Classificaço de Patologias usando Imagens Médicas

Carregar imagens do diretório

In [26]:
```
import os
current_dir = os.path.abspath(os.getcwd())
```

Converter base de dados para treino, validação e teste

In [27]:
```
#cria nova pasta para cachorros e gatos atendendo a estrutura do Keras/Tenso
folder = "/dataset"
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 [28]:
```
import tensorflow as tf

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

```
[]
2.6.1
```

In [29]:
```
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 [30]:
```
#
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 [31]:

```
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 [32]:

```
from tensorflow.keras.callbacks import ModelCheckpoint

callbacks = [
    ModelCheckpoint(
        filepath="classificacao07.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 307ms/step - loss: 3.3310 -
accuracy: 0.7479
Epoch 2/30
1092/1092 [==============================] - 331s 303ms/step - loss: 0.5550 -
accuracy: 0.7541
Epoch 3/30
1092/1092 [==============================] - 418s 382ms/step - loss: 0.5370 -
accuracy: 0.7601
Epoch 4/30
1092/1092 [==============================] - 467s 426ms/step - loss: 0.5175 -
accuracy: 0.7678
Epoch 5/30
1092/1092 [==============================] - 391s 355ms/step - loss: 0.4993 -
accuracy: 0.7757
Epoch 6/30
1092/1092 [==============================] - 327s 298ms/step - loss: 0.4998 -
accuracy: 0.7783
Epoch 7/30
1092/1092 [==============================] - 410s 376ms/step - loss: 0.4809 -
accuracy: 0.7859
Epoch 8/30
```

```
1092/1092 [==============================] - 410s 376ms/step - loss: 0.4846 -
accuracy: 0.7853
Epoch 9/30
1092/1092 [==============================] - 357s 326ms/step - loss: 0.4471 -
accuracy: 0.8026
Epoch 10/30
1092/1092 [==============================] - 338s 309ms/step - loss: 0.4288 -
accuracy: 0.8115
Epoch 11/30
1092/1092 [==============================] - 337s 309ms/step - loss: 0.4057 -
accuracy: 0.8234
Epoch 12/30
1092/1092 [==============================] - 339s 310ms/step - loss: 0.3936 -
accuracy: 0.8298
Epoch 13/30
1092/1092 [==============================] - 344s 315ms/step - loss: 0.3696 -
accuracy: 0.8401
Epoch 14/30
1092/1092 [==============================] - 365s 335ms/step - loss: 0.3336 -
accuracy: 0.8555
Epoch 15/30
1092/1092 [==============================] - 410s 375ms/step - loss: 0.3061 -
accuracy: 0.8693
Epoch 16/30
1092/1092 [==============================] - 375s 343ms/step - loss: 0.2952 -
accuracy: 0.8738
Epoch 17/30
1092/1092 [==============================] - 323s 296ms/step - loss: 0.2626 -
accuracy: 0.8889
Epoch 18/30
1092/1092 [==============================] - 320s 293ms/step - loss: 0.2496 -
accuracy: 0.8959
Epoch 19/30
1092/1092 [==============================] - 319s 292ms/step - loss: 0.2504 -
accuracy: 0.8973
Epoch 20/30
1092/1092 [==============================] - 328s 300ms/step - loss: 0.2377 -
accuracy: 0.9042
Epoch 21/30
1092/1092 [==============================] - 325s 297ms/step - loss: 0.2352 -
accuracy: 0.9057
Epoch 22/30
1092/1092 [==============================] - 318s 291ms/step - loss: 0.2063 -
accuracy: 0.9156
Epoch 23/30
1092/1092 [==============================] - 318s 292ms/step - loss: 0.1887 -
accuracy: 0.9247
Epoch 24/30
1092/1092 [==============================] - 320s 293ms/step - loss: 0.1701 -
accuracy: 0.9318
Epoch 25/30
1092/1092 [==============================] - 318s 291ms/step - loss: 0.1669 -
accuracy: 0.9341
Epoch 26/30
1092/1092 [==============================] - 316s 289ms/step - loss: 0.1623 -
accuracy: 0.9372
Epoch 27/30
1092/1092 [==============================] - 318s 291ms/step - loss: 0.1590 -
accuracy: 0.9389
Epoch 28/30
1092/1092 [==============================] - 352s 322ms/step - loss: 0.1458 -
accuracy: 0.9447
Epoch 29/30
1092/1092 [==============================] - 381s 349ms/step - loss: 0.1571 -
```

