

Curso: Redes Neurais e Deep Learning

Prof. Denilson Alves Pereira <https://sites.google.com/ufla.br/denilsonpereira/> Departamento de Ciência da Computação - Instituto de Ciências Exatas e Tecnológicas - Universidade Federal de Lavras

Atividade Prática 02

Instruções:

1. Siga os passos indicados em cada célula abaixo para completar a atividade.
2. Você deve inserir código somente entre as linhas marcadas com **INICIE O CÓDIGO AQUI** e **TERMINE O CÓDIGO AQUI**. Há uma indicação de quantas linhas de código são necessárias.
3. Em alguns pontos, confira o resultado esperado conforme marcado com **SAÍDA ESPERADA**.

Tempo estimado para execução: 1 hora

Versão: Junho, 2021

+ Código

+ Texto

O Problema a ser Resolvido

O objetivo da atividade é elaborar uma rede neural para predizer se o animal de uma imagem é um gato ou um cachorro.

Será usado um dataset contendo 8.000 imagens para treinamento e 4.000 para teste, sendo a metade delas de gatos e a outra metade, de cachorros. As imagens têm tamanhos diversos e estão no formato RGB.

Você vai praticar as seguintes habilidades:

- Ler arquivos de imagens e convertê-los para vetores de pixels
- Utilizar um gerador de dados aumentados para expandir o conjunto de imagens para treinamento da rede
- Configurar uma rede neural simples para um problema de classificação multiclasse de imagens. Embora só existam duas classes, a saída será configurada para predizer "cat" se a imagem é de um gato ou "dog" se a imagem é de um cachorro.

É importante destacar que a arquitetura de rede neural utilizada no nesta atividade **não** é a mais indicada para classificar imagens. Para uma melhor eficácia, devem ser adicionadas algumas camadas de Redes Neurais Convolucionais (CNN), que é assunto do próximo curso.

Pacotes

```

!pip install --upgrade pip
!pip install pandas
!pip install matplotlib
!pip install keras_preprocessing
!pip install tensorflow

```

```

Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-packages (24.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from p
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from ma
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (fr

```

Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from matplotlib>=3.5.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: keras_preprocessing in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.16.2)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: gast!=0.5.0,!0.5.1,!0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: ml-dtypes~=0.3.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: protobuf!=4.21.0,!4.21.1,!4.21.2,!4.21.3,!4.21.4,!4.21.5,<5.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: tensorboard<2.17,>=2.16 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: keras>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow>=2.10.0 in /usr/local/lib/python3.10/dist-packages)

```
# Importando o dataset diretamente do Kaggle
```

```
!kaggle datasets download -d d4rklucif3r/cat-and-dogs -p /content/  
!unzip -o /content/cat-and-dogs.zip -d /content/
```

➡ **A saída de streaming foi truncada nas últimas 5000 linhas.**

```
inflating: /content/dataset/training_set/cats/cat.37.jpg  
inflating: /content/dataset/training_set/cats/cat.370.jpg  
inflating: /content/dataset/training_set/cats/cat.3700.jpg  
inflating: /content/dataset/training_set/cats/cat.3701.jpg  
inflating: /content/dataset/training_set/cats/cat.3702.jpg  
inflating: /content/dataset/training_set/cats/cat.3703.jpg  
inflating: /content/dataset/training_set/cats/cat.3704.jpg  
inflating: /content/dataset/training_set/cats/cat.3705.jpg  
inflating: /content/dataset/training_set/cats/cat.3706.jpg  
inflating: /content/dataset/training_set/cats/cat.3707.jpg  
inflating: /content/dataset/training_set/cats/cat.3708.jpg  
inflating: /content/dataset/training_set/cats/cat.3709.jpg  
inflating: /content/dataset/training_set/cats/cat.371.jpg  
inflating: /content/dataset/training_set/cats/cat.3710.jpg  
inflating: /content/dataset/training_set/cats/cat.3711.jpg  
inflating: /content/dataset/training_set/cats/cat.3712.jpg  
inflating: /content/dataset/training_set/cats/cat.3713.jpg  
inflating: /content/dataset/training_set/cats/cat.3714.jpg  
inflating: /content/dataset/training_set/cats/cat.3715.jpg  
inflating: /content/dataset/training_set/cats/cat.3716.jpg  
inflating: /content/dataset/training_set/cats/cat.3717.jpg  
inflating: /content/dataset/training_set/cats/cat.3718.jpg  
inflating: /content/dataset/training_set/cats/cat.3719.jpg
```

