Classificaço de Patologias usando Imagens Médicas

Carregar imagens do diretório

In [110…

```
import os
current_dir = os.path.abspath(os.getcwd())
```

Converter base de dados para treino, validação e teste

In [111…

```
#cria nova pasta para cachorros e gatos atendendo a estrutura do Keras/Tenso
folder = "/novo"
train_folder = current_dir + folder + "/train"
#val_folder = current_dir + folder + "/val"
test_folder = current_dir + folder + "/test"
```

Fazer o Tensorflow carregar as imagens para a RNA

In [112…

```
import tensorflow as tf

print(tf.config.list_physical_devices('GPU'))
print(tf.__version__)
```

```
[]
2.6.1
```

In [113…

```
from tensorflow.keras.utils import image_dataset_from_directory
#image_dataset_from_directory monta uma estrutura de dados com imagens 180x18
# de 32 em 32 imagens
train_dataset = image_dataset_from_directory(train_folder,
                                             image_size=(180, 180),
                                             batch_size=32)

#validation_dataset = image_dataset_from_directory(val_folder,
                                                  #image_size=(180, 180),
                                                  #batch_size=32)

test_dataset = image_dataset_from_directory(test_folder,
                                            image_size=(180, 180),
                                            batch_size=32)
```

```
Found 34931 files belonging to 2 classes.
Found 484 files belonging to 2 classes.
```

In [114…

```
#
for data_batch, labels_batch in train_dataset:
    print("data batch shape:", data_batch.shape)
    print("labels batch shape:", labels_batch.shape)
    print(data_batch[0].shape)
    break
```

```
data batch shape: (32, 180, 180, 3)
labels batch shape: (32,)
(180, 180, 3)
```

Treinando o modelo

In [115…

```
from tensorflow import keras
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.layers.experimental.preprocessing import Rescaling

#cria uma arquitetura de uma rede neural profunda vazia
model = keras.Sequential()
#model.add(Rescaling(scale=1.0/255))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(180,
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(1, activation="sigmoid"))
model.compile(loss="binary_crossentropy",optimizer="adam",metrics=["accuracy"
#model.add(Dense(4, activation='softmax'))
#model.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['ac
```

