Treinamento

Nesse documento está o treinamento que foi utilizado nas redes neurais do trabalho

Importando bibliotecas

O primeiro passo do nosso treinamento é instalar o tensorflow e importar as bibliotecas que utilizaremos

In [58]:
```
pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\programdata\anaconda3\lib\site-packa
ges (2.7.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\programdata\anaconda3\lib\s
ite-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in c:\programdata\anaconda3\lib\site-pack
ages (from tensorflow) (3.2.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\programdata\anaconda3\lib\s
ite-packages (from tensorflow) (1.42.0)
Requirement already satisfied: wheel<1.0,>=0.32.0 in c:\programdata\anaconda3\lib\si
te-packages (from tensorflow) (0.37.0)
Requirement already satisfied: six>=1.12.0 in c:\programdata\anaconda3\lib\site-pack
ages (from tensorflow) (1.16.0)
Requirement already satisfied: gast<0.5.0,>=0.2.1 in c:\programdata\anaconda3\lib\si
te-packages (from tensorflow) (0.4.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\programdata\anaconda3\lib\sit
e-packages (from tensorflow) (3.3.0)
Requirement already satisfied: tensorflow-estimator<2.8,~=2.7.0rc0 in c:\programdata
\anaconda3\lib\site-packages (from tensorflow) (2.7.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in c:\programdata\anaconda
3\lib\site-packages (from tensorflow) (1.1.2)
Requirement already satisfied: wrapt>=1.11.0 in c:\programdata\anaconda3\lib\site-pa
ckages (from tensorflow) (1.12.1)
Requirement already satisfied: libclang>=9.0.1 in c:\programdata\anaconda3\lib\site-
packages (from tensorflow) (12.0.0)
Requirement already satisfied: absl-py>=0.4.0 in c:\programdata\anaconda3\lib\site-p
ackages (from tensorflow) (1.0.0)
Requirement already satisfied: termcolor>=1.1.0 in c:\programdata\anaconda3\lib\site
-packages (from tensorflow) (1.1.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\programdata\anaconda3\lib\sit
e-packages (from tensorflow) (1.6.3)
Requirement already satisfied: tensorboard~=2.6 in c:\programdata\anaconda3\lib\site
-packages (from tensorflow) (2.7.0)
Requirement already satisfied: numpy>=1.14.5 in c:\programdata\anaconda3\lib\site-pa
ckages (from tensorflow) (1.20.3)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.21.0 in c:\programdat
a\anaconda3\lib\site-packages (from tensorflow) (0.22.0)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\programdata\anaconda3
\lib\site-packages (from tensorflow) (3.10.0.2)
Requirement already satisfied: flatbuffers<3.0,>=1.12 in c:\programdata\anaconda3\li
b\site-packages (from tensorflow) (2.0)
Requirement already satisfied: protobuf>=3.9.2 in c:\programdata\anaconda3\lib\site-
packages (from tensorflow) (3.19.1)
Requirement already satisfied: keras<2.8,>=2.7.0rc0 in c:\programdata\anaconda3\lib
\site-packages (from tensorflow) (2.7.0)
Requirement already satisfied: markdown>=2.6.8 in c:\programdata\anaconda3\lib\site-
packages (from tensorboard~=2.6->tensorflow) (3.3.6)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\programdata\anaco
nda3\lib\site-packages (from tensorboard~=2.6->tensorflow) (1.8.0)
```

```
Requirement already satisfied: werkzeug>=0.11.15 in c:\programdata\anaconda3\lib\sit
e-packages (from tensorboard~=2.6->tensorflow) (2.0.2)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\programda
ta\anaconda3\lib\site-packages (from tensorboard~=2.6->tensorflow) (0.6.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\programdata\anaconda3\lib
\site-packages (from tensorboard~=2.6->tensorflow) (2.3.3)
Requirement already satisfied: requests<3,>=2.21.0 in c:\programdata\anaconda3\lib\s
ite-packages (from tensorboard~=2.6->tensorflow) (2.26.0)
Requirement already satisfied: setuptools>=41.0.0 in c:\programdata\anaconda3\lib\si
te-packages (from tensorboard~=2.6->tensorflow) (58.0.4)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\programdata\an
aconda3\lib\site-packages (from tensorboard~=2.6->tensorflow) (0.4.6)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\programdata\anaconda3\lib\site-pa
ckages (from google-auth<3,>=1.6.3->tensorboard~=2.6->tensorflow) (4.8)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in c:\programdata\anaconda3\li
b\site-packages (from google-auth<3,>=1.6.3->tensorboard~=2.6->tensorflow) (4.2.4)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\programdata\anaconda3\lib
\site-packages (from google-auth<3,>=1.6.3->tensorboard~=2.6->tensorflow) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\programdata\anaconda3
\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorf
low) (1.3.0)
Requirement already satisfied: importlib-metadata>=4.4 in c:\programdata\anaconda3\l
ib\site-packages (from markdown>=2.6.8->tensorboard~=2.6->tensorflow) (4.8.1)
Requirement already satisfied: zipp>=0.5 in c:\programdata\anaconda3\lib\site-packag
es (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard~=2.6->tensorflow) (3.
6.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\programdata\anaconda3\lib
\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard~=2.6-
>tensorflow) (0.4.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\programdata\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\si
te-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (2021.10.8)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\programdata\anaconda3
\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-pac
kages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (3.2)
Requirement already satisfied: oauthlib>=3.0.0 in c:\programdata\anaconda3\lib\site-
packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboa
rd~=2.6->tensorflow) (3.1.1)
Note: you may need to restart the kernel to use updated packages.
```