inflating: /content/dataset/training_set/cats/cat.372.jpg
inflating: /content/dataset/training_set/cats/cat.3720.jpg
inflating: /content/dataset/training_set/cats/cat.3721.jpg
inflating: /content/dataset/training_set/cats/cat.3722.jpg
inflating: /content/dataset/training_set/cats/cat.3723.jpg
inflating: /content/dataset/training_set/cats/cat.3724.jpg
inflating: /content/dataset/training_set/cats/cat.3725.jpg
inflating: /content/dataset/training_set/cats/cat.3726.jpg
inflating: /content/dataset/training_set/cats/cat.3727.jpg
inflating: /content/dataset/training_set/cats/cat.3728.jpg
inflating: /content/dataset/training_set/cats/cat.3729.jpg
inflating: /content/dataset/training_set/cats/cat.373.jpg
inflating: /content/dataset/training_set/cats/cat.3730.jpg
inflating: /content/dataset/training_set/cats/cat.3731.jpg
inflating: /content/dataset/training_set/cats/cat.3732.jpg
inflating: /content/dataset/training_set/cats/cat.3733.jpg
inflating: /content/dataset/training_set/cats/cat.3734.jpg
inflating: /content/dataset/training_set/cats/cat.3735.jpg
inflating: /content/dataset/training_set/cats/cat.3736.jpg
inflating: /content/dataset/training_set/cats/cat.3737.jpg
inflating: /content/dataset/training_set/cats/cat.3738.jpg
inflating: /content/dataset/training_set/cats/cat.3739.jpg
inflating: /content/dataset/training_set/cats/cat.374.jpg
inflating: /content/dataset/training_set/cats/cat.3740.jpg
inflating: /content/dataset/training_set/cats/cat.3741.jpg
inflating: /content/dataset/training_set/cats/cat.3742.jpg
inflating: /content/dataset/training_set/cats/cat.3743.jpg
inflating: /content/dataset/training_set/cats/cat.3744.jpg
inflating: /content/dataset/training_set/cats/cat.3745.jpg
inflating: /content/dataset/training_set/cats/cat.3746.jpg
inflating: /content/dataset/training_set/cats/cat.3747.jpg
inflating: /content/dataset/training_set/cats/cat.3748.jpg
inflating: /content/dataset/training_set/cats/cat.3749.jpg
inflating: /content/dataset/training_set/cats/cat.375.jpg

```
import os
import shutil

TESTDIR = "/content/dataset/test_set"
TRAININGDIR = "/content/dataset/training_set"

cats_dir = os.path.join(TESTDIR, 'cats')
dogs_dir = os.path.join(TESTDIR, 'dogs')

def move_files_to_parent(src_dir, parent_dir):
    for filename in os.listdir(src_dir):
        src_file = os.path.join(src_dir, filename)
        dest_file = os.path.join(parent_dir, filename)
        if os.path.isfile(src_file):
            shutil.move(src_file, dest_file)

move_files_to_parent(cats_dir, TESTDIR)
move_files_to_parent(dogs_dir, TESTDIR)

os.rmdir(cats_dir)
os.rmdir(dogs_dir)

print("Arquivos movidos e pastas removidas com sucesso!")

cats_dir = os.path.join(TRAININGDIR, 'cats')
dogs_dir = os.path.join(TRAININGDIR, 'dogs')

move_files_to_parent(cats_dir, TRAININGDIR)
move_files_to_parent(dogs_dir, TRAININGDIR)

os.rmdir(cats_dir)
os.rmdir(dogs_dir)
```



Arquivos movidos e pastas removidas com sucesso!

```
import numpy as np # scientific computing
import kagglehub
```

```
import tensorflow as tf # numerical computation using data flow graphs
from tensorflow import keras # deep learning
from keras_preprocessing.image import ImageDataGenerator, load_img # generate batches of tensor image
import pandas as pd # structured data
import matplotlib.pyplot as plt # scientific plotting library
import random # pseudo-random number generators
import os # for accessing directory structure

d4rklucif3r_cat_and_dogs_path = kagglehub.dataset_download('d4rklucif3r/cat-and-dogs')
```

📄 Downloading from [https://www.kaggle.com/api/v1/datasets/download/d4rklucif3r/cat-and-dogs?dataset=100%|██████████| 218M/218M \[00:05<00:00, 39.1MB/s\]](https://www.kaggle.com/api/v1/datasets/download/d4rklucif3r/cat-and-dogs?dataset=100%|██████████| 218M/218M [00:05<00:00, 39.1MB/s])Extracting files...

▼ Inicializações

```
IMAGE_WIDTH = 128 # largura da imagem
IMAGE_HEIGHT = 128 # altura da imagem
IMAGE_SIZE = (IMAGE_WIDTH, IMAGE_HEIGHT) # tamanho da imagem
IMAGE_CHANNELS = 3 # número de canais RGB da imagem
DIR_TRAIN = "/content/dataset/training_set/" # Diretório contendo dados de treinamento
DIR_TEST = "/content/dataset/test_set/" # Diretório contendo dados de teste
```

▼ Pré-Processamento dos Dados de Treino e de Teste

Fonte: <https://www.kaggle.com/d4rklucif3r/cat-and-dogs>

▼ Treino

```
# Read the images from the training directory and generates a pandas structure with the name of each image
filenames = os.listdir(DIR_TRAIN)
categories = []
for filename in filenames:
    categories.append(filename.split('.')[0]) # category = initial part of the file name

df_train = pd.DataFrame({
    'filename': filenames,
    'category': categories
})
```

```
df_train.head()
```

📄

| | filename | category |
|---|--------------|----------|
| 0 | dog.1073.jpg | dog |
| 1 | cat.752.jpg | cat |
| 2 | dog.3412.jpg | dog |
| 3 | cat.3465.jpg | cat |
| 4 | cat.1910.jpg | cat |

```
df_train.tail()
```