In [116…

```
from tensorflow.keras.callbacks import ModelCheckpoint

callbacks = [
    ModelCheckpoint(
        filepath="classificacao11.keras",
        save_best_only=True,
        monitor="loss"
    )
]

history = model.fit(
    train_dataset,
    epochs=100,
    #validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/100
1092/1092 [==============================] - 332s 304ms/step - loss: 2.5511 -
accuracy: 0.7446
Epoch 2/100
1092/1092 [==============================] - 326s 297ms/step - loss: 0.5042 -
accuracy: 0.7617
Epoch 3/100
1092/1092 [==============================] - 323s 296ms/step - loss: 0.5062 -
accuracy: 0.7689
Epoch 4/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.5158 -
accuracy: 0.7700
Epoch 5/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.4354 -
accuracy: 0.7951
Epoch 6/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.3374 -
accuracy: 0.8446
Epoch 7/100
1092/1092 [==============================] - 316s 289ms/step - loss: 0.2707 -
accuracy: 0.8857
Epoch 8/100
```

```
1092/1092 [==============================] - 317s 290ms/step - loss: 0.2369 -
accuracy: 0.9034
Epoch 9/100
1092/1092 [==============================] - 316s 290ms/step - loss: 0.1772 -
accuracy: 0.9307
Epoch 10/100
1092/1092 [==============================] - 316s 289ms/step - loss: 0.1387 -
accuracy: 0.9503
Epoch 11/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.1156 -
accuracy: 0.9570
Epoch 12/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.1025 -
accuracy: 0.9643
Epoch 13/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0943 -
accuracy: 0.9683
Epoch 14/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0825 -
accuracy: 0.9725
Epoch 15/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0761 -
accuracy: 0.9750
Epoch 16/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0820 -
accuracy: 0.9751
Epoch 17/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0661 -
accuracy: 0.9810
Epoch 18/100
1092/1092 [==============================] - 312s 285ms/step - loss: 0.0464 -
accuracy: 0.9862
Epoch 19/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0493 -
accuracy: 0.9859
Epoch 20/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0672 -
accuracy: 0.9823
Epoch 21/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0477 -
accuracy: 0.9869
Epoch 22/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0369 -
accuracy: 0.9904
Epoch 23/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0522 -
accuracy: 0.9876
Epoch 24/100
1092/1092 [==============================] - 310s 283ms/step - loss: 0.0427 -
accuracy: 0.9895
Epoch 25/100
1092/1092 [==============================] - 310s 283ms/step - loss: 0.0349 -
accuracy: 0.9922
Epoch 26/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0361 -
accuracy: 0.9924
Epoch 27/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0437 -
accuracy: 0.9910
Epoch 28/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0230 -
accuracy: 0.9942
Epoch 29/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0337 -
```

```
accuracy: 0.9925
Epoch 30/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0352 -
accuracy: 0.9930
Epoch 31/100
1092/1092 [==============================] - 310s 283ms/step - loss: 0.0315 -
accuracy: 0.9941
Epoch 32/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0367 -
accuracy: 0.9937
Epoch 33/100
1092/1092 [==============================] - 309s 283ms/step - loss: 0.0399 -
accuracy: 0.9939
Epoch 34/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0312 -
accuracy: 0.9946
Epoch 35/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0288 -
accuracy: 0.9950
Epoch 36/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0331 -
accuracy: 0.9952
Epoch 37/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0302 -
accuracy: 0.9948
Epoch 38/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0341 -
accuracy: 0.9954
Epoch 39/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0296 -
accuracy: 0.9960
Epoch 40/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0298 -
accuracy: 0.9956
Epoch 41/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0409 -
accuracy: 0.9947
Epoch 42/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0394 -
accuracy: 0.9951
Epoch 43/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0246 -
accuracy: 0.9963
Epoch 44/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0380 -
accuracy: 0.9952
Epoch 45/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0341 -
accuracy: 0.9960
Epoch 46/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0538 -
accuracy: 0.9948
Epoch 47/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0318 -
accuracy: 0.9960
Epoch 48/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0459 -
accuracy: 0.9953
Epoch 49/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0320 -
accuracy: 0.9967
Epoch 50/100
1092/1092 [==============================] - 312s 285ms/step - loss: 0.0378 -
accuracy: 0.9962
```

```
Epoch 51/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0387 -
accuracy: 0.9963
Epoch 52/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0370 -
accuracy: 0.9969
Epoch 53/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0264 -
accuracy: 0.9971
Epoch 54/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0381 -
accuracy: 0.9964
Epoch 55/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0380 -
accuracy: 0.9963
Epoch 56/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0336 -
accuracy: 0.9971
Epoch 57/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0394 -
accuracy: 0.9968
Epoch 58/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0532 -
accuracy: 0.9960
Epoch 59/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0350 -
accuracy: 0.9965
Epoch 60/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0294 -
accuracy: 0.9976
Epoch 61/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0395 -
accuracy: 0.9970
Epoch 62/100
1092/1092 [==============================] - 312s 285ms/step - loss: 0.0298 -
accuracy: 0.9973
Epoch 63/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0291 -
accuracy: 0.9973
Epoch 64/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0472 -
accuracy: 0.9966
Epoch 65/100
1092/1092 [==============================] - 317s 290ms/step - loss: 0.0291 -
accuracy: 0.9972
Epoch 66/100
1092/1092 [==============================] - 317s 290ms/step - loss: 0.0670 -
accuracy: 0.9956
Epoch 67/100
1092/1092 [==============================] - 316s 290ms/step - loss: 0.0418 -
accuracy: 0.9968
Epoch 68/100
1092/1092 [==============================] - 316s 289ms/step - loss: 0.0431 -
accuracy: 0.9971
Epoch 69/100
1092/1092 [==============================] - 315s 289ms/step - loss: 0.0411 -
accuracy: 0.9976
Epoch 70/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0352 -
accuracy: 0.9976
Epoch 71/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0512 -
accuracy: 0.9971
Epoch 72/100
```

```
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0390 -
accuracy: 0.9971
Epoch 73/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0412 -
accuracy: 0.9976
Epoch 74/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0343 -
accuracy: 0.9979
Epoch 75/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0370 -
accuracy: 0.9976
Epoch 76/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0387 -
accuracy: 0.9974
Epoch 77/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0423 -
accuracy: 0.9979
Epoch 78/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0690 -
accuracy: 0.9967
Epoch 79/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0418 -
accuracy: 0.9978
Epoch 80/100
1092/1092 [==============================] - 312s 285ms/step - loss: 0.0489 -
accuracy: 0.9975
Epoch 81/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0525 -
accuracy: 0.9973
Epoch 82/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0541 -
accuracy: 0.9978
Epoch 83/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0270 -
accuracy: 0.9983
Epoch 84/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0372 -
accuracy: 0.9976
Epoch 85/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0572 -
accuracy: 0.9975
Epoch 86/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0766 -
accuracy: 0.9964
Epoch 87/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0421 -
accuracy: 0.9979
Epoch 88/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0541 -
accuracy: 0.9972
Epoch 89/100
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0345 -
accuracy: 0.9983
Epoch 90/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0465 -
accuracy: 0.9977
Epoch 91/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0319 -
accuracy: 0.9982
Epoch 92/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0461 -
accuracy: 0.9974
Epoch 93/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0665 -
```

```
accuracy: 0.9972
Epoch 94/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0642 -
accuracy: 0.9975
Epoch 95/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0546 -
accuracy: 0.9978
Epoch 96/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0604 -
accuracy: 0.9973
Epoch 97/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0375 -
accuracy: 0.9985
Epoch 98/100
1092/1092 [==============================] - 311s 284ms/step - loss: 0.0600 -
accuracy: 0.9974
Epoch 99/100
1092/1092 [==============================] - 310s 284ms/step - loss: 0.0514 -
accuracy: 0.9977
Epoch 100/100
1092/1092 [==============================] - 310s 283ms/step - loss: 0.0502 -
accuracy: 0.9980
```