In [59]:

```
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

Importando Imagens

Após a adição das bibliotecas que vamos utilizar, importaremos as imagens do banco de dados.

In [60]:

```
import pathlib
data_dir = "banco de imagens/"
data_dir = pathlib.Path(data_dir)
```

Verificando imagens do banco

Essa parte é apenas para mostrar quantas imagens há no nosso banco de dados. Veja que não possui muita, isso por conta da quantidade que achamos de imagens de maçã podre que foram baixas.

In [61]:
```
image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
```

```
265
```

Treinando o modelo

Primeiro adicionamos os parametros para carregamento de imagens no keras

In [62]:
```
batch_size = 32
img_height = 180
img_width = 180
```

Depois dividimos a quantidade de dados que vai para o treinamento e o que vai para a validação. Que no caso da nossa base foi feito 25% de validação e 75% de treinamento

In [63]:
```
train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.25,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 291 files belonging to 3 classes.
Using 219 files for training.
```

In [64]:
```
val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.25,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 291 files belonging to 3 classes.
Using 72 files for validation.
```

In [65]:
```
class_names = train_ds.class_names
print(class_names)
```

```
['Maçãs boas', 'Maçãs podres', 'Negativo']
```

Ajustando configurações de desempenho

In [66]:
```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

In [ ]:

Padronizando dados

In [67]:
```
normalization_layer = layers.Rescaling(1./255)
normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
# Notice the pixel values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
```

```
0.0 1.0
```

In [68]:
```
data_augmentation = keras.Sequential(
  [
    layers.RandomFlip("horizontal",
                      input_shape=(img_height,
                                   img_width,
                                   3)),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
  ]
)
```

Criando o modelo

A baixo criamos o modelo, utilizando camadas do Keras

In [69]:
```
num_classes = len(class_names)

model = Sequential([
  data_augmentation,
  layers.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

Copilando Modelo

Na copilação utilizamos o otimizador Adam, por verificar que ele trazia melhores resultados que os outros para nosso trabalho.

In [70]:
```
model.compile(optimizer='Adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

Fazendo o treinamento do modelo

Primeiramente ajustamos o epochs que seria a quantidade de lotes a passar pelo modelo e depois o modelo com o epochs, o dado de treino e de validação.

