Classificaço de Patologias usando Imagens Médicas

Carregar imagens do diretório

In [1]:
```
import os
current_dir = os.path.abspath(os.getcwd())
```

Converter base de dados para treino, validação e teste

In [2]:
```
#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 [3]:
```
import tensorflow as tf

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

```
2022-06-27 08:37:48.066891: W tensorflow/stream_executor/platform/default/dso
_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: l
ibcudart.so.11.0: cannot open shared object file: No such file or directory
2022-06-27 08:37:48.066946: I tensorflow/stream_executor/cuda/cudart_stub.cc:
29] Ignore above cudart dlerror if you do not have a GPU set up on your machi
ne.
[]
2.6.1
```

```
2022-06-27 08:37:58.616591: W tensorflow/stream_executor/platform/default/dso
_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcud
a.so.1: cannot open shared object file: No such file or directory
2022-06-27 08:37:58.616665: W tensorflow/stream_executor/cuda/cuda_driver.cc:
269] failed call to cuInit: UNKNOWN ERROR (303)
2022-06-27 08:37:58.616712: I tensorflow/stream_executor/cuda/cuda_diagnostic
s.cc:156] kernel driver does not appear to be running on this host (pc): /pro
c/driver/nvidia/version does not exist
```

In [4]:
```
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.
```

```
2022-06-27 08:38:00.415840: I tensorflow/core/platform/cpu_feature_guard.cc:1
42] This TensorFlow binary is optimized with oneAPI Deep Neural Network Libra
ry (oneDNN) to use the following CPU instructions in performance-critical ope
rations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate c
ompiler flags.
```

In [5]:

```
#
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
```

```
2022-06-27 08:38:00.498187: I tensorflow/compiler/mlir/mlir_graph_optimizatio
n_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered
2)
data batch shape: (32, 180, 180, 3)
labels batch shape: (32,)
(180, 180, 3)
```

Treinando o modelo

In [6]:

```
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 [7]:

```
from tensorflow.keras.callbacks import ModelCheckpoint

callbacks = [
    ModelCheckpoint(
        filepath="classificacao15.keras",
        save_best_only=True,
        monitor="loss"
    )
]

history = model.fit(
    train_dataset,
    epochs=50,
```

```
    #validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/50
1092/1092 [==============================] - 360s 329ms/step - loss: 2.8272 -
accuracy: 0.7476
Epoch 2/50
1092/1092 [==============================] - 345s 316ms/step - loss: 0.4046 -
accuracy: 0.7978
Epoch 3/50
1092/1092 [==============================] - 347s 318ms/step - loss: 0.3606 -
accuracy: 0.8415
Epoch 4/50
1092/1092 [==============================] - 343s 314ms/step - loss: 0.3254 -
accuracy: 0.8657
Epoch 5/50
1092/1092 [==============================] - 339s 310ms/step - loss: 0.2764 -
accuracy: 0.8892
Epoch 6/50
1092/1092 [==============================] - 349s 319ms/step - loss: 0.2427 -
accuracy: 0.9076
Epoch 7/50
1092/1092 [==============================] - 350s 321ms/step - loss: 0.3993 -
accuracy: 0.8155
Epoch 8/50
1092/1092 [==============================] - 353s 323ms/step - loss: 0.2088 -
accuracy: 0.9207
Epoch 9/50
1092/1092 [==============================] - 350s 320ms/step - loss: 0.1511 -
accuracy: 0.9451
Epoch 10/50
1092/1092 [==============================] - 335s 307ms/step - loss: 0.1189 -
accuracy: 0.9585
Epoch 11/50
1092/1092 [==============================] - 338s 309ms/step - loss: 0.0864 -
accuracy: 0.9698
Epoch 12/50
1092/1092 [==============================] - 338s 310ms/step - loss: 0.0794 -
accuracy: 0.9729
Epoch 13/50
1092/1092 [==============================] - 344s 315ms/step - loss: 0.0637 -
accuracy: 0.9795
Epoch 14/50
1092/1092 [==============================] - 337s 309ms/step - loss: 0.0581 -
accuracy: 0.9820
Epoch 15/50
1092/1092 [==============================] - 356s 326ms/step - loss: 0.0635 -
accuracy: 0.9809
Epoch 16/50
1092/1092 [==============================] - 380s 348ms/step - loss: 0.0616 -
accuracy: 0.9826
Epoch 17/50
1092/1092 [==============================] - 369s 338ms/step - loss: 0.0487 -
accuracy: 0.9853
Epoch 18/50
1092/1092 [==============================] - 374s 338ms/step - loss: 0.0508 -
accuracy: 0.9864
Epoch 19/50
1092/1092 [==============================] - 374s 341ms/step - loss: 0.0494 -
accuracy: 0.9881
Epoch 20/50
1092/1092 [==============================] - 352s 321ms/step - loss: 0.0461 -
accuracy: 0.9886
Epoch 21/50
```