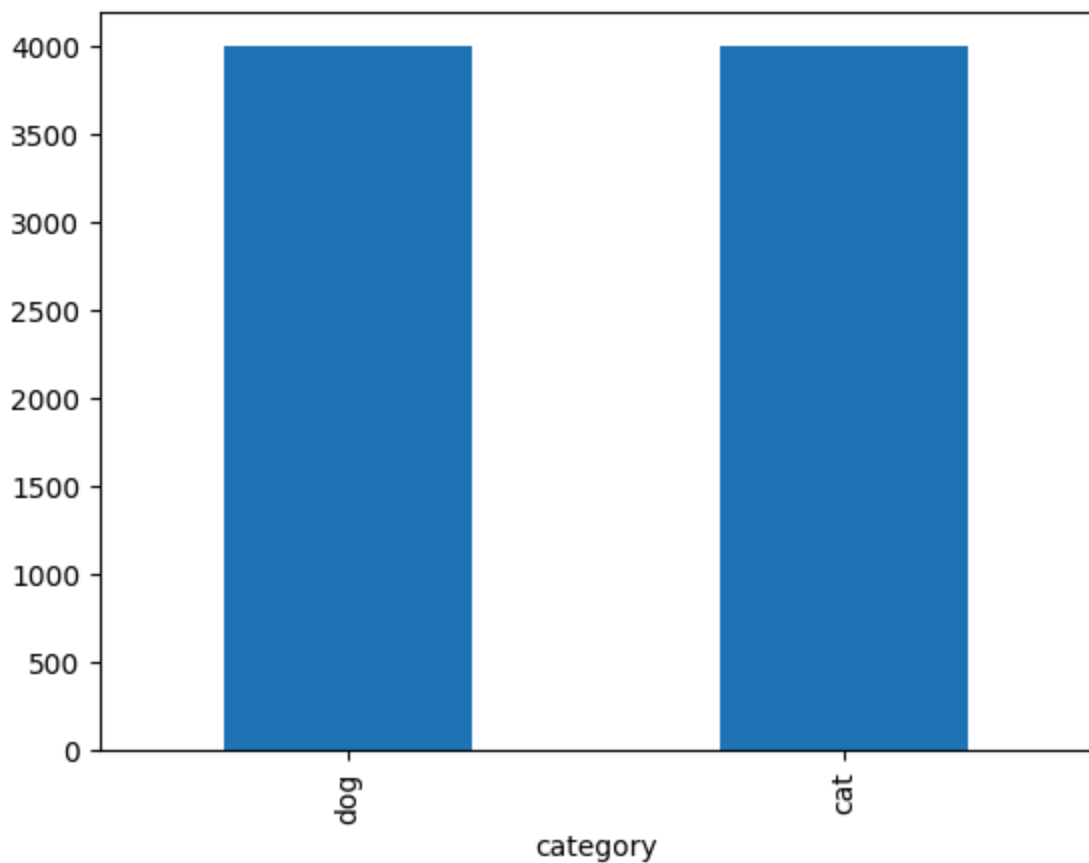


| | filename | category |
|-------------|--------------|----------|
| 7995 | dog.3284.jpg | dog |
| 7996 | dog.42.jpg | dog |
| 7997 | cat.2153.jpg | cat |
| 7998 | dog.2536.jpg | dog |
| 7999 | dog.2590.jpg | dog |

```
# number of cats and dogs
df_train['category'].value_counts().plot.bar()
```



