DEEPLEARNING

TENSORFLOW

https://www.tensorflow.org

Rafael Vieira Coelho

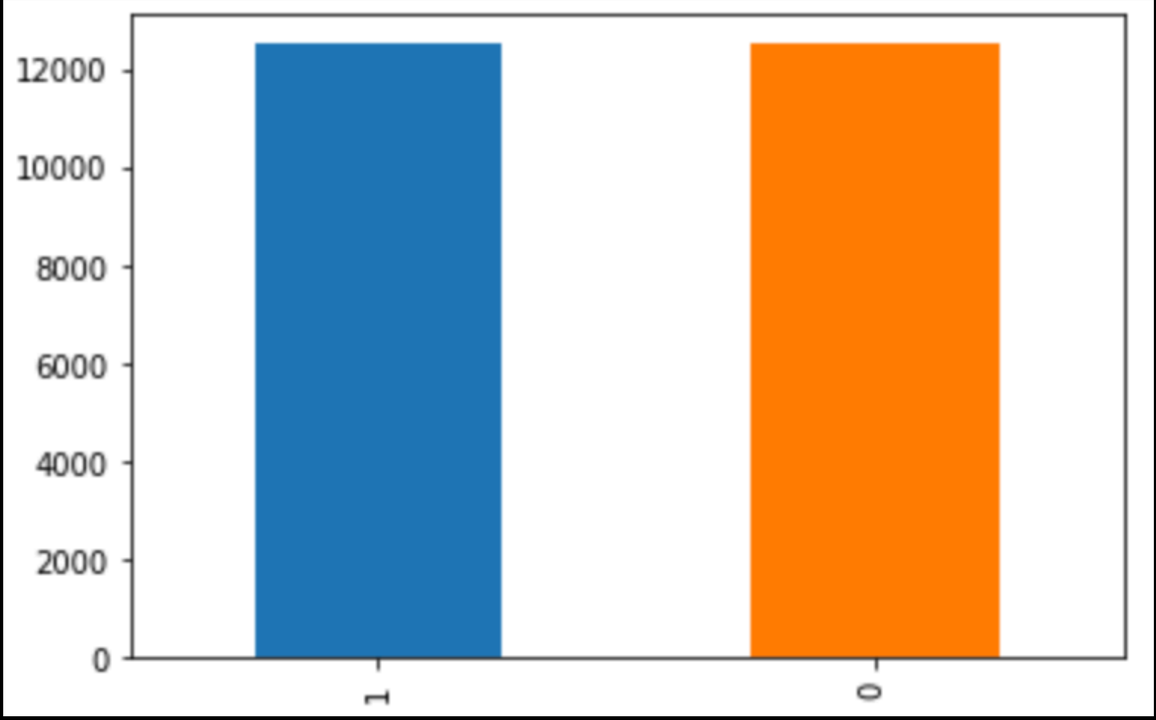
TENSORFLOW

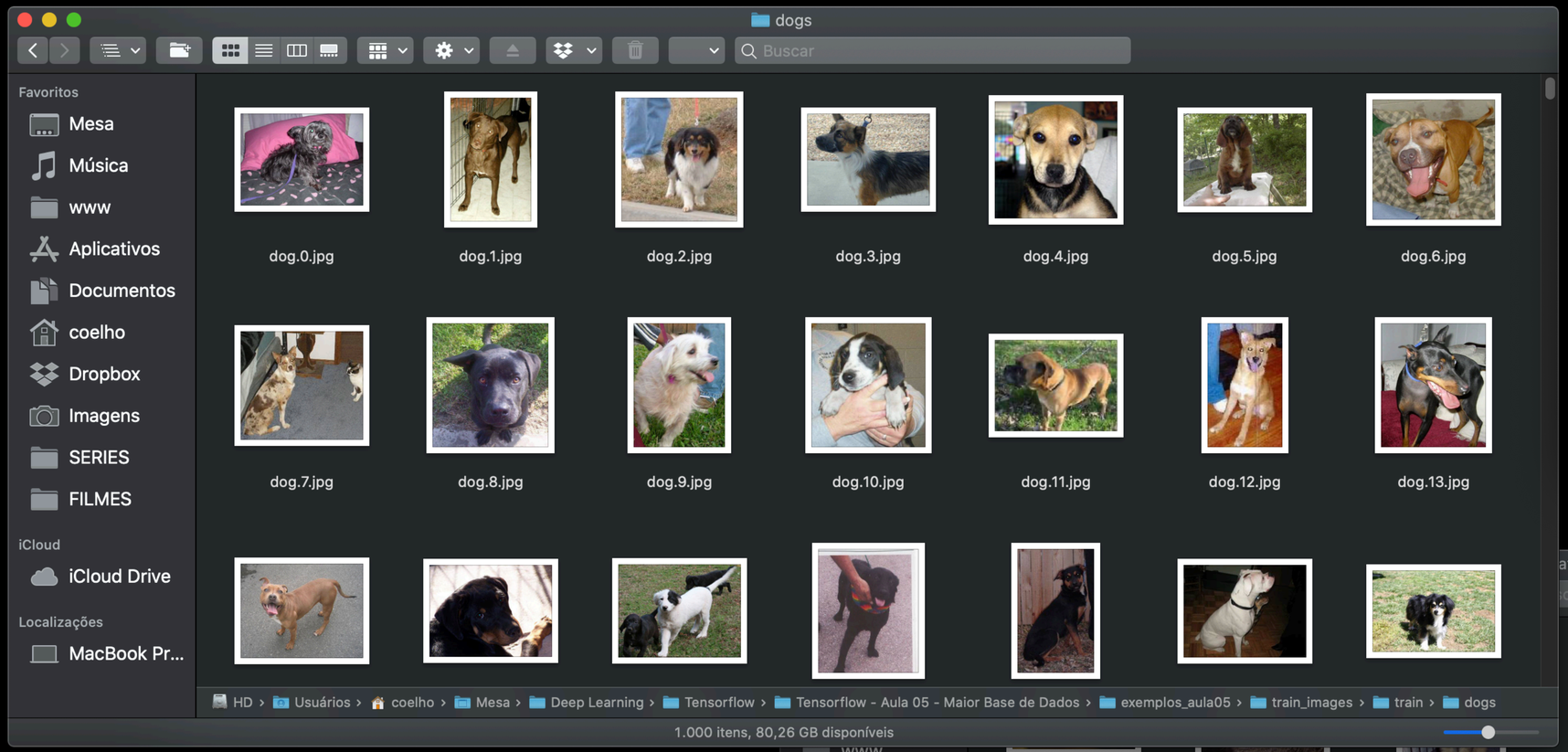
USANDO UMA BASE DE DADOS GRANDE