```
1092/1092 [==============================] - 382s 349ms/step - loss: 0.0470 -
accuracy: 0.9878
Epoch 22/50
1092/1092 [==============================] - 374s 341ms/step - loss: 0.0346 -
accuracy: 0.9908
Epoch 23/50
1092/1092 [==============================] - 355s 323ms/step - loss: 0.0464 -
accuracy: 0.9894
Epoch 24/50
1092/1092 [==============================] - 354s 323ms/step - loss: 0.0374 -
accuracy: 0.9911
Epoch 25/50
1092/1092 [==============================] - 344s 314ms/step - loss: 0.0519 -
accuracy: 0.9896
Epoch 26/50
1092/1092 [==============================] - 359s 327ms/step - loss: 0.0355 -
accuracy: 0.9927
Epoch 27/50
1092/1092 [==============================] - 342s 313ms/step - loss: 0.0338 -
accuracy: 0.9926
Epoch 28/50
1092/1092 [==============================] - 348s 319ms/step - loss: 0.0444 -
accuracy: 0.9917
Epoch 29/50
1092/1092 [==============================] - 344s 315ms/step - loss: 0.0292 -
accuracy: 0.9943
Epoch 30/50
1092/1092 [==============================] - 352s 322ms/step - loss: 0.0451 -
accuracy: 0.9916
Epoch 31/50
1092/1092 [==============================] - 363s 333ms/step - loss: 0.0439 -
accuracy: 0.9929
Epoch 32/50
1092/1092 [==============================] - 351s 322ms/step - loss: 0.0339 -
accuracy: 0.9939
Epoch 33/50
1092/1092 [==============================] - 349s 319ms/step - loss: 0.0525 -
accuracy: 0.9914
Epoch 34/50
1092/1092 [==============================] - 348s 319ms/step - loss: 0.0350 -
accuracy: 0.9941
Epoch 35/50
1092/1092 [==============================] - 347s 317ms/step - loss: 0.0441 -
accuracy: 0.9929
Epoch 36/50
1092/1092 [==============================] - 343s 314ms/step - loss: 0.0428 -
accuracy: 0.9940
Epoch 37/50
1092/1092 [==============================] - 343s 314ms/step - loss: 0.0377 -
accuracy: 0.9946
Epoch 38/50
1092/1092 [==============================] - 324s 296ms/step - loss: 0.0411 -
accuracy: 0.9939
Epoch 39/50
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0332 -
accuracy: 0.9954
Epoch 40/50
1092/1092 [==============================] - 334s 306ms/step - loss: 0.0477 -
accuracy: 0.9930
Epoch 41/50
1092/1092 [==============================] - 339s 311ms/step - loss: 0.0413 -
accuracy: 0.9947
Epoch 42/50
1092/1092 [==============================] - 332s 304ms/step - loss: 0.0310 -
```