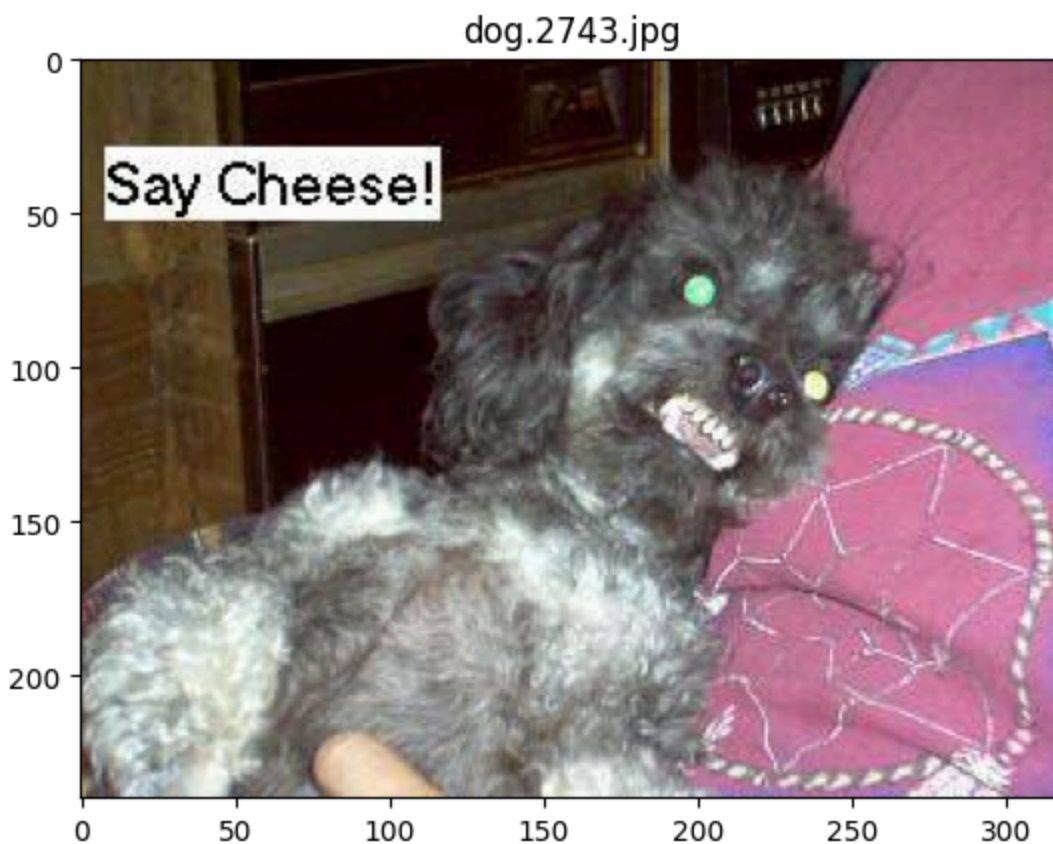
<Axes: xlabel='category'>



Exibe uma amostra

```
# Randomly chooses a file and displays its image
sample = random.choice(filenamees)
image = load_img(DIR_TRAIN+sample)
plt.imshow(image)
plt.title(sample)
```

↔ Text(0.5, 1.0, 'dog.2743.jpg')



```
# summarize some details about the image
print(image.format)
print(image.mode)
print(image.size)
```

↔ JPEG
RGB
(319, 240)

▼ Representação de imagens RGB

Em imagens coloridas, o padrão RGB (Red, Green, Blue) representa cada pixel como um vetor de três números para os três canais das cores primárias vermelho, verde e azul. Cada número varia de 0 a 255, representando a intensidade da cor correspondente. Os três valores combinados formam a cor do pixel. Por exemplo, a cor violeta pode ser representada como 128, 0, 128, uma mistura de intensidade moderada de vermelho e azul, sem o verde.

Já em imagens em escala de cinza (*grayscale*), cada pixel é representado com um único número (entre 0 e 255), o qual determina o quão preto é o pixel (0 é preto e 255 é branco brilhante).

```
# convert image to numpy array
data = np.asarray(image)
# summarize shape
print(data.shape)
```

↔ (240, 319, 3)

▼ Teste

```
# Read the images from the test directory and generates a pandas structure with the name of each file
filenames = os.listdir(DIR_TEST)
categories = []
for filename in filenames:
    categories.append(filename.split('.')[0]) # category = file name

df_test = pd.DataFrame({
    'filename': filenames,
    'category': categories
})
```

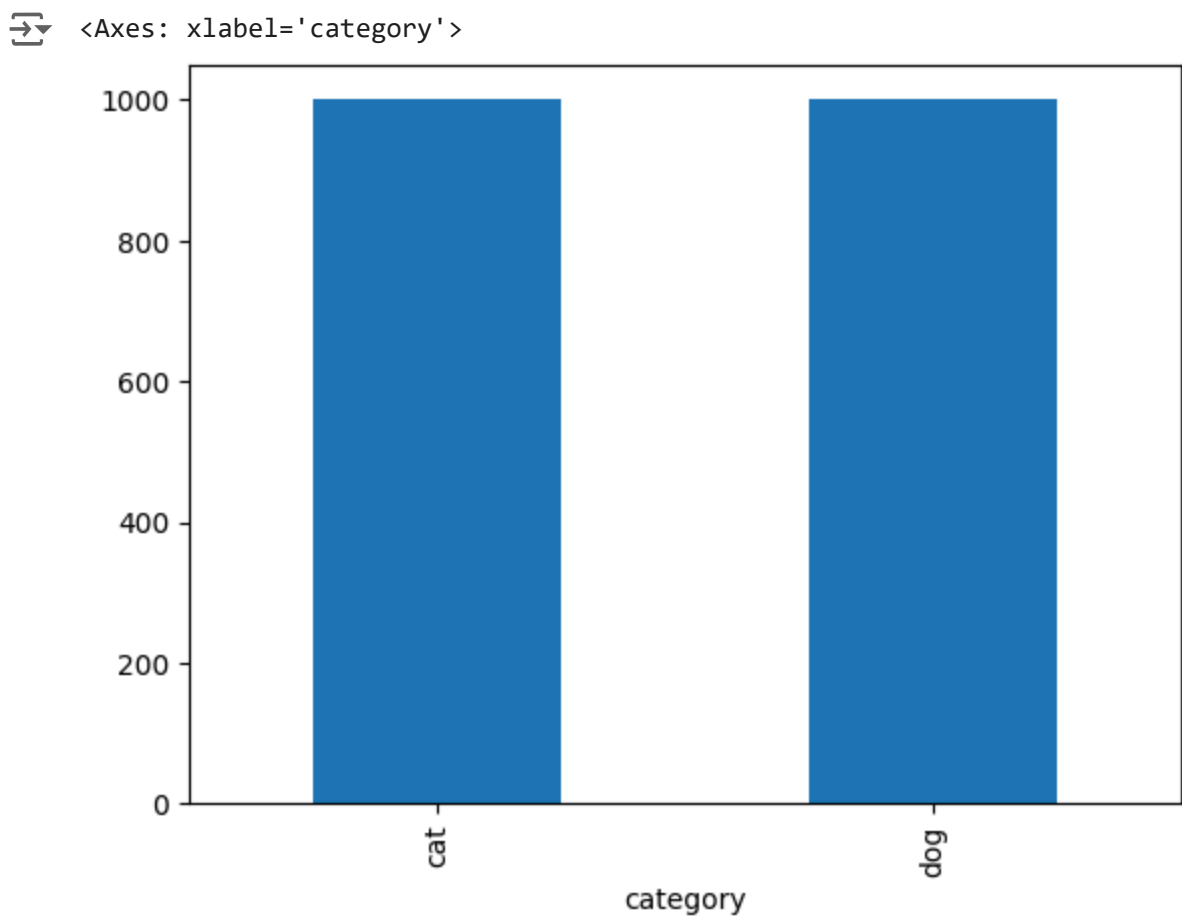
```
df_test.head()
```

| | filename | category |
|---|--------------|----------|
| 0 | cat.4697.jpg | cat |
| 1 | dog.4135.jpg | dog |
| 2 | cat.4244.jpg | cat |
| 3 | dog.4641.jpg | dog |
| 4 | dog.4213.jpg | dog |

```
df_test.tail()
```

