Classificaço de Patologias usando Imagens Médicas

Carregar imagens do diretório

In [25]:
```
import os
current_dir = os.path.abspath(os.getcwd())
```

Converter base de dados para treino, validação e teste

In [26]:
```
#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"

model_filepath = "keras/classificacao_02_05.keras"
conversao_path = "conversao/conversao_02_05"
```

Fazer o Tensorflow carregar as imagens para a RNA

In [27]:
```
import tensorflow as tf

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

```
[]
2.6.1
```

In [28]:
```
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, 1

validation_dataset = image_dataset_from_directory(val_folder,image_size=(180,

test_dataset = image_dataset_from_directory(test_folder, image_size=(180, 180
```

```
Found 34931 files belonging to 2 classes.
Found 16 files belonging to 2 classes.
Found 484 files belonging to 2 classes.
```

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

```
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'))
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 [31]:

```
from tensorflow.keras.callbacks import ModelCheckpoint

callbacks = [
    ModelCheckpoint(
        filepath = model_filepath,
        save_best_only = True,
        monitor = "val_loss"
    )
]

history = model.fit(
    train_dataset,
    epochs=100,
    validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/100
1092/1092 [==============================] - 344s 315ms/step - loss: 0.3935 -
accuracy: 0.8255 - val_loss: 0.3116 - val_accuracy: 0.8125
Epoch 2/100
1092/1092 [==============================] - 334s 306ms/step - loss: 0.2572 -
accuracy: 0.9008 - val_loss: 0.2713 - val_accuracy: 0.8125
Epoch 3/100
1092/1092 [==============================] - 331s 303ms/step - loss: 0.1957 -
accuracy: 0.9284 - val_loss: 0.1631 - val_accuracy: 0.9375
Epoch 4/100
1092/1092 [==============================] - 331s 303ms/step - loss: 0.1386 -
accuracy: 0.9524 - val_loss: 0.0924 - val_accuracy: 1.0000
Epoch 5/100
1092/1092 [==============================] - 327s 300ms/step - loss: 0.0854 -
accuracy: 0.9726 - val_loss: 0.0102 - val_accuracy: 1.0000
Epoch 6/100
1092/1092 [==============================] - 324s 296ms/step - loss: 0.0527 -
accuracy: 0.9830 - val_loss: 0.0103 - val_accuracy: 1.0000
Epoch 7/100
1092/1092 [==============================] - 323s 296ms/step - loss: 0.0338 -
accuracy: 0.9892 - val_loss: 0.0498 - val_accuracy: 1.0000
Epoch 8/100
1092/1092 [==============================] - 322s 295ms/step - loss: 0.0215 -
```

```
accuracy: 0.9936 - val_loss: 0.1570 - val_accuracy: 0.9375
Epoch 9/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0184 -
accuracy: 0.9942 - val_loss: 0.0113 - val_accuracy: 1.0000
Epoch 10/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0159 -
accuracy: 0.9955 - val_loss: 7.3032e-06 - val_accuracy: 1.0000
Epoch 11/100
1092/1092 [==============================] - 321s 293ms/step - loss: 0.0177 -
accuracy: 0.9949 - val_loss: 2.0642e-05 - val_accuracy: 1.0000
Epoch 12/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0120 -
accuracy: 0.9964 - val_loss: 6.8290e-05 - val_accuracy: 1.0000
Epoch 13/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0091 -
accuracy: 0.9974 - val_loss: 4.6673e-04 - val_accuracy: 1.0000
Epoch 14/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0102 -
accuracy: 0.9972 - val_loss: 6.8387e-04 - val_accuracy: 1.0000
Epoch 15/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0115 -
accuracy: 0.9964 - val_loss: 0.0042 - val_accuracy: 1.0000
Epoch 16/100
1092/1092 [==============================] - 323s 295ms/step - loss: 0.0086 -
accuracy: 0.9974 - val_loss: 1.5802e-04 - val_accuracy: 1.0000
Epoch 17/100
1092/1092 [==============================] - 322s 294ms/step - loss: 0.0099 -
accuracy: 0.9971 - val_loss: 0.0017 - val_accuracy: 1.0000