In [71]:

```
epochs= 20
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/20
7/7 [==============================] - 5s 501ms/step - loss: 1.6325 - accuracy: 0.33
33 - val_loss: 1.1200 - val_accuracy: 0.1389
Epoch 2/20
7/7 [==============================] - 3s 440ms/step - loss: 1.1017 - accuracy: 0.34
70 - val_loss: 1.0560 - val_accuracy: 0.4722
Epoch 3/20
7/7 [==============================] - 3s 434ms/step - loss: 1.0857 - accuracy: 0.37
44 - val_loss: 1.0014 - val_accuracy: 0.5417
Epoch 4/20
7/7 [==============================] - 3s 431ms/step - loss: 1.0923 - accuracy: 0.40
18 - val_loss: 1.0343 - val_accuracy: 0.5972
Epoch 5/20
7/7 [==============================] - 3s 424ms/step - loss: 1.0609 - accuracy: 0.49
77 - val_loss: 0.9847 - val_accuracy: 0.5556
Epoch 6/20
7/7 [==============================] - 3s 416ms/step - loss: 1.0344 - accuracy: 0.52
51 - val_loss: 0.9495 - val_accuracy: 0.5833
Epoch 7/20
7/7 [==============================] - 3s 430ms/step - loss: 1.0064 - accuracy: 0.51
14 - val_loss: 0.9722 - val_accuracy: 0.5694
Epoch 8/20
7/7 [==============================] - 3s 442ms/step - loss: 0.9642 - accuracy: 0.51
60 - val_loss: 1.0005 - val_accuracy: 0.5833
Epoch 9/20
7/7 [==============================] - 3s 455ms/step - loss: 0.9235 - accuracy: 0.57
99 - val_loss: 0.9651 - val_accuracy: 0.5972
Epoch 10/20
7/7 [==============================] - 3s 427ms/step - loss: 0.8944 - accuracy: 0.56
62 - val_loss: 0.9534 - val_accuracy: 0.6111
Epoch 11/20
7/7 [==============================] - 3s 450ms/step - loss: 0.8651 - accuracy: 0.58
90 - val_loss: 1.1665 - val_accuracy: 0.5278
Epoch 12/20
7/7 [==============================] - 3s 444ms/step - loss: 0.8483 - accuracy: 0.58
45 - val_loss: 1.1726 - val_accuracy: 0.4861
Epoch 13/20
7/7 [==============================] - 3s 488ms/step - loss: 0.7933 - accuracy: 0.63
47 - val_loss: 1.0889 - val_accuracy: 0.5972
Epoch 14/20
7/7 [==============================] - 3s 438ms/step - loss: 0.7293 - accuracy: 0.68
49 - val_loss: 0.9946 - val_accuracy: 0.6806
Epoch 15/20
7/7 [==============================] - 3s 462ms/step - loss: 0.7671 - accuracy: 0.66
67 - val_loss: 0.9441 - val_accuracy: 0.6389
Epoch 16/20
7/7 [==============================] - 3s 474ms/step - loss: 0.7084 - accuracy: 0.69
41 - val_loss: 0.7834 - val_accuracy: 0.7222
Epoch 17/20
7/7 [==============================] - 3s 467ms/step - loss: 0.7284 - accuracy: 0.65
75 - val_loss: 1.2335 - val_accuracy: 0.5694
Epoch 18/20
```

```
7/7 [==============================] - 3s 470ms/step - loss: 0.6238 - accuracy: 0.78
54 - val_loss: 0.8247 - val_accuracy: 0.7083
Epoch 19/20
7/7 [==============================] - 3s 471ms/step - loss: 0.6113 - accuracy: 0.73
06 - val_loss: 1.0228 - val_accuracy: 0.6667
Epoch 20/20
7/7 [==============================] - 3s 443ms/step - loss: 0.6323 - accuracy: 0.71
23 - val_loss: 0.8626 - val_accuracy: 0.7639
```

Verificando acuracidade do modelo

In [ ]:

In [72]:

