Classificaço de Patologias usando Imagens Médicas

Carregar imagens do diretório

In [65]:
```
import os
current_dir = os.path.abspath(os.getcwd())
```

Converter base de dados para treino, validação e teste

In [66]:
```
#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 [67]:
```
import tensorflow as tf

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

```
[]
2.6.1
```

In [68]:
```
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 [69]:
```
#
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 [70]:

```
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 [71]:

```
from tensorflow.keras.callbacks import ModelCheckpoint

callbacks = [
    ModelCheckpoint(
        filepath="classificacao09.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 [==============================] - 325s 297ms/step - loss: 4.7345 -
accuracy: 0.7389
Epoch 2/30
1092/1092 [==============================] - 322s 294ms/step - loss: 0.5582 -
accuracy: 0.7547
Epoch 3/30
1092/1092 [==============================] - 322s 294ms/step - loss: 0.5375 -
accuracy: 0.7596
Epoch 4/30
1092/1092 [==============================] - 322s 295ms/step - loss: 0.5190 -
accuracy: 0.7679
Epoch 5/30
1092/1092 [==============================] - 321s 294ms/step - loss: 0.5050 -
accuracy: 0.7748
Epoch 6/30
1092/1092 [==============================] - 322s 295ms/step - loss: 0.4891 -
accuracy: 0.7808
Epoch 7/30
1092/1092 [==============================] - 321s 294ms/step - loss: 0.4701 -
accuracy: 0.7910
Epoch 8/30
```

```
1092/1092 [==============================] - 325s 298ms/step - loss: 0.4508 -
accuracy: 0.8004
Epoch 9/30
1092/1092 [==============================] - 353s 323ms/step - loss: 0.4165 -
accuracy: 0.8173
Epoch 10/30
1092/1092 [==============================] - 342s 313ms/step - loss: 0.3811 -
accuracy: 0.8343
Epoch 11/30
1092/1092 [==============================] - 338s 308ms/step - loss: 0.3460 -
accuracy: 0.8530
Epoch 12/30
1092/1092 [==============================] - 334s 305ms/step - loss: 0.3192 -
accuracy: 0.8673
Epoch 13/30
1092/1092 [==============================] - 319s 292ms/step - loss: 0.2913 -
accuracy: 0.8797
Epoch 14/30
1092/1092 [==============================] - 316s 290ms/step - loss: 0.2560 -
accuracy: 0.8941
Epoch 15/30
1092/1092 [==============================] - 317s 290ms/step - loss: 0.2388 -
accuracy: 0.9026
Epoch 16/30
1092/1092 [==============================] - 317s 290ms/step - loss: 0.2121 -
accuracy: 0.9154
Epoch 17/30
1092/1092 [==============================] - 317s 290ms/step - loss: 0.2012 -
accuracy: 0.9223
Epoch 18/30
1092/1092 [==============================] - 317s 290ms/step - loss: 0.1723 -
accuracy: 0.9321
Epoch 19/30
1092/1092 [==============================] - 317s 290ms/step - loss: 0.1580 -
accuracy: 0.9409
Epoch 20/30
1092/1092 [==============================] - 316s 289ms/step - loss: 0.1466 -
accuracy: 0.9453
Epoch 21/30
1092/1092 [==============================] - 313s 287ms/step - loss: 0.1366 -
accuracy: 0.9513
Epoch 22/30
1092/1092 [==============================] - 313s 287ms/step - loss: 0.1146 -
accuracy: 0.9592
Epoch 23/30
1092/1092 [==============================] - 314s 288ms/step - loss: 0.1140 -
accuracy: 0.9598
Epoch 24/30
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0912 -
accuracy: 0.9694
Epoch 25/30
1092/1092 [==============================] - 320s 293ms/step - loss: 0.1071 -
accuracy: 0.9655
Epoch 26/30
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0988 -
accuracy: 0.9694
Epoch 27/30
1092/1092 [==============================] - 321s 293ms/step - loss: 0.1000 -
accuracy: 0.9691
Epoch 28/30
1092/1092 [==============================] - 319s 293ms/step - loss: 0.0859 -
accuracy: 0.9731
Epoch 29/30
1092/1092 [==============================] - 337s 308ms/step - loss: 0.0836 -
```

