Classificaço de Patologias usando Imagens Médicas

Carregar imagens do diretório

In [146…

```
import os
current_dir = os.path.abspath(os.getcwd())
```

Converter base de dados para treino, validação e teste

In [147…

```
#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 [148…

```
import tensorflow as tf

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

```
[]
2.6.1
```

In [149…

```
from tensorflow.keras.utils import image_dataset_from_directory
#image_dataset_from_directory monta uma estrutura de dados com imagens 180x1
# 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 [150…

```
#
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 [151…

```
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 [152…

```
from tensorflow.keras.callbacks import ModelCheckpoint

callbacks = [
    ModelCheckpoint(
        filepath="classificacao14.keras",
        save_best_only=True,
        monitor="loss"
    )
]

history = model.fit(
    train_dataset,
    epochs=50,
    #validation_data=validation_dataset,
    callbacks=callbacks)
```

```
Epoch 1/50
1092/1092 [==============================] - 326s 298ms/step - loss: 1.4464 -
accuracy: 0.7813
Epoch 2/50
1092/1092 [==============================] - 324s 297ms/step - loss: 0.3577 -
accuracy: 0.8448
Epoch 3/50
1092/1092 [==============================] - 323s 295ms/step - loss: 0.3208 -
accuracy: 0.8694
Epoch 4/50
1092/1092 [==============================] - 322s 294ms/step - loss: 0.2944 -
accuracy: 0.8825
Epoch 5/50
1092/1092 [==============================] - 328s 301ms/step - loss: 0.2693 -
accuracy: 0.8951
Epoch 6/50
1092/1092 [==============================] - 329s 301ms/step - loss: 0.2290 -
accuracy: 0.9108
Epoch 7/50
1092/1092 [==============================] - 329s 301ms/step - loss: 0.2072 -
accuracy: 0.9203
Epoch 8/50
```

```
1092/1092 [==============================] - 323s 296ms/step - loss: 0.1745 -
accuracy: 0.9344
Epoch 9/50
1092/1092 [==============================] - 318s 291ms/step - loss: 0.1451 -
accuracy: 0.9467
Epoch 10/50
1092/1092 [==============================] - 318s 291ms/step - loss: 0.1379 -
accuracy: 0.9512
Epoch 11/50
1092/1092 [==============================] - 318s 291ms/step - loss: 0.1082 -
accuracy: 0.9615
Epoch 12/50
1092/1092 [==============================] - 317s 290ms/step - loss: 0.1053 -
accuracy: 0.9636
Epoch 13/50
1092/1092 [==============================] - 317s 290ms/step - loss: 0.1007 -
accuracy: 0.9650
Epoch 14/50
1092/1092 [==============================] - 314s 288ms/step - loss: 0.0889 -
accuracy: 0.9705
Epoch 15/50
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0836 -
accuracy: 0.9727
Epoch 16/50
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0882 -
accuracy: 0.9733
Epoch 17/50
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0784 -
accuracy: 0.9774
Epoch 18/50
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0765 -
accuracy: 0.9782
Epoch 19/50
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0762 -
accuracy: 0.9782
Epoch 20/50
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0706 -
accuracy: 0.9800
Epoch 21/50
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0738 -
accuracy: 0.9799
Epoch 22/50
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0695 -
accuracy: 0.9823
Epoch 23/50
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0615 -
accuracy: 0.9844
Epoch 24/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0718 -
accuracy: 0.9829
Epoch 25/50
1092/1092 [==============================] - 314s 287ms/step - loss: 0.0663 -
accuracy: 0.9837
Epoch 26/50
1092/1092 [==============================] - 319s 292ms/step - loss: 0.0608 -
accuracy: 0.9861
Epoch 27/50
1092/1092 [==============================] - 317s 291ms/step - loss: 0.0568 -
accuracy: 0.9868
Epoch 28/50
1092/1092 [==============================] - 317s 290ms/step - loss: 0.0682 -
accuracy: 0.9853
Epoch 29/50
1092/1092 [==============================] - 316s 290ms/step - loss: 0.0701 -
```

