Lab 5 - BCC406

REDES NEURAIS E APRENDIZAGEM EM PROFUNDIDADE

Redes de Convolução (CNN)

Prof. Eduardo e Prof. Pedro

Objetivos:

- Uso de modelos para biometria.
- Uso de modelos pré-treinados para biometria.
- Notebook baseado em tensorflow e Keras.

Data da entrega : 23/09

- Execute todo notebook e salve tudo em um PDF **nomeado** como "NomeSobrenome-LabX.pdf"
- Envie o PDF via google FORM

Biometria

Biometria nada mais é do que uma medida biológica, do grego *bios* - vida, e *metricos* - medida. O conceito de biometria surgiu em 1858, contudo o seu uso como tecnologia de segurança data de 1972. Essa abordagem sobrepuja o sistema de login-senha, pois se baseia em características únicas do indivíduo e que são difíceis de se copiar.

É possível realizar o reconhecimento de um indivíduo por meio de técnicas computacionais em conjunto com alguma modalidade biométrica. Para este processo dá-se o nome de Sistemas Biométricos. Várias partes do corpo humano podem ser usadas para a realização do reconhecimento, como, por exemplo: face, íris e a impressão digital, sendo esta considerada como a primeira biometria usada. Ela não se restringe somente a características físicas, mas também comportamentais (forma de andar) ou a união de ambas.

A prática de hoje envolve a biometria (classificação) de uma base de olhos. Você deve desenvolver uma rede neural para resolver o problema. Você pode propor uma nova rede ou fazer *transfer learning* de uma rede existente.

Os resultados devem ser postados nesse Google Sheets. O aluno com maior **acurácia** terá um ponto extra. Resultados distantes do melhor resultado serão penalizados. O melhor resultado será comparado com um *baseline*.

A base de dados está disponível no Drive da disciplina na pasta de *datasets/eye*. Dentro da pasta, há uma pasta *test* e *train*. A primeira são as imagens que devem ser usadas para reportar os resultados no Google Sheets e a segunda usada para treinar o modelo. Dentro de cada uma das duas pastas, há outras 50 pastas (labels), cada uma com imagens de cada classe.

Código

```
# Importações
import numpy as np
import os
import cv2
from keras.models import Sequential
from tensorflow.keras import layers, models
from sklearn.metrics import classification_report
from keras.layers import Conv2D, MaxPool2D, Dropout
from tensorflow.keras.applications import ResNet152V2, MobileNet
from keras.applications.densenet import DenseNet201
```

```
# Ligar ao Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```

```
# Ler base de dados
data_shape = (224, 224)

root_dir = '/content/drive/My Drive/eyes/train/'
files = [os.path.join(path, name) for path, subdirs, files in os.walk(root_dir) for name i
train_x = np.asarray([cv2.resize(cv2.imread(name, cv2.IMREAD_COLOR), data_shape, interpola
train_y = np.asarray([int(name[-10:-7]) for name in files])

root_dir = '/content/drive/My Drive/eyes/test/'
files = [os.path.join(path, name) for path, subdirs, files in os.walk(root_dir) for name i
test_x = np.asarray([cv2.resize(cv2.imread(name, cv2.IMREAD_COLOR), data_shape, interpolat
test_y = np.asarray([int(name[-10:-7]) for name in files])
```

```
# Tamanho da entrada e número de classes
input_size = (train_x.shape[1], train_x.shape[2],train_x.shape[3])
n_classes = 51
```

```
# Montar Modelo 1

model1 = models.Sequential()

model1.add(layers.InputLayer(input_shape=input_size))
model1.add(MobileNet(include_top=False, input_shape=input_size))
```

```
model1.add(Conv2D(256, 2, 2, padding="same", activation="relu"))
model1.add(Dropout(0.05))
model1.add(Conv2D(4,2,2))

"""model1.add(Conv2D(512, 3, padding="same", activation="relu"))
model1.add(Conv2D(512, 3, padding="same", activation="relu"))
model1.add(Conv2D(256, 3, 2, padding="same", activation="relu"))
model1.add(Conv2D(256, 2, 2, padding="same", activation="relu"))
model1.add(Dropout(0.05))
model1.add(Conv2D(4,2,2))"""

model1.add(layers.Flatten())

model1.add(layers.Dense(n_classes, activation='softmax', name='CamadaClassificacao'))

model1.summary()

# Compilar
model1.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accurac

#Treinar
model1.fit(train_x, train_y, epochs=70)

#Obter Resultados
y_pred = model1.predict(test_x, verbose=1)
y_pred_bool = np.argmax(y_pred, axis=1)

print(list(y_pred_bool))
print(list(test_y.reshape(-1)))

print(classification_report(test_y.reshape(-1), y_pred_bool))
#print("\n\nAcurácia: ", sum([ 1 for i in zip(y_pred_bool, test_y.reshape(-1)) if i[0] ==
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 mobilenet_1.00_224 (Functio  (None, 7, 7, 1024)       3228864
 nal)

 conv2d_12 (Conv2D)          (None, 4, 4, 256)         1048832

 dropout_5 (Dropout)         (None, 4, 4, 256)         0

 conv2d_13 (Conv2D)          (None, 2, 2, 4)           4100

 flatten_5 (Flatten)         (None, 16)                0

 CamadaClassificacao (Dense)  (None, 51)               867

=================================================================
Total params: 4,282,663
Trainable params: 4,260,775
Non-trainable params: 21,888
_________________________________________________________________
Epoch 1/70
6/6 [==============================] - 3s 162ms/step - loss: 4.1713 - accuracy: 0.
```

```
Epoch 2/70
6/6 [==============================] - 1s 159ms/step - loss: 0.7536 - accuracy: 0.
Epoch 3/70
6/6 [==============================] - 1s 159ms/step - loss: 0.1538 - accuracy: 0.
Epoch 4/70
6/6 [==============================] - 1s 159ms/step - loss: 0.0568 - accuracy: 0.
Epoch 5/70
6/6 [==============================] - 1s 160ms/step - loss: 0.0343 - accuracy: 0.
Epoch 6/70
6/6 [==============================] - 1s 160ms/step - loss: 0.0091 - accuracy: 1.
Epoch 7/70
6/6 [==============================] - 1s 161ms/step - loss: 0.0610 - accuracy: 0.
Epoch 8/70
6/6 [==============================] - 1s 162ms/step - loss: 0.0090 - accuracy: 1.
Epoch 9/70
6/6 [==============================] - 1s 160ms/step - loss: 0.0029 - accuracy: 1.
Epoch 10/70
6/6 [==============================] - 1s 159ms/step - loss: 0.0030 - accuracy: 1.
Epoch 11/70
6/6 [==============================] - 1s 162ms/step - loss: 0.0015 - accuracy: 1.
Epoch 12/70
6/6 [==============================] - 1s 158ms/step - loss: 5.9821e-04 - accuracy
Epoch 13/70
6/6 [==============================] - 1s 159ms/step - loss: 5.2100e-04 - accuracy
Epoch 14/70
6/6 [==============================] - 1s 156ms/step - loss: 4.6537e-04 - accuracy
Epoch 15/70
6/6 [==============================] - 1s 164ms/step - loss: 6.1284e-04 - accuracy
Epoch 16/70
6/6 [==============================] - 1s 159ms/step - loss: 3.6607e-04 - accuracy
Epoch 17/70
6/6 [==============================] - 1s 162ms/step - loss: 2.4982e-04 - accuracy
Epoch 18/70
```