg').convert('RGB')$
$img = img.resize(image_s hape)$
$img = np.expand_d ims(img, axis = 0)$
$img = np.array(img)$

index = (np.argmax(predictions, axis = 1))
board = boardClasses[int(index)]

Considerações:

Modelo = 17 epochs, porque modelos com 25 epochs ou mais acabam tendo um overfit maior (loss maior), a acuracia continua dentro da casa de 99Menos epochs mantém a acuracia de 99

Conclusão

Conseguimos fazer o programa a guardar os padrões através da análise dos pixels das imagens já pré-definidas. Pegamos as imagens e convertemos todos os dados dos pixels para formato de array. Nisso criamos os modelos que vão guardando os padrões de diversas imagens à serem reconhecidas, gerando então o aprendizado em si. Como resultado obtivemos sucesso e a imagem que foi selecionada foi reconhecida pelo programa, mesmo não tendo exatamente 100