| | filename | category |
|------|--------------|----------|
| 1995 | cat.4039.jpg | cat |
| 1996 | cat.4355.jpg | cat |
| 1997 | dog.4655.jpg | dog |
| 1998 | cat.4139.jpg | cat |
| 1999 | cat.4362.jpg | cat |

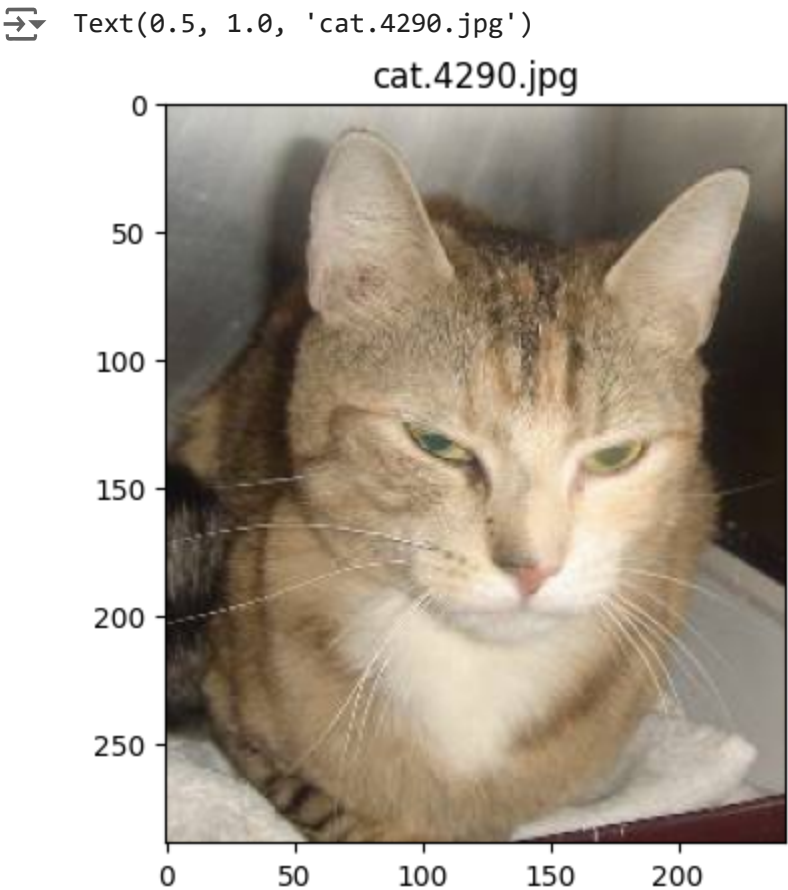
```
# number of cats and dogs
df_test['category'].value_counts().plot.bar()
```



Exibe uma amostra

Escolha um arquivo aleatório do conjunto de teste e exiba sua imagem

```
# Randomly chooses a file and displays its image
### INICIE O CÓDIGO AQUI ### (4 linhas de código)
sample = random.choice(filenamees)
image = load_img(DIR_TEST+sample)
plt.imshow(image)
plt.title(sample)
### TERMINE O CÓDIGO AQUI ###
```



Data Augmentation

Data Augmentation (aumento de dados) é uma técnica usada para expandir artificialmente o conjunto de dados de treinamento. Ela ajuda a evitar o problema de *overfitting*, que faz com que o modelo de treinamento fique muito ajustado aos dados usados para o treinamento, e não gera um bom desempenho em outros conjuntos de dados.

No caso das imagens de gatos e cachorros, a ideia é alterar os dados de treino com pequenas transformações nas imagens, de forma a reproduzir as variações que ocorrem quando alguém tira uma foto. Por exemplo, centralização, diferentes escalas (gatos/cachorros grandes e pequenos), a imagem está rotacionada etc. Além disso, o dataset contém imagens de tamanhos diferentes, e precisam ser padronizadas para dar entrada na rede neural.

As abordagens para *data augmentation* alteram os dados de treino de forma a mudar a representação vetorial, mas mantendo o mesmo rótulo de classe. Algumas transformações populares são: grayscales, horizontal flips, vertical flips, random crops, color jitters, translations, rotations, shift, brightness, and zoom..

Fazendo algumas dessas transformações nos dados de treino, é possível multiplicar o número de exemplos de treino, de forma a criar um modelo mais robusto.