```
accuracy: 0.9745
Epoch 30/30
1092/1092 [==============================] - 342s 313ms/step - loss: 0.0764 -
accuracy: 0.9777
```

In [72]:

```
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 178, 178, 32)      896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 89, 89, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 87, 87, 64)        18496
_________________________________________________________________
flatten_4 (Flatten)          (None, 484416)            0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 484417
=================================================================
Total params: 503,809
Trainable params: 503,809
Non-trainable params: 0
_________________________________________________________________
```

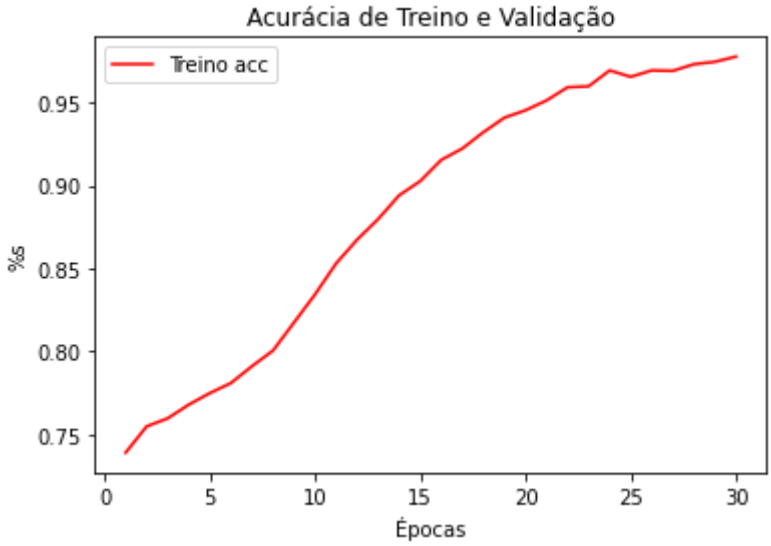
In [73]:

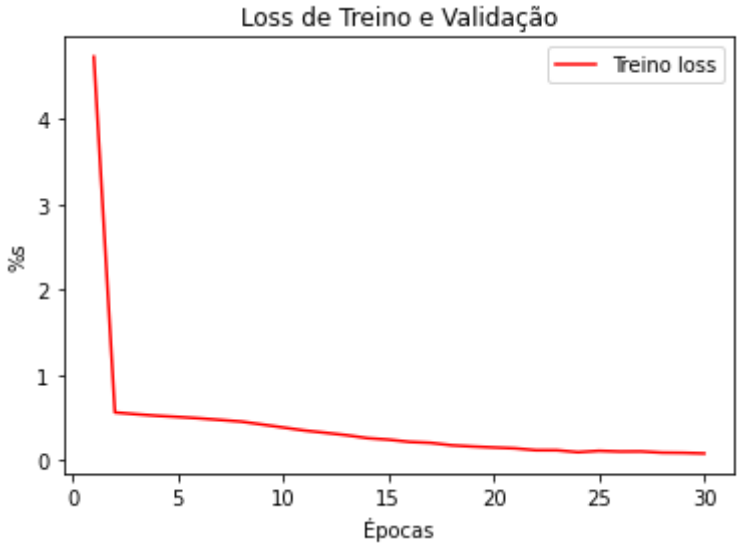
```
#https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=pt-br#al
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, "conversao_01_09")
```

Visualização de Resultados

In [74]:

```
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 [75]:
```
#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 [76]:
```
test_loss, test_acc = model.evaluate(test_dataset)
print(f"Test accuracy: {test_acc:.3f}")
```

```
16/16 [==============================] - 1s 59ms/step - loss: 2.7442 - accura
cy: 0.7376
Test accuracy: 0.738
```

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/