

# Classificação de Patologias usando Imagens Médicas

## Carregar imagens do diretório

```
In [41]: import os
current_dir = os.path.abspath(os.getcwd())
```

## Converter base de dados para treino, validação e teste

```
In [42]: #cria nova pasta para cachorros e gatos atendendo a estrutura do Keras/TensorFlow
folder = "/oct_2017"
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 [43]: import tensorflow as tf

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

```
[]
2.6.1
```

```
In [44]: from tensorflow.keras.utils import image_dataset_from_directory
#image_dataset_from_directory monta uma estrutura de dados com imagens 180x180
# 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 [45]: #
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 [46]:

```
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 [47]:

```
from tensorflow.keras.callbacks import ModelCheckpoint

callbacks = [
    ModelCheckpoint(
        filepath="classificacao08.keras",
        save_best_only=True,
        monitor="loss"
    )
]

history = model.fit(
    train_dataset,
    epochs=30,
    #validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/30
1092/1092 [=====] - 336s 306ms/step - loss: 2.7089 -
accuracy: 0.7752
Epoch 2/30
1092/1092 [=====] - 322s 295ms/step - loss: 0.3830 -
accuracy: 0.8258
Epoch 3/30
1092/1092 [=====] - 322s 295ms/step - loss: 0.3465 -
accuracy: 0.8523
Epoch 4/30
1092/1092 [=====] - 322s 295ms/step - loss: 0.3222 -
accuracy: 0.8707
Epoch 5/30
1092/1092 [=====] - 322s 295ms/step - loss: 0.2836 -
accuracy: 0.8879
Epoch 6/30
1092/1092 [=====] - 321s 294ms/step - loss: 0.2328 -
accuracy: 0.9109
Epoch 7/30
1092/1092 [=====] - 315s 289ms/step - loss: 0.2113 -
accuracy: 0.9204
Epoch 8/30
```

```
1092/1092 [=====] - 315s 289ms/step - loss: 0.1815 -  
accuracy: 0.9323  
Epoch 9/30  
1092/1092 [=====] - 315s 288ms/step - loss: 0.1495 -  
accuracy: 0.9455  
Epoch 10/30  
1092/1092 [=====] - 315s 288ms/step - loss: 0.1379 -  
accuracy: 0.9509  
Epoch 11/30  
1092/1092 [=====] - 314s 287ms/step - loss: 0.1177 -  
accuracy: 0.9585  
Epoch 12/30  
1092/1092 [=====] - 314s 287ms/step - loss: 0.0921 -  
accuracy: 0.9693  
Epoch 13/30  
1092/1092 [=====] - 313s 286ms/step - loss: 0.0965 -  
accuracy: 0.9685  
Epoch 14/30  
1092/1092 [=====] - 312s 286ms/step - loss: 0.0822 -  
accuracy: 0.9730  
Epoch 15/30  
1092/1092 [=====] - 313s 286ms/step - loss: 0.0696 -  
accuracy: 0.9784  
Epoch 16/30  
1092/1092 [=====] - 313s 286ms/step - loss: 0.0868 -  
accuracy: 0.9731  
Epoch 17/30  
1092/1092 [=====] - 313s 286ms/step - loss: 0.0788 -  
accuracy: 0.9769  
Epoch 18/30  
1092/1092 [=====] - 313s 286ms/step - loss: 0.0637 -  
accuracy: 0.9811  
Epoch 19/30  
1092/1092 [=====] - 314s 287ms/step - loss: 0.0606 -  
accuracy: 0.9827  
Epoch 20/30  
1092/1092 [=====] - 313s 287ms/step - loss: 0.0739 -  
accuracy: 0.9806  
Epoch 21/30  
1092/1092 [=====] - 313s 286ms/step - loss: 0.0628 -  
accuracy: 0.9832  
Epoch 22/30  
1092/1092 [=====] - 317s 291ms/step - loss: 0.0649 -  
accuracy: 0.9836  
Epoch 23/30  
1092/1092 [=====] - 318s 291ms/step - loss: 0.0478 -  
accuracy: 0.9877  
Epoch 24/30  
1092/1092 [=====] - 317s 291ms/step - loss: 0.0696 -  
accuracy: 0.9842  
Epoch 25/30  
1092/1092 [=====] - 317s 290ms/step - loss: 0.0654 -  
accuracy: 0.9852  
Epoch 26/30  
1092/1092 [=====] - 318s 291ms/step - loss: 0.0817 -  
accuracy: 0.9843  
Epoch 27/30  
1092/1092 [=====] - 315s 289ms/step - loss: 0.0534 -  
accuracy: 0.9879  
Epoch 28/30  
1092/1092 [=====] - 313s 286ms/step - loss: 0.0528 -  
accuracy: 0.9889  
Epoch 29/30  
1092/1092 [=====] - 313s 286ms/step - loss: 0.0681 -
```

