```
from keras.models import load_model
import tensorflow as tf
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.resnet50 import ResNet50
from keras.applications import ResNet50V2  # adicionei isso
from keras.applications.resnet50 import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, f1_score, precision_score, rec
import time

import warnings
warnings.filterwarnings("ignore")


# Montando o google drive para acessar imagens
from google.colab import drive
drive.mount('/content/gdrive')
path = "/content/gdrive/MyDrive/Colab Notebooks/"
```

```
Mounted at /content/gdrive
```

Tamanho das imagens

```
from tensorflow.keras.layers import Flatten, Dense, Dropout, Conv2D, MaxPool
# re-size all the images to this
IMAGE_SIZE = [400, 400]
```

```
train_path = path+'ebhi-split-2categorias/train'
valid_path = path+'ebhi-split-2categorias/val'
test_path = path+'ebhi-split-2categorias/test'
```

Declarando o modelo resnet50v2

```
resnet = ResNet50V2(input_shape=(400, 400, 3),
                    weights='imagenet', include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-ap
94668760/94668760 [==============================] - 1s 0us/step
```

Não treina camadas

```
for layer in resnet.layers:
    layer.trainable = False
```

Adicionando as nossas camadas

```
x = Flatten()(resnet.output)
# x = Dense(64, activation='relu')(x) # descomentei e troquei de 1000 para 6
#prediction = Dense(len(folders), activation='softmax')(x)
prediction = Dense(1, activation='sigmoid')(x)
```

Criando o modelo

```
model = Model(inputs=resnet.input, outputs=prediction)
```

adicionei

model.add(Conv2D(64, (3,3), activation='relu'))

model.add(MaxPooling2D(2,2))

model.add(Flatten())

model.add(Dense(64, activation='relu'))

model.add(Dropout(0.4))

model.add(Dense(2, activation='softmax'))

ate aqui

Recursos ×

Você assinou Colab Pro. Saiba mais.
Disponíveis: 84.67 unidades de computação
Taxa de uso: aproximadamente 1.96 por hora
Você tem 1 sessão ativa. Gerenciar sessões

(GPU) de back-end do Google Compute Engine em Python 3
Mostrando os recursos de 14:35 a 14:59

RAM do sistema
5.6 / 12.7 GB

RAM da GPU
14.2 / 15.0 GB

Disco
31.2 / 166.8 GB

Visualizando a estrutura do modelo

```
model.summary()
```

```
 rmalization)

 conv5_block2_2_relu (Activ  (None, 13, 13, 512)          0         ['c
 ation)

 conv5_block2_3_conv (Conv2  (None, 13, 13, 2048)         1050624   ['c
 D)

 conv5_block2_out (Add)      (None, 13, 13, 2048)         0         ['c
                                                                     'c

 conv5_block3_preact_bn (Ba  (None, 13, 13, 2048)         8192      ['c
 tchNormalization)

 conv5_block3_preact_relu (  (None, 13, 13, 2048)         0         ['c
 Activation)                                                        ']

 conv5_block3_1_conv (Conv2  (None, 13, 13, 512)          1048576   ['c
 D)                                                                 0]

 conv5_block3_1_bn (BatchNo  (None, 13, 13, 512)          2048      ['c
 rmalization)

 conv5_block3_1_relu (Activ  (None, 13, 13, 512)          0         ['c
 ation)

 conv5_block3_2_pad (ZeroPa  (None, 15, 15, 512)          0         ['c
 dding2D)

 conv5_block3_2_conv (Conv2  (None, 13, 13, 512)          2359296   ['c
 D)

 conv5_block3_2_bn (BatchNo  (None, 13, 13, 512)          2048      ['c
 rmalization)

 conv5_block3_2_relu (Activ  (None, 13, 13, 512)          0         ['c
 ation)

 conv5_block3_3_conv (Conv2  (None, 13, 13, 2048)         1050624   ['c
 D)

 conv5_block3_out (Add)      (None, 13, 13, 2048)         0         ['c
                                                                     'c

 post_bn (BatchNormalizatio  (None, 13, 13, 2048)         8192      ['c
 n)

 post_relu (Activation)      (None, 13, 13, 2048)         0         ['p

 flatten (Flatten)           (None, 346112)               0         ['p

 dense (Dense)               (None, 1)                    346113    ['f

=========================================================================
Total params: 23910913 (91.21 MB)
Trainable params: 346113 (1.32 MB)
Non-trainable params: 23564800 (89.89 MB)
```

Declarando para o modelo as funções de custo e otimização

```
model.compile(
    loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
    optimizer='adam',
    metrics=['accuracy']
)
```

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range = 0.2, #comentei
                                   zoom_range=0.2,
                                   horizontal_flip=True)
```

```
test_datagen = ImageDataGenerator(rescale=1./255)
valid_datagen = ImageDataGenerator(rescale=1./255)
```

```
training_set = train_datagen.flow_from_directory(train_path,
                                                 target_size=(400, 400),
                                                 batch_size=32,
```

```
                                                 color_mode='rgb',  # adicio
                                                 class_mode='binary')
```