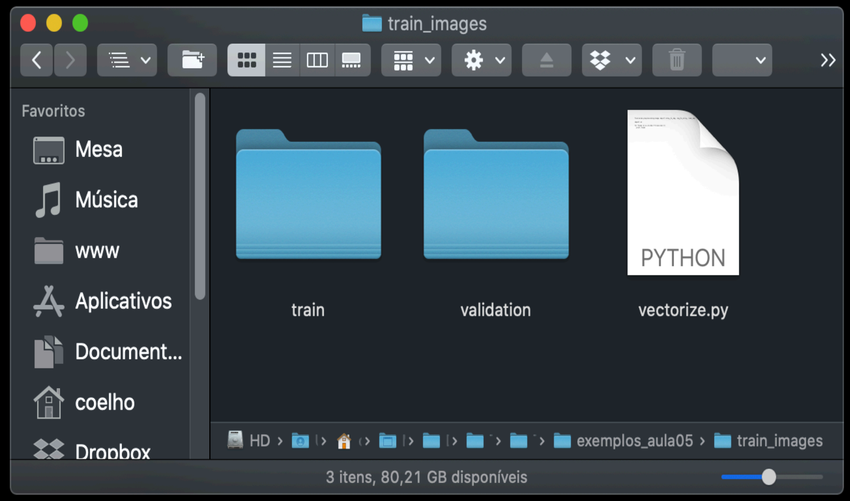
0) DOWNLOAD DA BASE DE DADOS

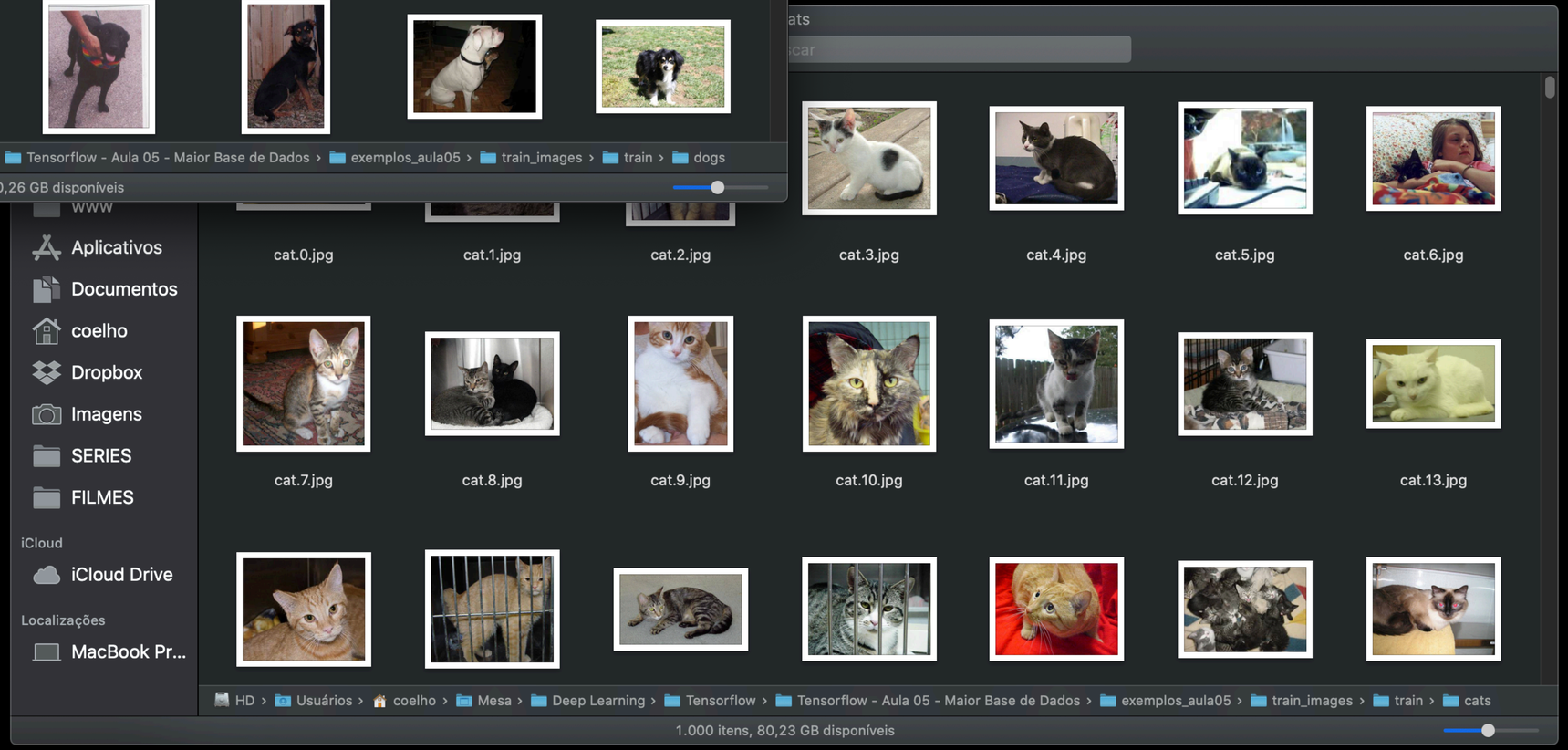
- https://storage.googleapis.com/mledu-datasets/cats_and_dogs_filtered.zip

- A base de dados contém 25.000 imagens, sendo metade de cachorros (label 1) e metade de gatos (label 0).









Temos duas pastas:
uma para treino
e outra para validação.