```
accuracy: 0.9865
Epoch 30/30
1092/1092 [=====] - 313s 286ms/step - loss: 0.0553 -
accuracy: 0.9883
```

In [48]:

```
model.summary()
```

Model: "sequential\_2"

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 89, 89, 32)	0
conv2d_5 (Conv2D)	(None, 87, 87, 64)	18496
flatten_2 (Flatten)	(None, 484416)	0
dense_2 (Dense)	(None, 1)	484417
Total params: 503,809		
Trainable params: 503,809		
Non-trainable params: 0		

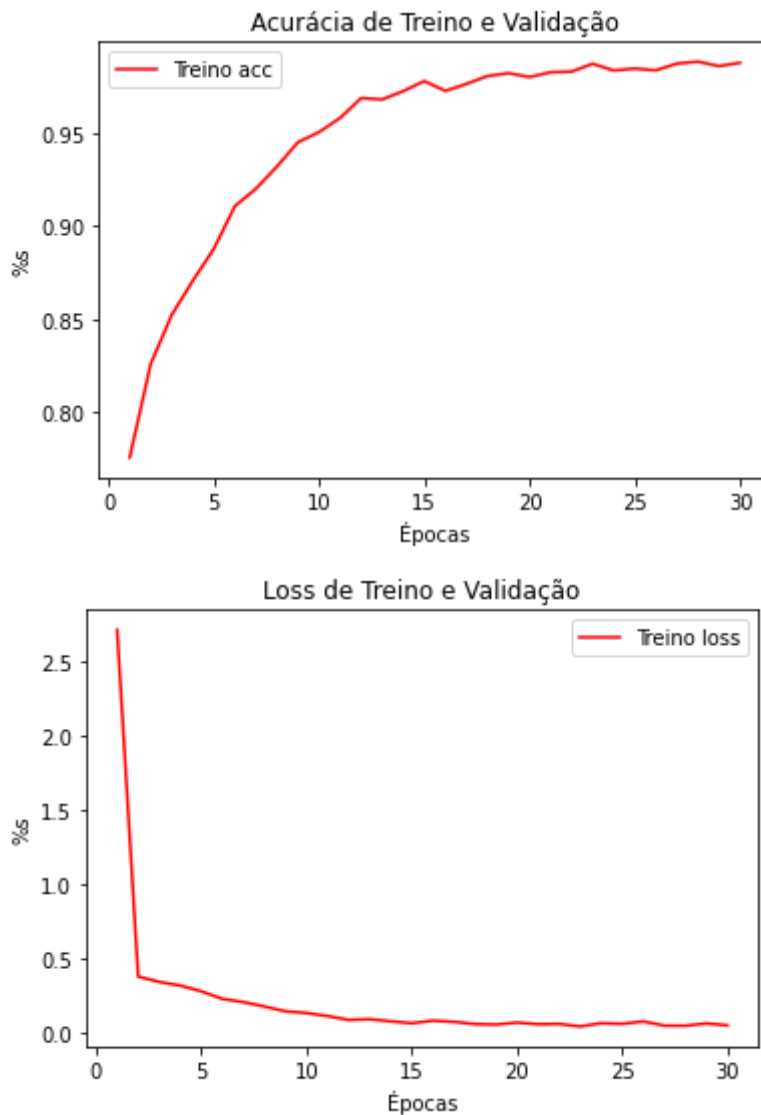
In [49]:

```
#https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=pt-br#ali
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, "conversao_01_08")
```

## Visualização de Resultados

In [50]:

```
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 [51]: #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_json)
# serialize weights to HDF5
#model.save_weights("classificacao01.h5")
#print("Saved model to disk")
```

```
In [52]: test_loss, test_acc = model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

16/16 [=====] - 1s 62ms/step - loss: 0.7747 - accuracy: 0.9277  
Test accuracy: 0.928

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/>