In [117…

```
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 178, 178, 32)      896
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 89, 89, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 87, 87, 64)        18496
_________________________________________________________________
flatten_9 (Flatten)          (None, 484416)            0
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 484417
=================================================================
Total params: 503,809
Trainable params: 503,809
Non-trainable params: 0
_________________________________________________________________
```

In [118…

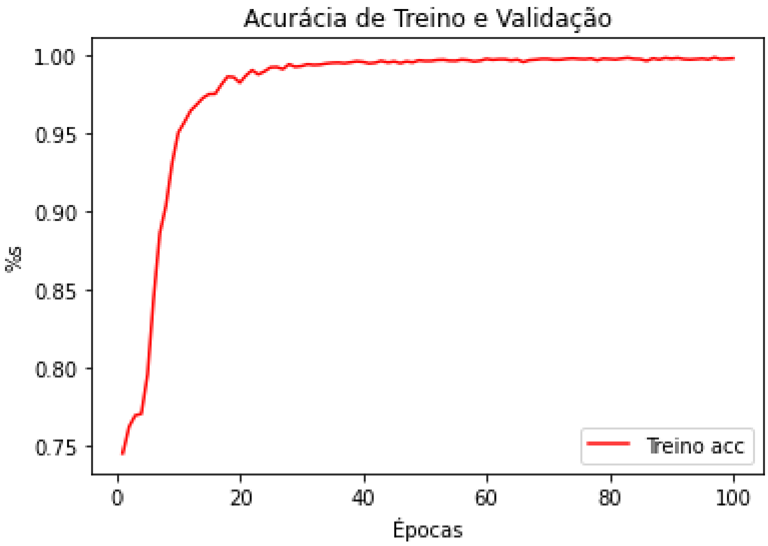
```
#https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=pt-br#alt
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, "conversao_01_11")
```