1) IMPORTANDO A BASE DE DADOS

ex1_loading_dataset.py

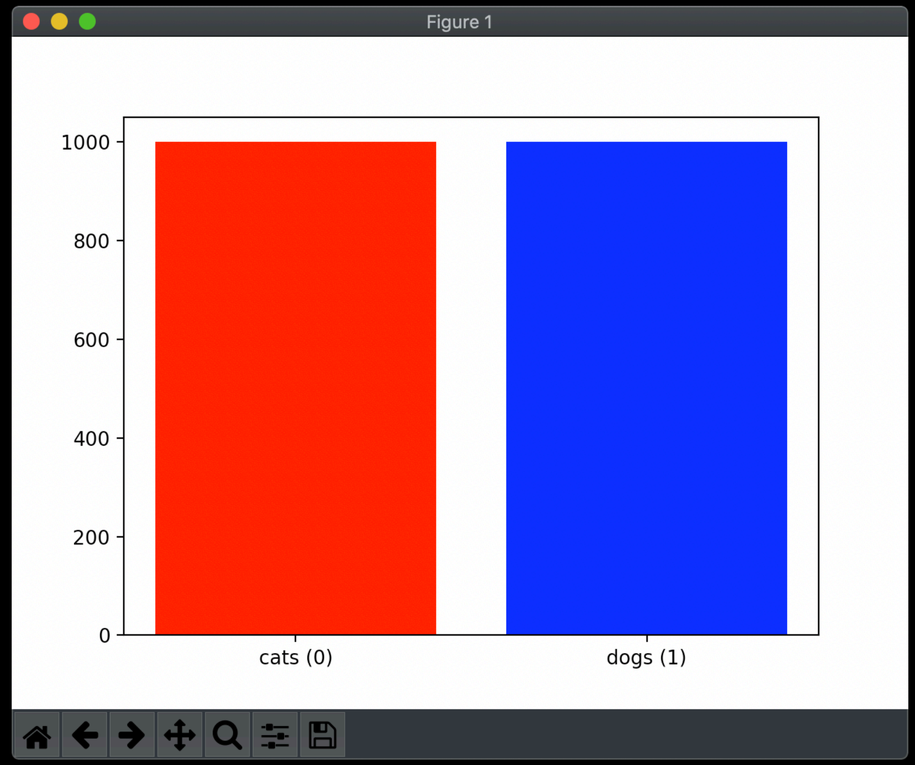
```python
def main():
    # 1 - loading data
    base_dir = './train_images/'
    train_dir = os.path.join(base_dir, 'train')
    validation_dir = os.path.join(base_dir, 'validation')
    # Directory with our training cat/dog pictures
    train_cats_dir = os.path.join(train_dir, 'cats')
    train_dogs_dir = os.path.join(train_dir, 'dogs')
    # Directory with our validation cat/dog pictures
    validation_cats_dir = os.path.join(validation_dir, 'cats')
    validation_dogs_dir = os.path.join(validation_dir, 'dogs')
    train_cat_fnames = os.listdir(train_cats_dir)
    train_dog_fnames = os.listdir(train_dogs_dir)
    print(train_cat_fnames[:10])
    print(train_dog_fnames[:10])
    print('total training cat images :', len(os.listdir(train_cats_dir)))
    print('total training dog images :', len(os.listdir(train_dogs_dir)))
    print('total validation cat images :', len(os.listdir(validation_cats_dir)))
    print('total validation dog images :', len(os.listdir(validation_dogs_dir)))
```

```
['cat.952.jpg', 'cat.946.jpg', 'cat.6.jpg', '
['dog.775.jpg', 'dog.761.jpg', 'dog.991.jpg',
total training cat images : 1000
total training dog images : 1000
total validation cat images : 500
total validation dog images : 500
```

1) IMPORTANDO A BASE DE DADOS

ex1_loading_dataset.py

```
38      # show the images quantity
39      categories = []
40      filenames = []
41      load_images(train_dogs_dir, categories, filenames)
42      load_images(train_cats_dir, categories, filenames)
43  #    load_images(validation_cats_dir, categories, filenames)
44  #    load_images(validation_dogs_dir, categories, filenames)
45      df = pd.DataFrame({
46          'filename': filenames,
47          'category': categories
48      })
49      print(df.head())
50      plt.bar(['cats (0)'], df['category'].value_counts()[0], color='r')
51      plt.bar(['dogs (1)'], df['category'].value_counts()[1], color='b')
52      plt.show()
```