Epoch 18/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0088 -
accuracy: 0.9975 - val_loss: 9.9075e-06 - val_accuracy: 1.0000
Epoch 19/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0091 -
accuracy: 0.9975 - val_loss: 3.8737e-04 - val_accuracy: 1.0000
Epoch 20/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0068 -
accuracy: 0.9988 - val_loss: 2.6043e-04 - val_accuracy: 1.0000
Epoch 21/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0072 -
accuracy: 0.9981 - val_loss: 1.1479e-05 - val_accuracy: 1.0000
Epoch 22/100
1092/1092 [==============================] - 332s 304ms/step - loss: 0.0099 -
accuracy: 0.9971 - val_loss: 1.3398e-04 - val_accuracy: 1.0000
Epoch 23/100
1092/1092 [==============================] - 340s 311ms/step - loss: 0.0077 -
accuracy: 0.9980 - val_loss: 6.7631e-07 - val_accuracy: 1.0000
Epoch 24/100
1092/1092 [==============================] - 339s 310ms/step - loss: 0.0075 -
accuracy: 0.9984 - val_loss: 4.5198e-07 - val_accuracy: 1.0000
Epoch 25/100
1092/1092 [==============================] - 340s 311ms/step - loss: 0.0069 -
accuracy: 0.9984 - val_loss: 4.4490e-05 - val_accuracy: 1.0000
Epoch 26/100
1092/1092 [==============================] - 332s 304ms/step - loss: 0.0060 -
accuracy: 0.9984 - val_loss: 5.5844e-06 - val_accuracy: 1.0000
Epoch 27/100
1092/1092 [==============================] - 325s 298ms/step - loss: 0.0052 -
accuracy: 0.9991 - val_loss: 1.7703e-05 - val_accuracy: 1.0000
Epoch 28/100
1092/1092 [==============================] - 325s 298ms/step - loss: 0.0075 -
accuracy: 0.9986 - val_loss: 3.5046e-05 - val_accuracy: 1.0000
Epoch 29/100
1092/1092 [==============================] - 322s 295ms/step - loss: 0.0080 -
accuracy: 0.9982 - val_loss: 1.3964e-06 - val_accuracy: 1.0000
```

```
Epoch 30/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0068 -
accuracy: 0.9983 - val_loss: 1.4012e-06 - val_accuracy: 1.0000
Epoch 31/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0056 -
accuracy: 0.9992 - val_loss: 4.6282e-07 - val_accuracy: 1.0000
Epoch 32/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0062 -
accuracy: 0.9990 - val_loss: 5.8975e-07 - val_accuracy: 1.0000
Epoch 33/100
1092/1092 [==============================] - 321s 293ms/step - loss: 0.0070 -
accuracy: 0.9988 - val_loss: 2.8528e-04 - val_accuracy: 1.0000
Epoch 34/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0068 -
accuracy: 0.9988 - val_loss: 1.0006e-04 - val_accuracy: 1.0000
Epoch 35/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0054 -
accuracy: 0.9991 - val_loss: 1.2965e-04 - val_accuracy: 1.0000
Epoch 36/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0058 -
accuracy: 0.9983 - val_loss: 0.0012 - val_accuracy: 1.0000
Epoch 37/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0074 -
accuracy: 0.9985 - val_loss: 2.3372e-04 - val_accuracy: 1.0000
Epoch 38/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0070 -
accuracy: 0.9989 - val_loss: 8.2097e-06 - val_accuracy: 1.0000
Epoch 39/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0056 -
accuracy: 0.9992 - val_loss: 8.0858e-06 - val_accuracy: 1.0000
Epoch 40/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0048 -
accuracy: 0.9993 - val_loss: 4.1088e-07 - val_accuracy: 1.0000
Epoch 41/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0051 -
accuracy: 0.9991 - val_loss: 8.2491e-05 - val_accuracy: 1.0000
Epoch 42/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0059 -
accuracy: 0.9990 - val_loss: 6.7173e-06 - val_accuracy: 1.0000
Epoch 43/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0047 -
accuracy: 0.9991 - val_loss: 8.6679e-07 - val_accuracy: 1.0000
Epoch 44/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0060 -
accuracy: 0.9991 - val_loss: 1.4989e-06 - val_accuracy: 1.0000