```
accuracy: 0.9855
Epoch 30/50
1092/1092 [==============================] - 316s 289ms/step - loss: 0.0570 -
accuracy: 0.9875
Epoch 31/50
1092/1092 [==============================] - 316s 289ms/step - loss: 0.0619 -
accuracy: 0.9885
Epoch 32/50
1092/1092 [==============================] - 316s 289ms/step - loss: 0.0693 -
accuracy: 0.9874
Epoch 33/50
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0590 -
accuracy: 0.9887
Epoch 34/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0551 -
accuracy: 0.9891
Epoch 35/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0607 -
accuracy: 0.9895
Epoch 36/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0673 -
accuracy: 0.9891
Epoch 37/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0614 -
accuracy: 0.9901
Epoch 38/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0814 -
accuracy: 0.9885
Epoch 39/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0695 -
accuracy: 0.9908
Epoch 40/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0530 -
accuracy: 0.9916
Epoch 41/50
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0543 -
accuracy: 0.9913
Epoch 42/50
1092/1092 [==============================] - 313s 286ms/step - loss: 0.0679 -
accuracy: 0.9908
Epoch 43/50
1092/1092 [==============================] - 313s 287ms/step - loss: 0.0522 -
accuracy: 0.9924
Epoch 44/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0610 -
accuracy: 0.9918
Epoch 45/50
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0840 -
accuracy: 0.9902
Epoch 46/50
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0580 -
accuracy: 0.9923
Epoch 47/50
1092/1092 [==============================] - 312s 286ms/step - loss: 0.0532 -
accuracy: 0.9931
Epoch 48/50
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0671 -
accuracy: 0.9923
Epoch 49/50
1092/1092 [==============================] - 311s 285ms/step - loss: 0.0666 -
accuracy: 0.9917
Epoch 50/50
1092/1092 [==============================] - 312s 285ms/step - loss: 0.0574 -
accuracy: 0.9933
```

In [153…

```
model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 178, 178, 32)      896
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 89, 89, 32)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 87, 87, 64)        18496
_________________________________________________________________
flatten_12 (Flatten)         (None, 484416)            0
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 484417
=================================================================
Total params: 503,809
Trainable params: 503,809
Non-trainable params: 0
_________________________________________________________________
```

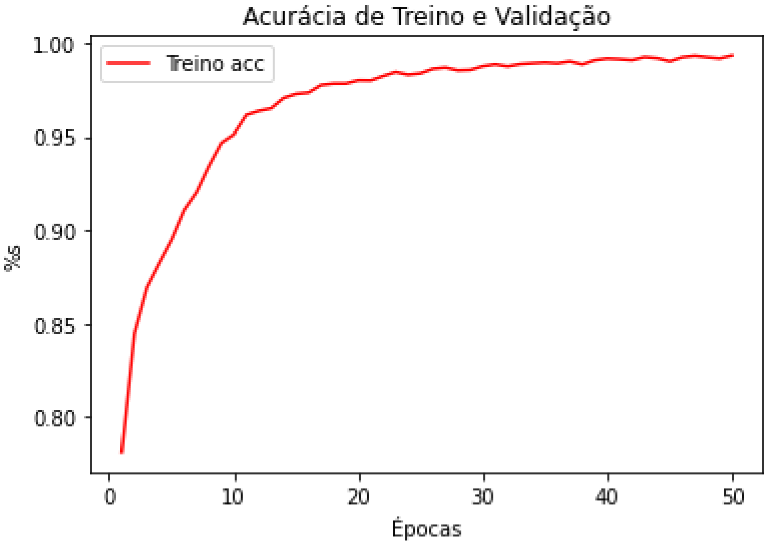
In [154…

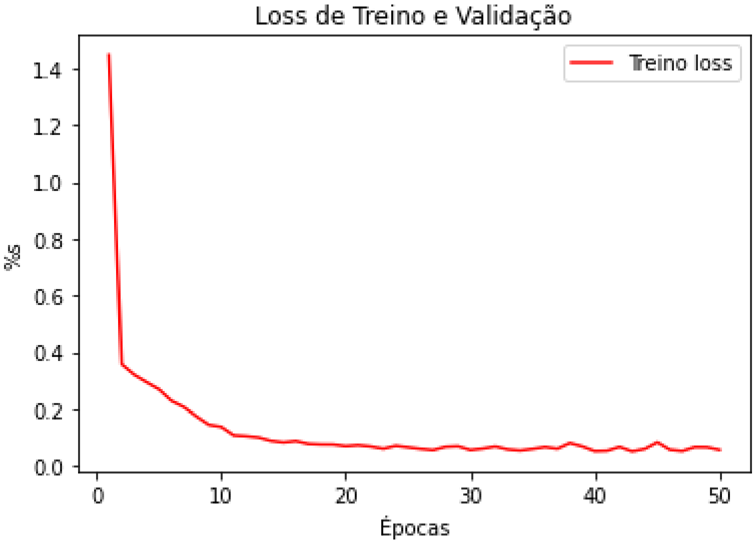
```
#https://www.tensorflow.org/js/tutorials/conversion/import_keras?hl=pt-br#al
import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, "conversao_01_14")
```

Visualização de Resultados

In [155…

```
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 [156…

```
#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 [157…

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

```
16/16 [==============================] - 2s 65ms/step - loss: 0.7466 - accura
cy: 0.9649
Test accuracy: 0.965
```

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/