Tutorial para configurar *data augmentation* de imagens em Keras: <https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/>

▼ Gerador de Treino

Documentação:
https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

Este exemplo foi adaptado de <https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification>.

```
batch_size = 15
train_datagen = ImageDataGenerator(
    rotation_range = 15,
    rescale = 1./255,
    shear_range = 0.1,
    zoom_range = 0.2,
    horizontal_flip = True,
    width_shift_range = 0.1,
    height_shift_range = 0.1
)

# Takes the dataframe and the path to a directory and generates batches.
# The generated batches contain augmented/normalized data.
train_generator = train_datagen.flow_from_dataframe(
    df_train,
    DIR_TRAIN,
    x_col = 'filename',
    y_col = 'category',
    target_size = IMAGE_SIZE,
    class_mode = 'categorical',
    batch_size = batch_size
)
```

➡ Found 8000 validated image filenames belonging to 2 classes.

▼ Gerador de Teste

Um gerador de dados pode também ser usado para especificar o dataset de validação e o dataset de teste. Normalmente, uma instância separada de ImageDataGenerator é criada para que se possa configurar a mesma escala de pixels que aquela usada na instância ImageDataGenerator do dataset de treinamento, mas ela não usará *data augmentation*. Isso ocorre porque *data augmentation* é somente usada como uma técnica para expandir artificialmente o dataset de treinamento, para se melhorar o desempenho de um dataset menor.

```
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_dataframe(
    df_test,
    DIR_TEST,
    x_col = 'filename',
    y_col = 'category',
    target_size = IMAGE_SIZE,
    class_mode = 'categorical',
    batch_size = batch_size
)
```

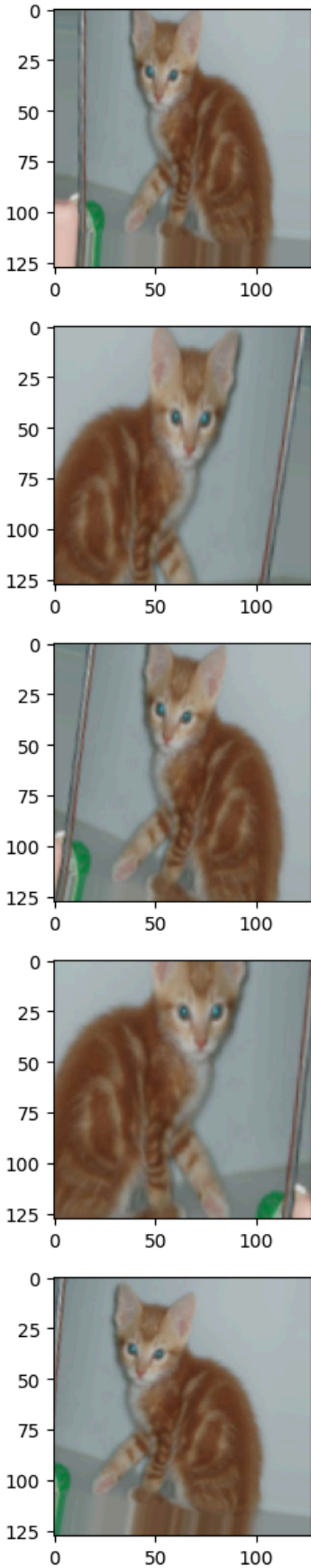
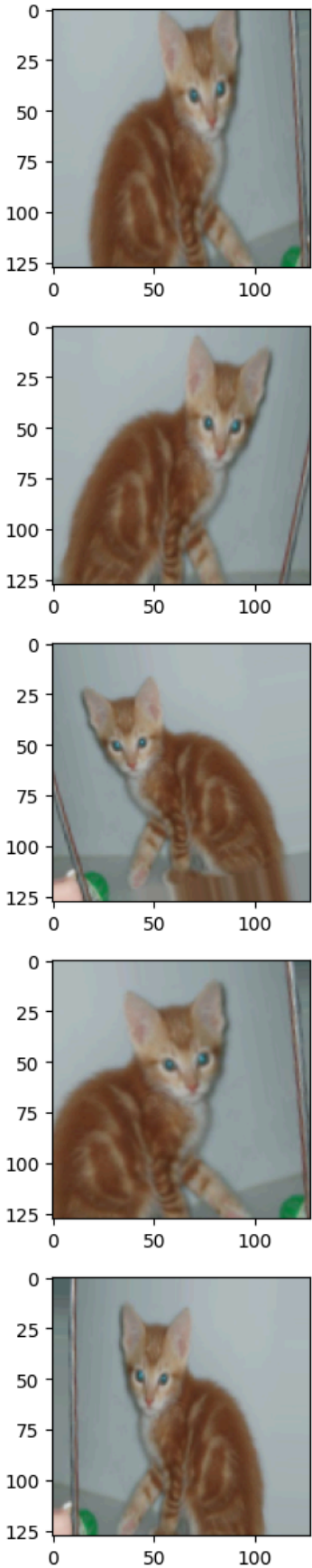
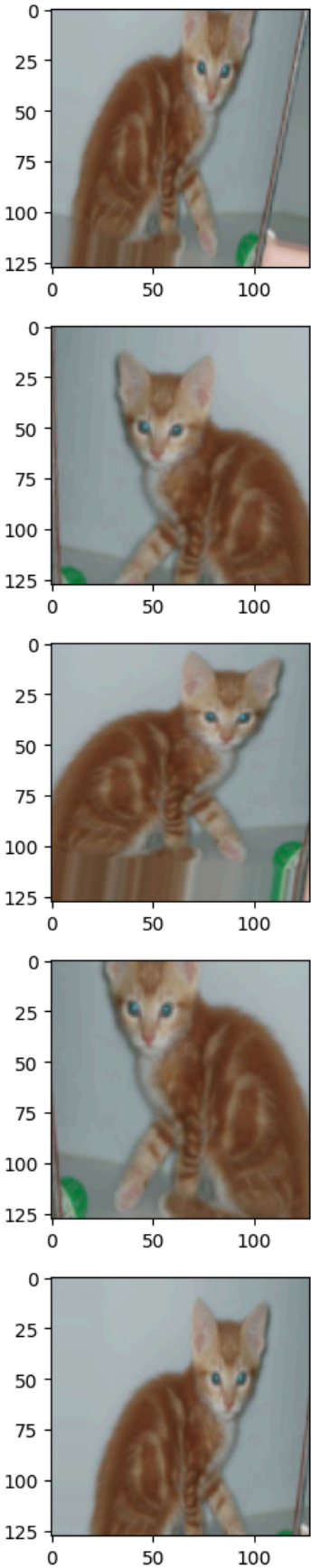
➡ Found 2000 validated image filenames belonging to 2 classes.

