Classificaço de Patologias usando Imagens Médicas

Carregar imagens do diretório

In [122…

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

Converter base de dados para treino, validação e teste

In [123…

```
#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 [124…

```
import tensorflow as tf

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

```
[]
2.6.1
```

In [125…

```
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 [126…

```
#
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 [127…

```
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 [128…

```
from tensorflow.keras.callbacks import ModelCheckpoint

callbacks = [
    ModelCheckpoint(
        filepath="classificacao12.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 [==============================] - 327s 299ms/step - loss: 3.1812 -
accuracy: 0.7450
Epoch 2/100
1092/1092 [==============================] - 322s 295ms/step - loss: 0.5551 -
accuracy: 0.7538
Epoch 3/100
1092/1092 [==============================] - 323s 295ms/step - loss: 0.5384 -
accuracy: 0.7580
Epoch 4/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.5180 -
accuracy: 0.7629
Epoch 5/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.5031 -
accuracy: 0.7718
Epoch 6/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.4727 -
accuracy: 0.7868
Epoch 7/100
1092/1092 [==============================] - 316s 289ms/step - loss: 0.4633 -
accuracy: 0.7955
Epoch 8/100
```

```
1092/1092 [==============================] - 316s 290ms/step - loss: 0.4391 -
accuracy: 0.8048
Epoch 9/100
1092/1092 [==============================] - 317s 290ms/step - loss: 0.4030 -
accuracy: 0.8219
Epoch 10/100
1092/1092 [==============================] - 316s 289ms/step - loss: 0.3686 -
accuracy: 0.8393
Epoch 11/100
1092/1092 [==============================] - 315s 288ms/step - loss: 0.3478 -
accuracy: 0.8490
Epoch 12/100
1092/1092 [==============================] - 315s 288ms/step - loss: 0.3196 -
accuracy: 0.8642
Epoch 13/100
1092/1092 [==============================] - 315s 288ms/step - loss: 0.3057 -
accuracy: 0.8707
Epoch 14/100
1092/1092 [==============================] - 315s 288ms/step - loss: 0.2560 -
accuracy: 0.8941
Epoch 15/100
1092/1092 [==============================] - 314s 288ms/step - loss: 0.2263 -
accuracy: 0.9080
Epoch 16/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.2041 -
accuracy: 0.9168
Epoch 17/100
1092/1092 [==============================] - 314s 288ms/step - loss: 0.1905 -
accuracy: 0.9244
Epoch 18/100
1092/1092 [==============================] - 314s 288ms/step - loss: 0.1798 -
accuracy: 0.9308
Epoch 19/100
1092/1092 [==============================] - 314s 288ms/step - loss: 0.1576 -
accuracy: 0.9399
Epoch 20/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.1287 -
accuracy: 0.9520
Epoch 21/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.1156 -
accuracy: 0.9576
Epoch 22/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.1207 -
accuracy: 0.9572
Epoch 23/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.1053 -
accuracy: 0.9630
Epoch 24/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0889 -
accuracy: 0.9680
Epoch 25/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0980 -
accuracy: 0.9656
Epoch 26/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0782 -
accuracy: 0.9749
Epoch 27/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0746 -
accuracy: 0.9756
Epoch 28/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0654 -
accuracy: 0.9794
Epoch 29/100
1092/1092 [==============================] - 314s 288ms/step - loss: 0.0640 -
```

```
accuracy: 0.9794
Epoch 30/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0621 -
accuracy: 0.9812
Epoch 31/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0620 -
accuracy: 0.9816
Epoch 32/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0597 -
accuracy: 0.9825
Epoch 33/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0664 -
accuracy: 0.9814
Epoch 34/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0639 -
accuracy: 0.9828
Epoch 35/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0474 -
accuracy: 0.9866
Epoch 36/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0515 -
accuracy: 0.9855
Epoch 37/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0583 -
accuracy: 0.9852
Epoch 38/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0618 -
accuracy: 0.9838
Epoch 39/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0473 -
accuracy: 0.9873
Epoch 40/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0429 -
accuracy: 0.9897
Epoch 41/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0514 -
accuracy: 0.9874
Epoch 42/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0426 -
accuracy: 0.9890
Epoch 43/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0527 -
accuracy: 0.9871
Epoch 44/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0506 -
accuracy: 0.9877
Epoch 45/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0486 -
accuracy: 0.9883
Epoch 46/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0490 -
accuracy: 0.9886
Epoch 47/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0638 -
accuracy: 0.9868
Epoch 48/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0519 -
accuracy: 0.9889
Epoch 49/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0572 -
accuracy: 0.9883
Epoch 50/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0497 -
accuracy: 0.9899
```

```
Epoch 51/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0425 -
accuracy: 0.9902
Epoch 52/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0425 -
accuracy: 0.9909
Epoch 53/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0898 -
accuracy: 0.9843
Epoch 54/100
1092/1092 [==============================] - 314s 288ms/step - loss: 0.0748 -
accuracy: 0.9866
Epoch 55/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0525 -
accuracy: 0.9899
Epoch 56/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0662 -
accuracy: 0.9890
Epoch 57/100
1092/1092 [==============================] - 314s 288ms/step - loss: 0.0438 -
accuracy: 0.9915
Epoch 58/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0528 -
accuracy: 0.9904
Epoch 59/100
1092/1092 [==============================] - 314s 288ms/step - loss: 0.0469 -
accuracy: 0.9905
Epoch 60/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0650 -
accuracy: 0.9887
Epoch 61/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0483 -
accuracy: 0.9908
Epoch 62/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0508 -
accuracy: 0.9911
Epoch 63/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0463 -
accuracy: 0.9914
Epoch 64/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0553 -
accuracy: 0.9918
Epoch 65/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0429 -
accuracy: 0.9918
Epoch 66/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0578 -
accuracy: 0.9902
Epoch 67/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0575 -
accuracy: 0.9916
Epoch 68/100
1092/1092 [==============================] - 314s 288ms/step - loss: 0.0547 -
accuracy: 0.9915
Epoch 69/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0468 -
accuracy: 0.9919
Epoch 70/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0514 -
accuracy: 0.9910
Epoch 71/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0445 -
accuracy: 0.9926
Epoch 72/100
```