```
accuracy: 0.9957
Epoch 43/50
1092/1092 [==============================] - 345s 316ms/step - loss: 0.0495 -
accuracy: 0.9948
Epoch 44/50
1092/1092 [==============================] - 370s 338ms/step - loss: 0.0523 -
accuracy: 0.9945
Epoch 45/50
1092/1092 [==============================] - 371s 340ms/step - loss: 0.0465 -
accuracy: 0.9943
Epoch 46/50
1092/1092 [==============================] - 326s 299ms/step - loss: 0.0438 -
accuracy: 0.9954
Epoch 47/50
1092/1092 [==============================] - 336s 308ms/step - loss: 0.0430 -
accuracy: 0.9957
Epoch 48/50
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0535 -
accuracy: 0.9946
Epoch 49/50
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0386 -
accuracy: 0.9960
Epoch 50/50
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0432 -
accuracy: 0.9955
```

In [8]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 178, 178, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 89, 89, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 87, 87, 64)        18496
_________________________________________________________________
flatten (Flatten)            (None, 484416)            0
_________________________________________________________________
dense (Dense)                (None, 1)                 484417
=================================================================
Total params: 503,809
Trainable params: 503,809
Non-trainable params: 0
_________________________________________________________________
```

In [9]:

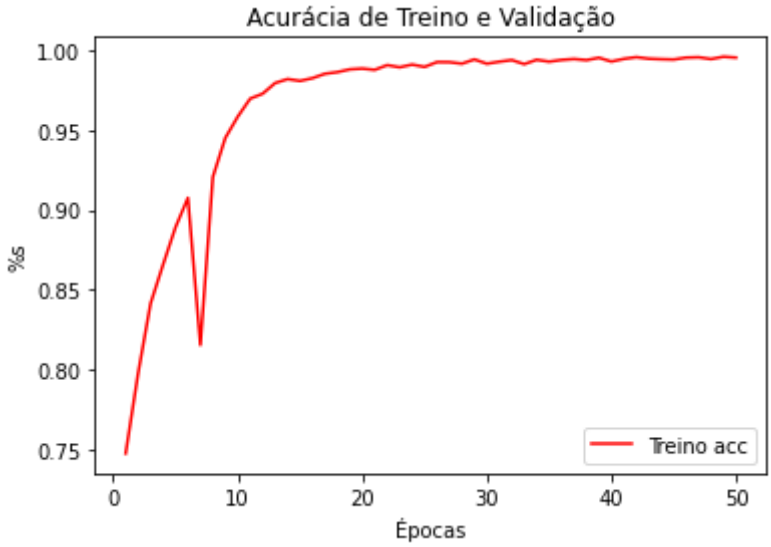
```
#https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=pt-br#al
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, "conversao_01_15")
```

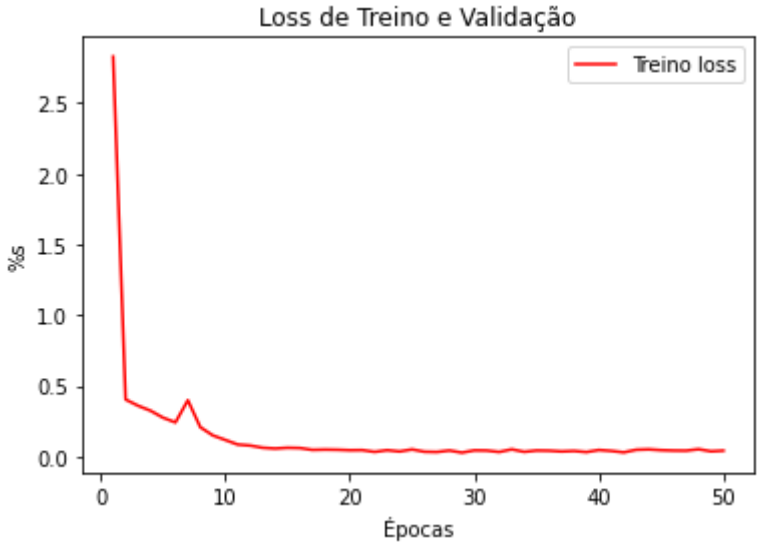
Visualização de Resultados

In [10]:

```
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 [11]:

```
#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 [12]:
```
test_loss, test_acc = model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
16/16 [==============================] - 3s 62ms/step - loss: 0.2670 - accura
cy: 0.9855
Test accuracy: 0.986
```

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/