Classificaço de Patologias usando Imagens Médicas

Carregar imagens do diretório

In [53]:
```python
import os
current_dir = os.path.abspath(os.getcwd())
```

Converter base de dados para treino, validação e teste

In [54]:
```python
#cria nova pasta para cachorros e gatos atendendo a estrutura do Keras/Tensor
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 [55]:
```python
import tensorflow as tf

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

```
[]
2.6.1
```

In [56]:
```python
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 [57]:
```python
#
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 [58]:
```python
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=['a
```

In [59]:
```python
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 [==============================] - 342s 311ms/step - loss: 1.7896 -
accuracy: 0.7477
Epoch 2/30
1092/1092 [==============================] - 356s 326ms/step - loss: 0.5553 -
accuracy: 0.7550
Epoch 3/30
1092/1092 [==============================] - 367s 336ms/step - loss: 0.5379 -
accuracy: 0.7598
Epoch 4/30
1092/1092 [==============================] - 338s 310ms/step - loss: 0.5254 -
accuracy: 0.7662
Epoch 5/30
1092/1092 [==============================] - 385s 353ms/step - loss: 0.5104 -
accuracy: 0.7721
Epoch 6/30
1092/1092 [==============================] - 401s 367ms/step - loss: 0.4858 -
accuracy: 0.7830
Epoch 7/30
1092/1092 [==============================] - 352s 323ms/step - loss: 0.4542 -
accuracy: 0.7982
Epoch 8/30
```

```
1092/1092 [==============================] - 333s 305ms/step - loss: 0.4348 -
accuracy: 0.8088
Epoch 9/30
1092/1092 [==============================] - 338s 309ms/step - loss: 0.4159 -
accuracy: 0.8177
Epoch 10/30
1092/1092 [==============================] - 344s 314ms/step - loss: 0.3980 -
accuracy: 0.8265
Epoch 11/30
1092/1092 [==============================] - 329s 301ms/step - loss: 0.3913 -
accuracy: 0.8298
Epoch 12/30
1092/1092 [==============================] - 370s 339ms/step - loss: 0.3588 -
accuracy: 0.8444
Epoch 13/30
1092/1092 [==============================] - 340s 312ms/step - loss: 0.3341 -
accuracy: 0.8562
Epoch 14/30
1092/1092 [==============================] - 359s 329ms/step - loss: 0.3124 -
accuracy: 0.8654
Epoch 15/30
1092/1092 [==============================] - 374s 343ms/step - loss: 0.3008 -
accuracy: 0.8726
Epoch 16/30
1092/1092 [==============================] - 399s 365ms/step - loss: 0.2791 -
accuracy: 0.8835
Epoch 17/30
1092/1092 [==============================] - 483s 442ms/step - loss: 0.2524 -
accuracy: 0.8957
Epoch 18/30
1092/1092 [==============================] - 412s 376ms/step - loss: 0.2642 -
accuracy: 0.8918
Epoch 19/30
1092/1092 [==============================] - 337s 309ms/step - loss: 0.2274 -
accuracy: 0.9105
Epoch 20/30
1092/1092 [==============================] - 320s 293ms/step - loss: 0.2126 -
accuracy: 0.9137
Epoch 21/30
1092/1092 [==============================] - 326s 299ms/step - loss: 0.1863 -
accuracy: 0.9249
Epoch 22/30
1092/1092 [==============================] - 320s 293ms/step - loss: 0.1831 -
accuracy: 0.9285
Epoch 23/30
1092/1092 [==============================] - 320s 293ms/step - loss: 0.1565 -
accuracy: 0.9389
Epoch 24/30
1092/1092 [==============================] - 320s 293ms/step - loss: 0.1566 -
accuracy: 0.9403
Epoch 25/30
1092/1092 [==============================] - 319s 292ms/step - loss: 0.1403 -
accuracy: 0.9475
Epoch 26/30
1092/1092 [==============================] - 320s 292ms/step - loss: 0.1603 -
accuracy: 0.9420
Epoch 27/30
1092/1092 [==============================] - 315s 289ms/step - loss: 0.1417 -
accuracy: 0.9490
Epoch 28/30
1092/1092 [==============================] - 315s 288ms/step - loss: 0.1269 -
accuracy: 0.9532
Epoch 29/30
1092/1092 [==============================] - 315s 288ms/step - loss: 0.1149 -
```

```
accuracy: 0.9584
Epoch 30/30
1092/1092 [==============================] - 315s 288ms/step - loss: 0.1199 -
accuracy: 0.9583
```