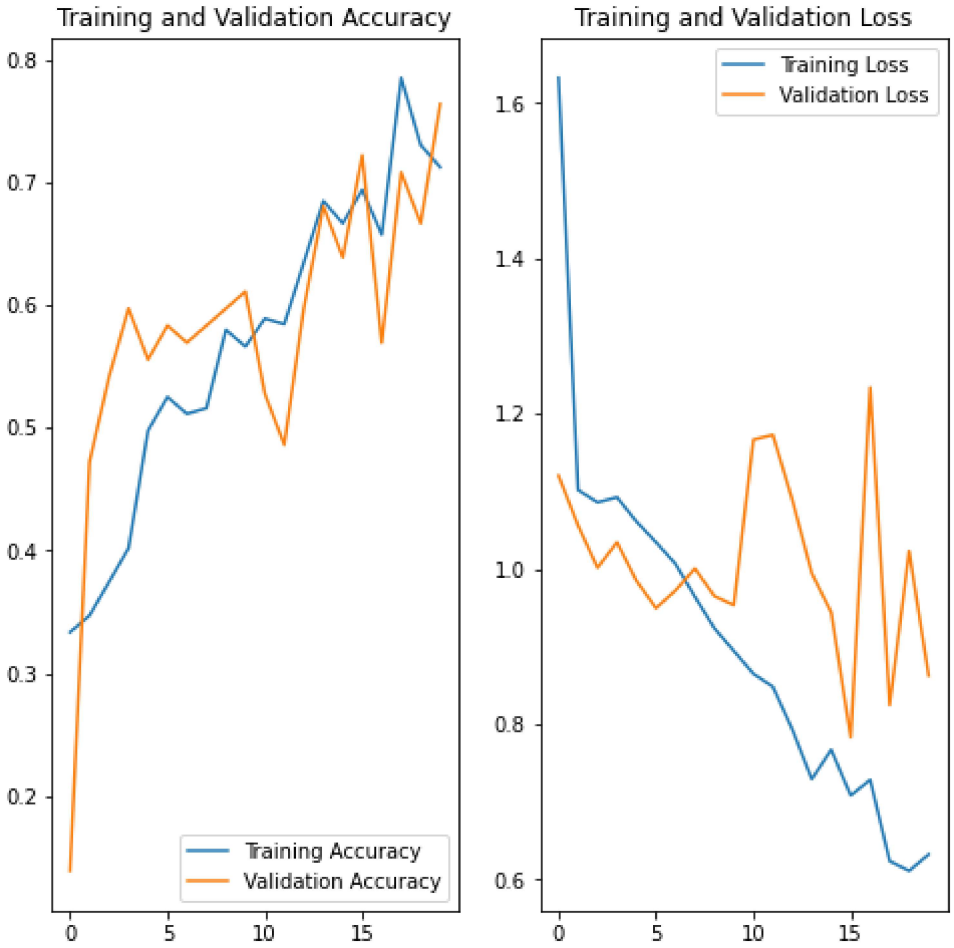
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Verificando com alguns exemplos

In [73]:
```
PIL.Image.open(str("gatin.jpg"))
```

Out[73]:

In [74]:
```
img = tf.keras.utils.load_img("gatin.jpg"
```

```
    , target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "Essa imagem é um(a) {} com {:.2f} por cento de confiança"
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
Essa imagem é um(a) Negativo com 81.07 por cento de confiança
```

In [76]:

```
PIL.Image.open(str("maçãboa.jpg"))
```

Out[76]:

In [77]:

```
img = tf.keras.utils.load_img("maçãboa.jpg"
    , target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "Essa imagem é um(a) {} com {:.2f} por cento de confiança"
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
Essa imagem é um(a) Maçãs boas com 97.39 por cento de confiança
```

In [87]:

```
PIL.Image.open(str("macaruim.jpg"))
```

Out[87]:

In [88]:
```
img = tf.keras.utils.load_img("macaruim.jpg"
    , target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "Essa imagem é um(a) {} com {:.2f} por cento de confiança"
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
Essa imagem é um(a) Maçãs podres com 89.56 por cento de confiança
```

Salvando Modelo

Nessa linha salvamos o modelo Keras em .h5, que nos permite tanto utilizar o modelo em outro lugar quanto continuar o treinamento em outra oportunidade.

In [82]:
```
model.save('modeloMaca.h5')
```

In [ ]:
```

```