Epoch 45/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0062 -
accuracy: 0.9990 - val_loss: 2.0805e-08 - val_accuracy: 1.0000
Epoch 46/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0059 -
accuracy: 0.9990 - val_loss: 3.2363e-07 - val_accuracy: 1.0000
Epoch 47/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0052 -
accuracy: 0.9992 - val_loss: 4.0642e-07 - val_accuracy: 1.0000
Epoch 48/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0042 -
accuracy: 0.9992 - val_loss: 3.1203e-08 - val_accuracy: 1.0000
Epoch 49/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0041 -
accuracy: 0.9994 - val_loss: 1.6831e-06 - val_accuracy: 1.0000
Epoch 50/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0062 -
accuracy: 0.9991 - val_loss: 2.4143e-06 - val_accuracy: 1.0000
Epoch 51/100
```

```
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0059 -
accuracy: 0.9988 - val_loss: 4.7077e-05 - val_accuracy: 1.0000
Epoch 52/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0050 -
accuracy: 0.9993 - val_loss: 5.0560e-07 - val_accuracy: 1.0000
Epoch 53/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0042 -
accuracy: 0.9995 - val_loss: 1.1581e-06 - val_accuracy: 1.0000
Epoch 54/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0035 -
accuracy: 0.9994 - val_loss: 3.5953e-06 - val_accuracy: 1.0000
Epoch 55/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0039 -
accuracy: 0.9994 - val_loss: 1.5893e-06 - val_accuracy: 1.0000
Epoch 56/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0040 -
accuracy: 0.9993 - val_loss: 9.2422e-06 - val_accuracy: 1.0000
Epoch 57/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0045 -
accuracy: 0.9992 - val_loss: 9.3071e-08 - val_accuracy: 1.0000
Epoch 58/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0036 -
accuracy: 0.9996 - val_loss: 4.8299e-05 - val_accuracy: 1.0000
Epoch 59/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0035 -
accuracy: 0.9994 - val_loss: 1.2323e-06 - val_accuracy: 1.0000
Epoch 60/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0044 -
accuracy: 0.9995 - val_loss: 8.1400e-06 - val_accuracy: 1.0000
Epoch 61/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0034 -
accuracy: 0.9994 - val_loss: 1.7197e-05 - val_accuracy: 1.0000
Epoch 62/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0065 -
accuracy: 0.9989 - val_loss: 2.1268e-05 - val_accuracy: 1.0000
Epoch 63/100
1092/1092 [==============================] - 317s 291ms/step - loss: 0.0039 -
accuracy: 0.9995 - val_loss: 9.9137e-06 - val_accuracy: 1.0000
Epoch 64/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0029 -
accuracy: 0.9996 - val_loss: 1.2899e-05 - val_accuracy: 1.0000
Epoch 65/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0042 -
accuracy: 0.9993 - val_loss: 2.3994e-06 - val_accuracy: 1.0000
Epoch 66/100
1092/1092 [==============================] - 318s 291ms/step - loss: 0.0031 -
accuracy: 0.9995 - val_loss: 2.1254e-06 - val_accuracy: 1.0000
Epoch 67/100
1092/1092 [==============================] - 317s 291ms/step - loss: 0.0018 -
accuracy: 0.9995 - val_loss: 8.1328e-07 - val_accuracy: 1.0000
Epoch 68/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0033 -
accuracy: 0.9994 - val_loss: 1.3556e-05 - val_accuracy: 1.0000
Epoch 69/100
1092/1092 [==============================] - 319s 293ms/step - loss: 0.0025 -
accuracy: 0.9997 - val_loss: 9.5307e-07 - val_accuracy: 1.0000
Epoch 70/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0041 -
accuracy: 0.9993 - val_loss: 8.1051e-06 - val_accuracy: 1.0000
Epoch 71/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0029 -
accuracy: 0.9997 - val_loss: 1.7381e-06 - val_accuracy: 1.0000
Epoch 72/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0049 -
```

```
accuracy: 0.9991 - val_loss: 5.1666e-05 - val_accuracy: 1.0000