In [60]:

```python
model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 178, 178, 32)	896
max_pooling2d_3 (MaxPooling2	(None, 89, 89, 32)	0
conv2d_7 (Conv2D)	(None, 87, 87, 64)	18496
flatten_3 (Flatten)	(None, 484416)	0
dense_3 (Dense)	(None, 1)	484417

```
Total params: 503,809
Trainable params: 503,809
Non-trainable params: 0
```

In [61]:

```python
#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_08")
```

Visualização de Resultados

In [62]:

```python
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()
```

Acurácia de Treino e Validação

Loss de Treino e Validação
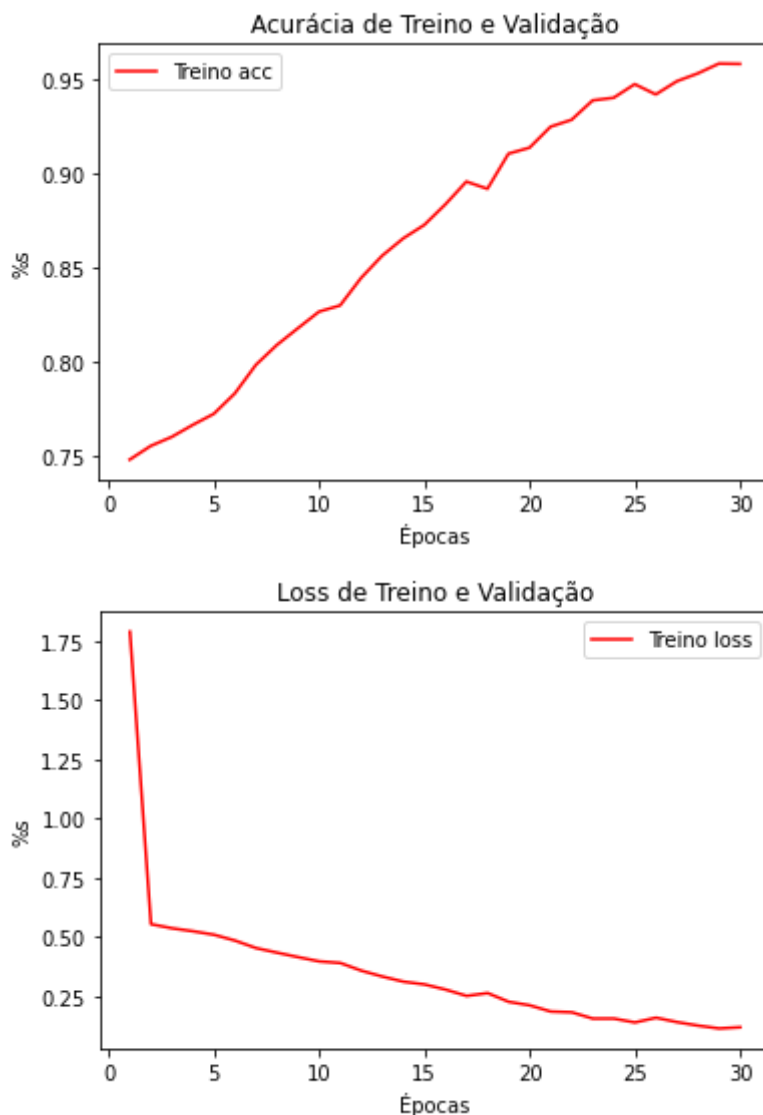
Resultados do Conjunto de Teste

```
In [63]:    #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_j:
            # serialize weights to HDF5
            #model.save_weights("classificacao01.h5")
            #print("Saved model to disk")
```

```
In [64]:    test_loss, test_acc = model.evaluate(test_dataset)
            print(f"Test accuracy: {test_acc:.3f}")
```

```
16/16 [==============================] - 3s 69ms/step - loss: 2.1487 - accura
cy: 0.6839
Test accuracy: 0.684
```

```
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/