```
Found 1163 images belonging to 2 classes.
```

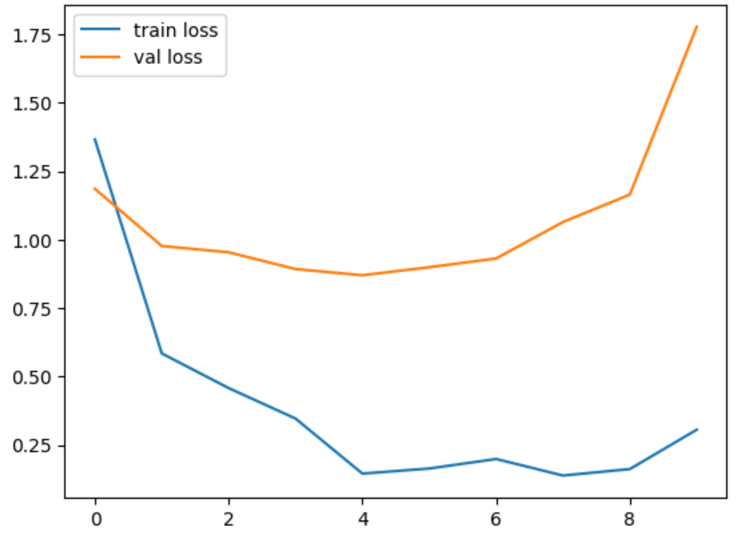
```
test_set = test_datagen.flow_from_directory(valid_path,
                                            target_size=(400, 400),
                                            batch_size=32,
                                            shuffle=False,
                                            color_mode='rgb',  # adicionei i
                                            class_mode='binary')
valid_set = valid_datagen.flow_from_directory(valid_path,
                                              target_size=(400, 400),
                                              batch_size=32,
                                              class_mode='binary')
```

```
Found 387 images belonging to 2 classes.
Found 387 images belonging to 2 classes.
```

Treinando o modelo

```
r = model.fit(
    training_set,
    validation_data=valid_set,
    epochs=10,
    steps_per_epoch=len(training_set),
    validation_steps=len(valid_set)
)
```

```
Epoch 1/10
37/37 [==============================] - 359s 9s/step - loss: 1.3663 - ac
Epoch 2/10
37/37 [==============================] - 82s 2s/step - loss: 0.5844 - acc
Epoch 3/10
37/37 [==============================] - 82s 2s/step - loss: 0.4578 - acc
Epoch 4/10
37/37 [==============================] - 81s 2s/step - loss: 0.3464 - acc
Epoch 5/10
37/37 [==============================] - 82s 2s/step - loss: 0.1452 - acc
Epoch 6/10
37/37 [==============================] - 84s 2s/step - loss: 0.1638 - acc
Epoch 7/10
37/37 [==============================] - 81s 2s/step - loss: 0.1986 - acc
Epoch 8/10
37/37 [==============================] - 81s 2s/step - loss: 0.1382 - acc
Epoch 9/10
37/37 [==============================] - 81s 2s/step - loss: 0.1617 - acc
Epoch 10/10
37/37 [==============================] - 81s 2s/step - loss: 0.3055 - acc
<matplotlib.legend.Legend at 0x787a2d72bcd0>
```

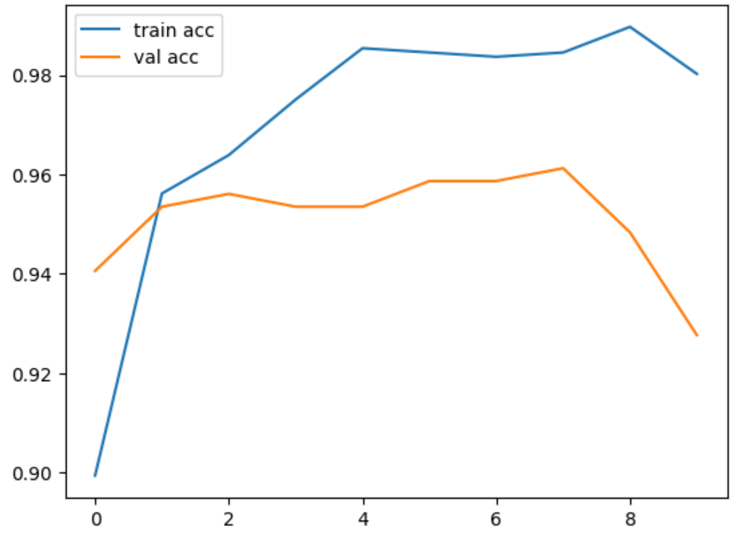



Perda do treino

```
# loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val_loss')
```

```
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.savefig('LossVal_loss_resnet')
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
<matplotlib.legend.Legend at 0x7879e9356470>
```




Acurácias do treino

```
# accuracies
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.savefig('AccVal_acc_resnet')
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

```
model.save('hist_model_resnet.h5')
```