```
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0422 -
accuracy: 0.9934
Epoch 73/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0394 -
accuracy: 0.9936
Epoch 74/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0502 -
accuracy: 0.9921
Epoch 75/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0782 -
accuracy: 0.9891
Epoch 76/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0838 -
accuracy: 0.9904
Epoch 77/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0531 -
accuracy: 0.9930
Epoch 78/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0581 -
accuracy: 0.9923
Epoch 79/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0464 -
accuracy: 0.9930
Epoch 80/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0461 -
accuracy: 0.9937
Epoch 81/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0605 -
accuracy: 0.9928
Epoch 82/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0479 -
accuracy: 0.9930
Epoch 83/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0625 -
accuracy: 0.9929
Epoch 84/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0614 -
accuracy: 0.9927
Epoch 85/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0597 -
accuracy: 0.9918
Epoch 86/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0662 -
accuracy: 0.9916
Epoch 87/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0392 -
accuracy: 0.9941
Epoch 88/100
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0762 -
accuracy: 0.9916
Epoch 89/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0631 -
accuracy: 0.9926
Epoch 90/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0702 -
accuracy: 0.9918
Epoch 91/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0489 -
accuracy: 0.9943
Epoch 92/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0506 -
accuracy: 0.9940
Epoch 93/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0707 -
```

```
accuracy: 0.9920
Epoch 94/100
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0572 -
accuracy: 0.9933
Epoch 95/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0809 -
accuracy: 0.9916
Epoch 96/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0597 -
accuracy: 0.9938
Epoch 97/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0624 -
accuracy: 0.9933
Epoch 98/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0618 -
accuracy: 0.9938
Epoch 99/100
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0569 -
accuracy: 0.9940
Epoch 100/100
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0762 -
accuracy: 0.9933
```

In [129…

```
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_20 (Conv2D)           (None, 178, 178, 32)      896
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 89, 89, 32)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 87, 87, 64)        18496
_________________________________________________________________
flatten_10 (Flatten)         (None, 484416)            0
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 484417
=================================================================
Total params: 503,809
Trainable params: 503,809
Non-trainable params: 0
_________________________________________________________________
```

In [130…

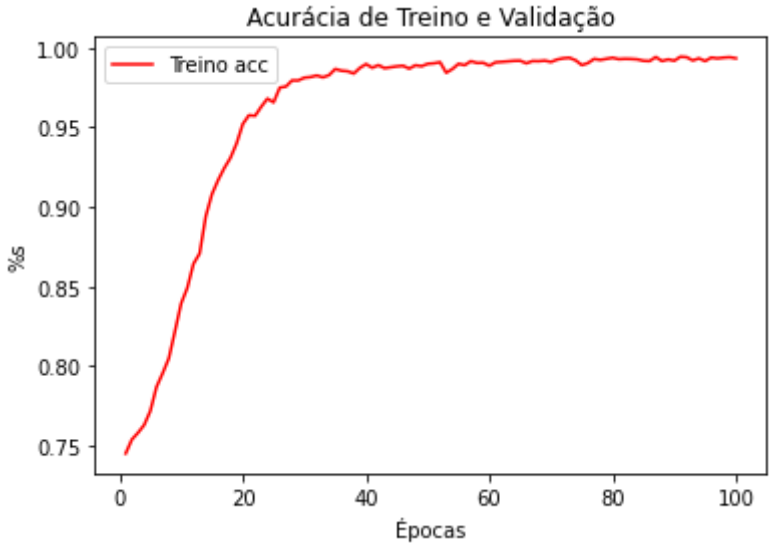
```
#https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=pt-br#al
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, "conversao_01_12")
```

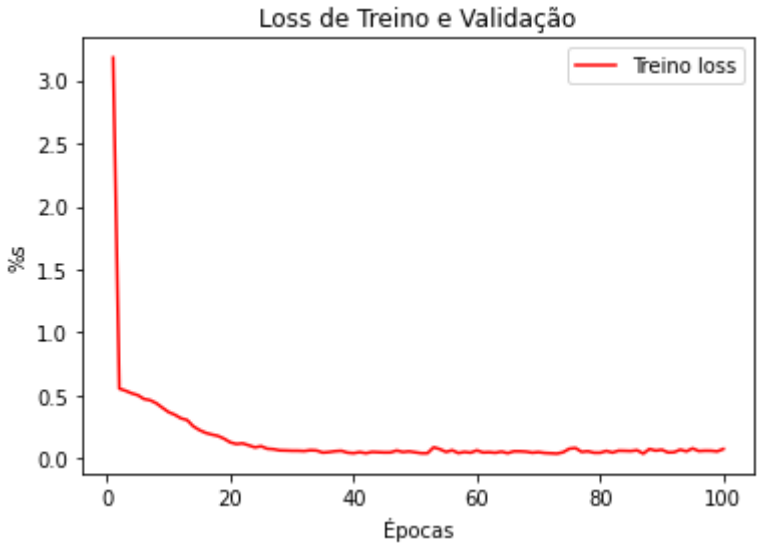
Visualização de Resultados

In [131…

```
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 [132…

```
#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 [133…

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

```
16/16 [==============================] - 3s 79ms/step - loss: 4.6813 - accura
cy: 0.8698
Test accuracy: 0.870
```

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/