```
accuracy: 0.9419
Epoch 30/30
1092/1092 [==============================] - 388s 355ms/step - loss: 0.1414 -
accuracy: 0.9466
```

In [33]:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_2 (Conv2D)            (None, 178, 178, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 89, 89, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 87, 87, 64)        18496
_________________________________________________________________
flatten_1 (Flatten)          (None, 484416)            0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 484417
=================================================================
Total params: 503,809
Trainable params: 503,809
Non-trainable params: 0
_________________________________________________________________
```

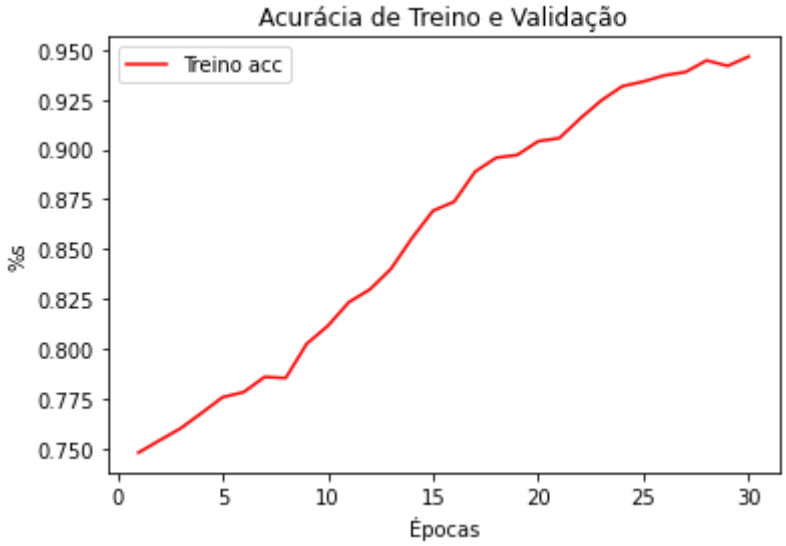
In [34]:

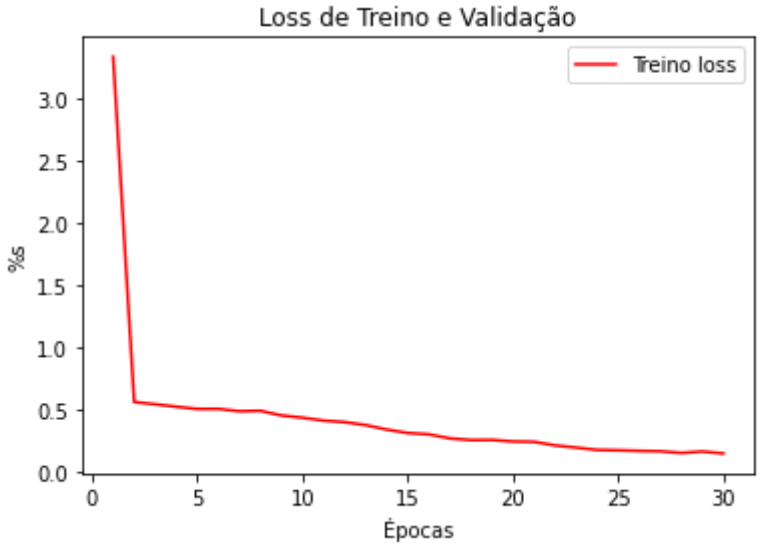
```
#https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=pt-br#ali
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, "conversao_01_07")
```

Visualização de Resultados

In [35]:

```
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 [36]:
```
#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 [37]:
```
test_loss, test_acc = model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
16/16 [==============================] - 3s 64ms/step - loss: 1.4651 - accura
cy: 0.5702
Test accuracy: 0.570
```

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/