Epoch 73/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0037 -
accuracy: 0.9991 - val_loss: 4.7827e-05 - val_accuracy: 1.0000
Epoch 74/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0029 -
accuracy: 0.9997 - val_loss: 1.9835e-04 - val_accuracy: 1.0000
Epoch 75/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0053 -
accuracy: 0.9991 - val_loss: 0.0015 - val_accuracy: 1.0000
Epoch 76/100
1092/1092 [==============================] - 322s 295ms/step - loss: 0.0040 -
accuracy: 0.9995 - val_loss: 3.8716e-05 - val_accuracy: 1.0000
Epoch 77/100
1092/1092 [==============================] - 321s 293ms/step - loss: 0.0049 -
accuracy: 0.9994 - val_loss: 1.2568e-06 - val_accuracy: 1.0000
Epoch 78/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0037 -
accuracy: 0.9994 - val_loss: 3.6274e-05 - val_accuracy: 1.0000
Epoch 79/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0032 -
accuracy: 0.9993 - val_loss: 7.7390e-06 - val_accuracy: 1.0000
Epoch 80/100
1092/1092 [==============================] - 321s 293ms/step - loss: 0.0055 -
accuracy: 0.9994 - val_loss: 2.3285e-04 - val_accuracy: 1.0000
Epoch 81/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0034 -
accuracy: 0.9993 - val_loss: 0.0012 - val_accuracy: 1.0000
Epoch 82/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0042 -
accuracy: 0.9993 - val_loss: 1.1437e-04 - val_accuracy: 1.0000
Epoch 83/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0032 -
accuracy: 0.9995 - val_loss: 9.1555e-08 - val_accuracy: 1.0000
Epoch 84/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0034 -
accuracy: 0.9995 - val_loss: 9.0984e-07 - val_accuracy: 1.0000
Epoch 85/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0032 -
accuracy: 0.9994 - val_loss: 2.7898e-08 - val_accuracy: 1.0000
Epoch 86/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0028 -
accuracy: 0.9995 - val_loss: 4.7681e-07 - val_accuracy: 1.0000
Epoch 87/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0035 -
accuracy: 0.9994 - val_loss: 3.7821e-07 - val_accuracy: 1.0000
Epoch 88/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0027 -
accuracy: 0.9996 - val_loss: 1.3743e-07 - val_accuracy: 1.0000
Epoch 89/100
1092/1092 [==============================] - 321s 294ms/step - loss: 0.0035 -
accuracy: 0.9995 - val_loss: 2.4086e-04 - val_accuracy: 1.0000
Epoch 90/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0030 -
accuracy: 0.9995 - val_loss: 1.6605e-06 - val_accuracy: 1.0000
Epoch 91/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0033 -
accuracy: 0.9996 - val_loss: 6.0598e-07 - val_accuracy: 1.0000
Epoch 92/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0032 -
accuracy: 0.9995 - val_loss: 2.0479e-07 - val_accuracy: 1.0000
Epoch 93/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0033 -
accuracy: 0.9994 - val_loss: 1.6943e-06 - val_accuracy: 1.0000
```

```
Epoch 94/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0025 -
accuracy: 0.9996 - val_loss: 2.6327e-07 - val_accuracy: 1.0000
Epoch 95/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0026 -
accuracy: 0.9996 - val_loss: 8.5314e-06 - val_accuracy: 1.0000
Epoch 96/100
1092/1092 [==============================] - 320s 293ms/step - loss: 0.0022 -
accuracy: 0.9997 - val_loss: 9.3352e-05 - val_accuracy: 1.0000
Epoch 97/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0027 -
accuracy: 0.9995 - val_loss: 1.0828e-05 - val_accuracy: 1.0000
Epoch 98/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0032 -
accuracy: 0.9995 - val_loss: 1.6004e-04 - val_accuracy: 1.0000
Epoch 99/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0032 -
accuracy: 0.9995 - val_loss: 5.4909e-04 - val_accuracy: 1.0000
Epoch 100/100
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0021 -
accuracy: 0.9995 - val_loss: 1.9363e-05 - val_accuracy: 1.0000
```