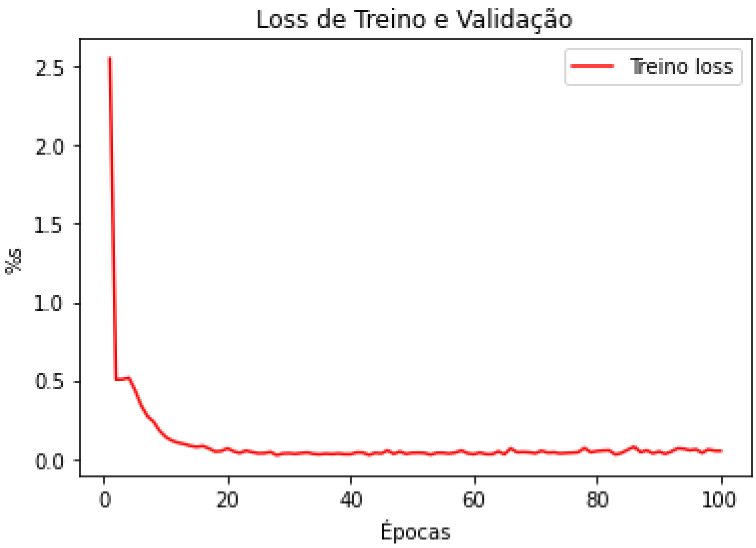
Visualização de Resultados

In [119…

```
import matplotlib.pyplot as plt
accuracy = history.history["accuracy"]
#val_accuracy = history.history["val_accuracy"]
loss = history.history["loss"]
#val_loss = history.history["val_loss"]
epochs = range(1, len(accuracy) + 1)
plt.plot(epochs, accuracy, "r", label="Treino acc")
#plt.plot(epochs, val_accuracy, "b", label="Val acc")
plt.xlabel("Épocas")
plt.ylabel("%s")
```

```
plt.title("Acurácia de Treino e Validação")
plt.legend()
plt.figure()
plt.plot(epochs, loss, "r", label="Treino loss")
#plt.plot(epochs, val_loss, "b", label="Val loss")
plt.xlabel("Épocas")
plt.ylabel("%s")
plt.title("Loss de Treino e Validação")
plt.legend()
plt.show()
```





Resultados do Conjunto de Teste

In [120…

```
#from tensorflow import keras
#model = keras.models.load_model("classificacao01.keras")
# serialize model to JSON
#model_json = model.to_json()
#with open("classificacao01.json", "w") as json_file:json_file.write(model_js
# serialize weights to HDF5
#model.save_weights("classificacao01.h5")
#print("Saved model to disk")
```

In [121…

```
test_loss, test_acc = model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
16/16 [==============================] - 2s 79ms/step - loss: 0.7895 - accura
cy: 0.9835
Test accuracy: 0.983
```

In [ ]:

In [ ]:

In [ ]:

Referências

- https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/
- https://stackoverflow.com/questions/3430372/how-do-i-get-the-full-path-of-the-current-files-directory
- https://www.geeksforgeeks.org/python-list-files-in-a-directory/
- https://pynative.com/python-random-sample/
- https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-to-classify-photos-of-dogs-and-cats/
- https://www.mygreatlearning.com/blog/keras-tutorial/
- https://www.machinecurve.com/index.php/2020/03/30/how-to-use-conv2d-with-keras/
- https://www.pyimagesearch.com/2021/06/30/how-to-use-the-modelcheckpoint-callback-with-keras-and-tensorflow/