1) IMPORTANDO A BASE DE DADOS

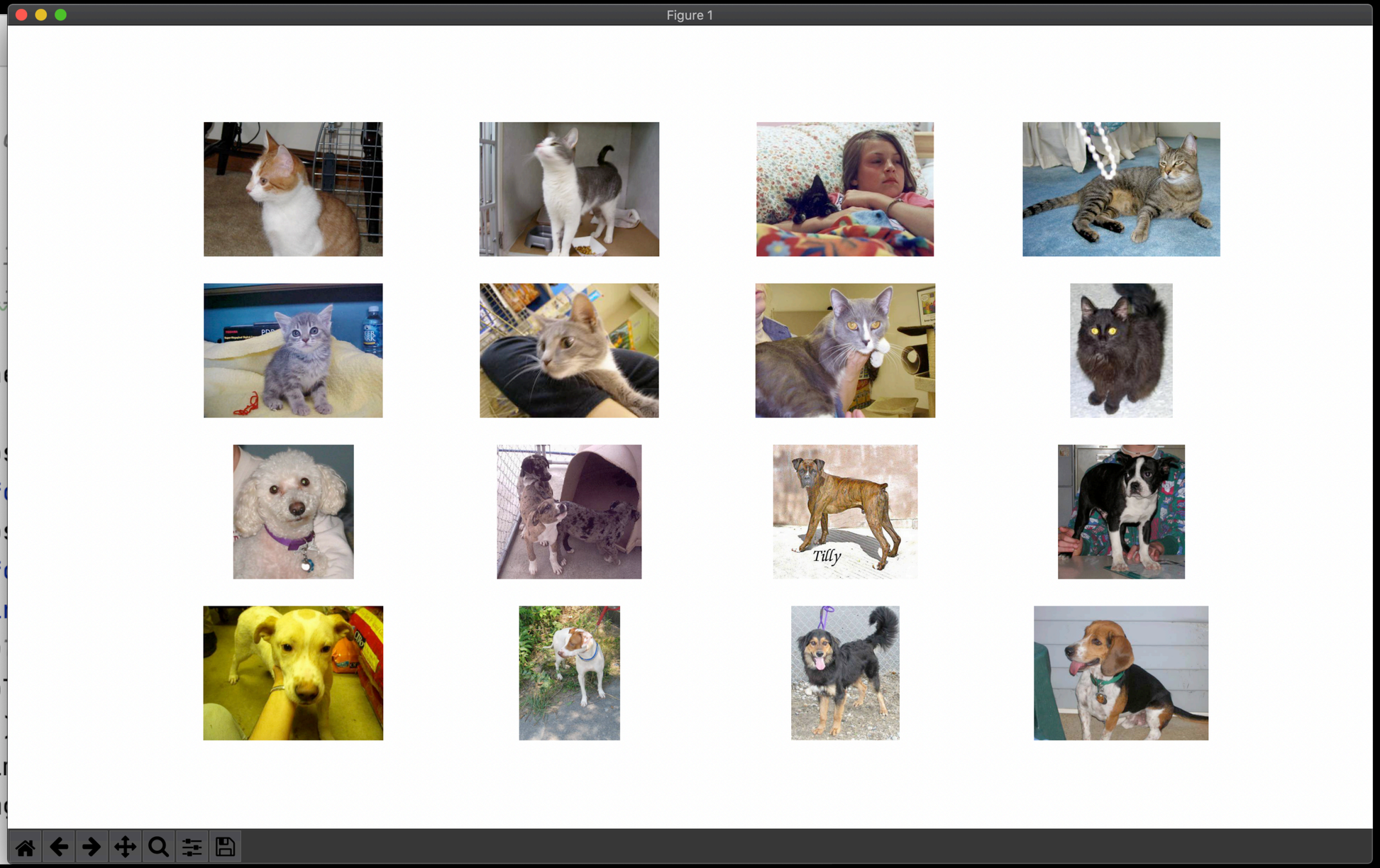
ex1_loading_dataset.py

```
    # show images
    # Parameters for our graph; we'll output images in a 4x4 configuration
    nrows = 4
    ncols = 4
    pic_index = 0  # Index for iterating over images
    # Set up matplotlib fig, and size it to fit 4x4 pics
    fig = plt.gcf()
    fig.set_size_inches(ncols * 4, nrows * 4)
    pic_index += 8
    next_cat_pix = [os.path.join(train_cats_dir, fname)
                    for fname in train_cat_fnames[pic_index - 8:pic_index]]
    next_dog_pix = [os.path.join(train_dogs_dir, fname)
                    for fname in train_dog_fnames[pic_index - 8:pic_index]]
    for i, img_path in enumerate(next_cat_pix + next_dog_pix):
        # Set up subplot; subplot indices start at 1
        sp = plt.subplot(nrows, ncols, i + 1)
        sp.axis('Off')  # Don't show axes (or gridlines)
        img = mpimg.imread(img_path)
        plt.imshow(img)
    plt.show()
```