In [32]:

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling_2 (Rescaling)      (None, 180, 180, 3)       0

_________________________________________________________________
conv2d_4 (Conv2D)            (None, 178, 178, 32)      896

_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (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 [33]:

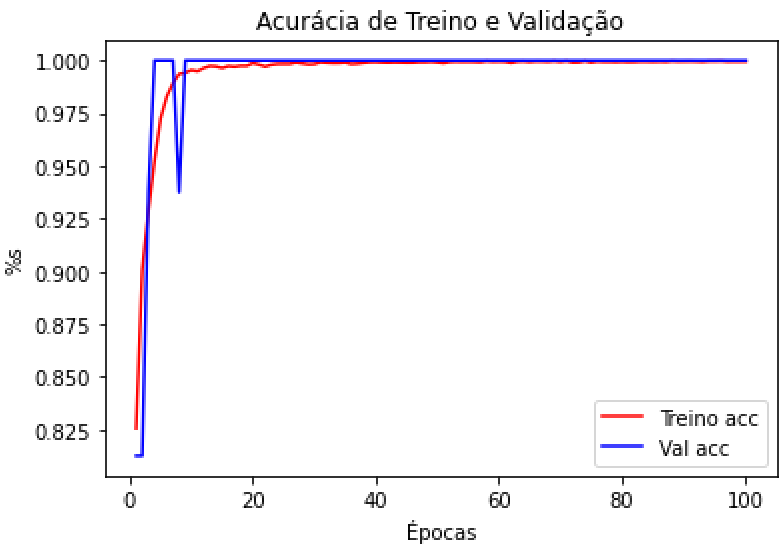
```
#https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=pt-br#alt
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, conversao_path)
```

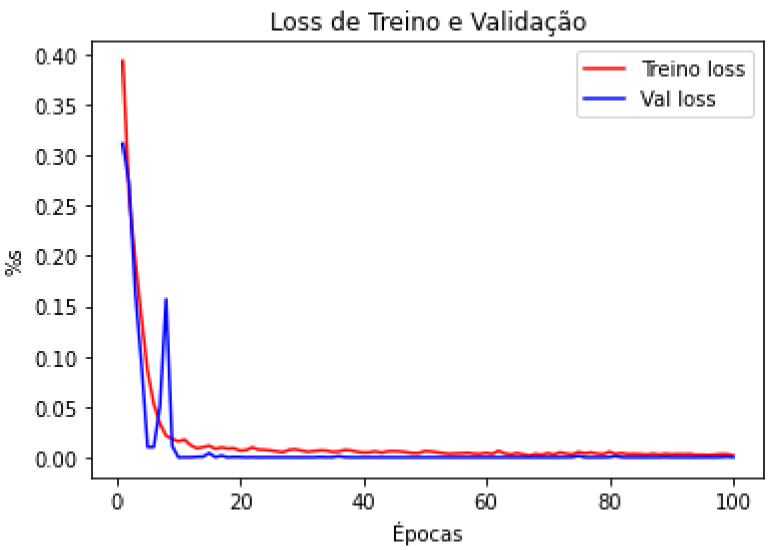
Visualização de Resultados

In [34]:

```
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 [35]:
```
from tensorflow import keras
model = keras.models.load_model(model_filepath)
```

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

```
16/16 [==============================] - 1s 57ms/step - loss: 0.0362 - 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/