Etapa de Testes

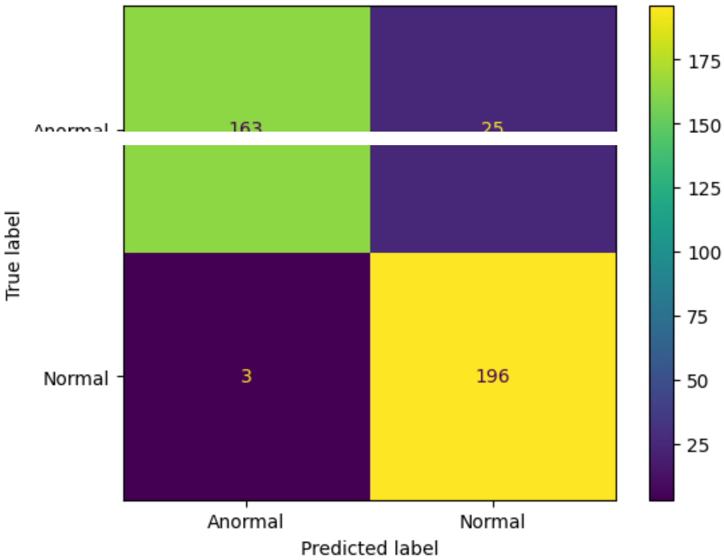
```
resnet_model = model

t = time.time()
# Usando o modelo para predição das amostras de teste
y_pred = resnet_model.predict(test_set)
# Reset
#test_set.reset()
#loss, acc = resnet_model.evaluate(test_set)
#aux = np.argmax(aux, axis=1)
y_pred = np.where(y_pred > 0.5, 1, 0).flatten()
print("y predito:")
print(y_pred)
y_true = test_set.classes
print("y real:")
print(y_true)
# Método para calcular o valor F1-Score
print('F1-Score: {}'.format(f1_score(y_true, y_pred, average='macro')))
# Método para calcular a Precision
print('Precision : {}'.format(precision_score(y_true, y_pred, average='macro
# Método para calcular o Recall
print('Recall: {}'.format(recall_score(y_true, y_pred, average='macro')))

print('Matriz de Confusão:')
cm = confusion_matrix(y_true, y_pred)
cm_display = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=['Ano
cm_display.plot()
plt.savefig('Matriz-resnet')
plt.show()

print ('Accuracy score: ', accuracy_score(y_true, y_pred))
#print('Acuracia obtida com o resnet no Conjunto de Teste EBHI: {:.2f}'.form
#    acc))
```

```
13/13 [==============================] - 11s 784ms/step
y predito:
[0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 1 1 0 0 0 0 1 1 0 0 0 0 0 0
 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
y real:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
F1-Score: 0.9271186440677966
Precision : 0.934402769448836
Recall: 0.9259729498556613
Matriz de Confusão:
```




```
Accuracy score:  0.9276485788113695
```

Avaliando no UnitoPatho

```
test_path_uni = path+'/dataset-unitopatho/'
test_datagen_uni = ImageDataGenerator(rescale=1./255)
test_set_uni = test_datagen_uni.flow_from_directory(test_path_uni,
                                            target_size=(400, 400),
                                            batch_size=32,
                                            shuffle=False,
                                            class_mode='binary',classes=['ANOR
```

```
y_pred = resnet_model.predict(test_set_uni)
# Reset
#test_set_uni.reset()
#loss, acc = resnet_model.evaluate(test_set_uni)
#aux = np.argmax(aux, axis=1)
y_pred = np.where(y_pred > 0.5, 1, 0).flatten()
print("y predito:")
print(y_pred)
y_true = test_set_uni.classes
print("y real:")
print(y_true)
# Método para calcular o valor F1-Score
```

```
print('F1-Score: {}'.format(f1_score(y_true, y_pred, average='macro')))
# Método para calcular a Precision
print('Precision : {}'.format(precision_score(y_true, y_pred, average='macro')
# Método para calcular o Recall
print('Recall: {}'.format(recall_score(y_true, y_pred, average='macro')))


print('Matriz de Confusão:')
cm = confusion_matrix(y_true, y_pred)
cm_display = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=['Anorm
cm_display.plot()
plt.savefig('Matriz-resnet-UNITOPATHO')
plt.show()

print ('Accuracy score: ', accuracy_score(y_true, y_pred))
#print('Acuracia obtida com o resnet no Conjunto de Teste UNITOPATHO: {:.2f}'
#    acc))

print(classification_report(y_true, y_pred))
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
print('Acurácia cada classe')
cm.diagonal()
```

```
Found 600 images belonging to 2 classes.
19/19 [==============================] - 216
y predito:
[0 1 1 0 1 1 1 0 1 0 0 1 1 1 1 0 1 1 1 0 0 1
 0 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 1
 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1
 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0
 0 0 0 1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1
 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0
 0 0 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 0 1 1
 0 0 1 1 1 0 0 1 0 0 0 0 0 0 1 1 1 1 1 1 0 1
 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 0 0 0 0 0 0
 0 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 1 1 1 1
 1 1 1 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1
 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1
 1 1 1 0 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 0
 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 0 1 1 1 0 1 0
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 1 1 0 1 0 1 1 1]
y real:
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Alterar o tipo de ambiente de execução