▼ Exemplo das Imagens Aumentadas

```
df_example = df_train.sample(n=1).reset_index(drop=True)
example_generator = train_datagen.flow_from_dataframe(
    df_example,
    DIR_TRAIN,
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='categorical'
)
```

➡ Found 1 validated image filenames belonging to 1 classes.

```
plt.figure(figsize=(12,12))
for i in range(0, 15):
    plt.subplot(5, 3, i+1)
    for X_batch, Y_batch in example_generator:
        image = X_batch[0]
        plt.imshow(image)
        break
plt.tight_layout()
plt.show()
```



▾ Definição do Modelo

Crie um modelo em Keras com a seguinte configuração:

- Camada de entrada: no formato dos dados de entrada do problema

- Camada *Flatten*: camada para achatar o vetor de entrada. No padrão RGB, cada imagem é representada como um vetor de 3 dimensões. Entretanto, cada entrada em uma rede neural é composta por um vetor de apenas uma dimensão. Assim, é necessário "achatar" o vetor da imagem para uma dimensão. Isso é feito pela camada *Flatten*, que cria um vetor de uma dimensão concatenando sequencialmente os valores nos vetores dos três canais da imagem. Desta forma, se o vetor imagem tem o formato (128, 128, 3), por exemplo, o vetor achatado terá dimensão (49.152, 1), o resultado de 128 x 128 x 3.
- Camada 1: 32 neurônios, função de ativação *ReLU*
- Camada 2: 64 neurônios, função de ativação *ReLU*
- Camada 3: 32 neurônios, função de ativação *ReLU*
- Camada 4 (saída): 2 neurônios, função de ativação *Softmax*. Embora tenha apenas duas classes, o problema deve ser modelado como uma classificação multiclasse ("cat" ou "dog"). Assim, a camada de saída deve ter duas unidades e usar a função Softmax.

Documentação:

Keras: <https://keras.io/>

Flatten layer: https://keras.io/api/layers/reshaping_layers/flatten/

Classe *Model*: <https://keras.io/api/models/model/>

Funções de ativação disponíveis: <https://keras.io/api/layers/activations/>

```
### INICIE O CÓDIGO AQUI ### (7 linhas de código)
inputs = keras.Input(shape=(128,128,3))
x = keras.layers.Flatten()(inputs)
x = keras.layers.Dense(units=32, activation="relu")(x)
x = keras.layers.Dense(units=64, activation="relu")(x)
x = keras.layers.Dense(units=32, activation="relu")(x)
outputs = keras.layers.Dense(units=2, activation="softmax")(x)
model = keras.Model(inputs=inputs, outputs=outputs)
### TERMINE O CÓDIGO AQUI ###
```

```
# Prints a summary of the network, showing its architecture and parameters.
model.summary()
```

 Model: "functional"

| Layer (type) | Output Shape | Param # |
|--|--|---------------------------|
| input_layer (InputLayer) | (None , 128 , 128 , 3) | 0 |
| flatten (Flatten) | (None , 49152) | 0 |
| dense (Dense) | (None , 32) | 1,572,896 |
| dense_1 (Dense) | (None , 64) | 2,112 |
| dense_2 (Dense) | (None , 32) | 2,080 |
| dense_3 (Dense) | (None , 2) | 66 |

Total params: [1,577,154](#) (6.02 MB)
Trainable params: [1,577,154](#) (6.02 MB)
Non-trainable params: [0](#) (0.00 B)

SAÍDA ESPERADA:

Confira a configuração da rede e o total de parâmetros = 1,577,154

Compilação do Modelo

Compile o model usando os seguintes parâmetros:

- Função de perda: categorical_crossentropy
- Otimizador: adam
- Métricas: accuracy

Dica 1: use a função *compile*: https://keras.io/api/models/model_training_apis/

Dica 2: relação de funções de perda: https://www.tensorflow.org/api_docs/python/tf/keras/losses

Dica 3: relação de otimizadores: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers

Dica 4: relação de métricas: <https://keras.io/api/metrics/>

```
### INICIE O CÓDIGO AQUI ### (1 linha de código)
model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
### TERMINE O CÓDIGO AQUI ###
```

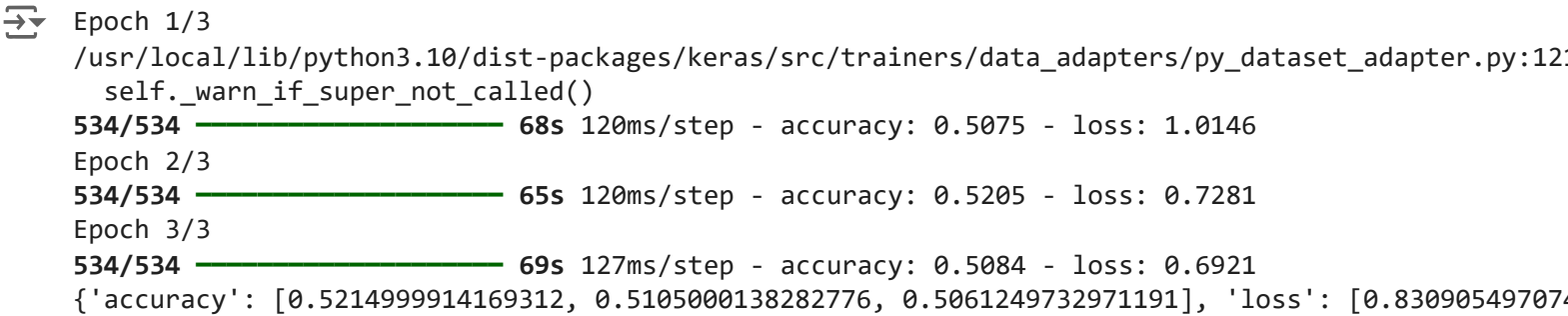
✓ Treinamento do Modelo

Ajusta o modelo aos dados de treinamento. Devem ser fornecidos os dados de treinamento e o número de épocas (iterações). Uma época é composta por uma única passagem por todos os exemplos do conjunto de treino.

Efetue o treinamento do modelo usando 3 épocas com os dados aumentados do gerador de treino *train_generator*.

Dica: use a função *fit*: https://keras.io/api/models/model_training_apis/

```
### INICIE O CÓDIGO AQUI ### (1 linha de código)
history = model.fit(train_generator, epochs = 3)
### TERMINE O CÓDIGO AQUI ###
print(history.history) # print per-epoch timeseries of metrics values
```



```
Epoch 1/3
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:12:
self._warn_if_super_not_called()
534/534 ————— 68s 120ms/step - accuracy: 0.5075 - loss: 1.0146
Epoch 2/3
534/534 ————— 65s 120ms/step - accuracy: 0.5205 - loss: 0.7281
Epoch 3/3
534/534 ————— 69s 127ms/step - accuracy: 0.5084 - loss: 0.6921
{'accuracy': [0.5214999914169312, 0.5105000138282776, 0.5061249732971191], 'loss': [0.83090549707,
```

SAÍDA ESPERADA:

Na época 3, tem-se o seguinte resultado (aproximado):

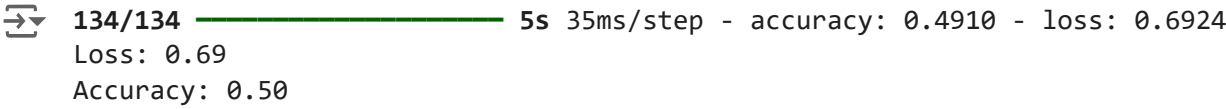
```
{'loss': [0.8931004405021667, 0.6976273059844971, 0.6935341954231262], 'accuracy':
[0.5130000114440918, 0.5195000171661377, 0.5037500262260437]}
```

✓ Avaliação do Modelo

Avalie o desempenho da rede no conjunto de teste usando o gerador de dados aumentados *test_generator*.

Dica: use a função *evaluate*: https://keras.io/api/models/model_training_apis/

```
### INICIE O CÓDIGO AQUI ### (1 linha de código)
loss, acc = model.evaluate(test_generator, steps=len(test_generator))
### TERMINE O CÓDIGO AQUI ###
print("Loss: %.2f" % loss, "\nAccuracy: %.2f" % acc)
```



```
134/134 ————— 5s 35ms/step - accuracy: 0.4910 - loss: 0.6924
Loss: 0.69
Accuracy: 0.50
```

SAÍDA ESPERADA:

Valores aproximados:

Accuracy: 0.50


Apresente a predição do conjunto de teste usando o gerador de dados aumentados *test_generator*.

Função *predict*: https://keras.io/api/models/model_training_apis/

Função *argmax*: <https://numpy.org/doc/stable/reference/generated/numpy.argmax.html>

Na classificação multiclasse, `model.predict` retorna um vetor de probabilidades para cada classe. A função `argmax` retorna o índice do vetor com a maior probabilidade, neste caso, 0 para gato e 1 para cachorro.

```
### INICIE O CÓDIGO AQUI ### (1 linha de código)
predictions = model.predict(test_generator)
### TERMINE O CÓDIGO AQUI ###
preds = [np.argmax(x, axis=-1) for x in predictions]
print("Predictions: ", ["cat" if x == 0 else "dog" for x in preds[:30]]) # print the first 30 predictions
print("\nCorrect:      ", [x for x in df_test['category'][:30]])
```

 **134/134** ————— **4s 29ms/step**

Predictions: ['dog', 'dog', 'dog', 'dog', 'dog', 'dog', 'dog', 'dog', 'dog', 'dog', 'dog', 'dog']

Correct: ['cat', 'dog', 'cat', 'dog', 'dog', 'cat', 'cat', 'cat', 'dog', 'dog', 'dog', 'dog']