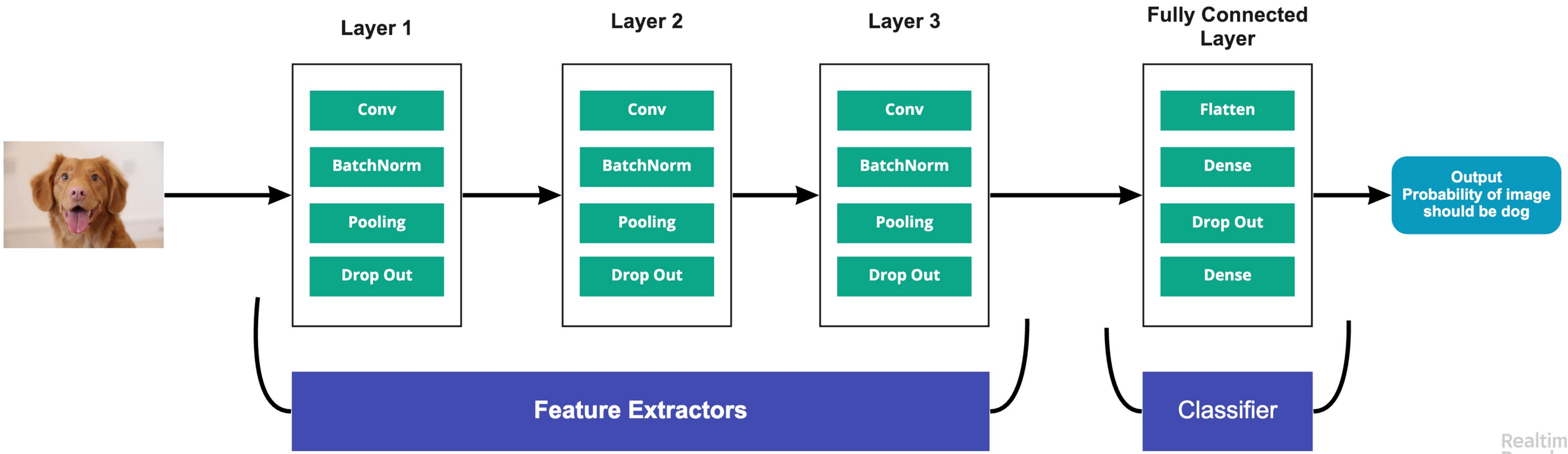
1) IMPORTANDO A BASE DE DADOS

ex1_loading_dataset.py

```
55      # show images
56      # Parameters for
57      nrows = 4
58      ncols = 4
59      pic_index = 0  #
60      # Set up matplotli
61      fig = plt.gcf()
62      fig.set_size_inch
63      pic_index += 8
64      next_cat_pix = [os
65                      fo
66      next_dog_pix = [os
67                      fo
68      for i, img_path in
69          # Set up subp
70          sp = plt.subp
71          sp.axis('Off')
72          img = mpimg.i
73          plt.imshow(im
74      plt.show()
```



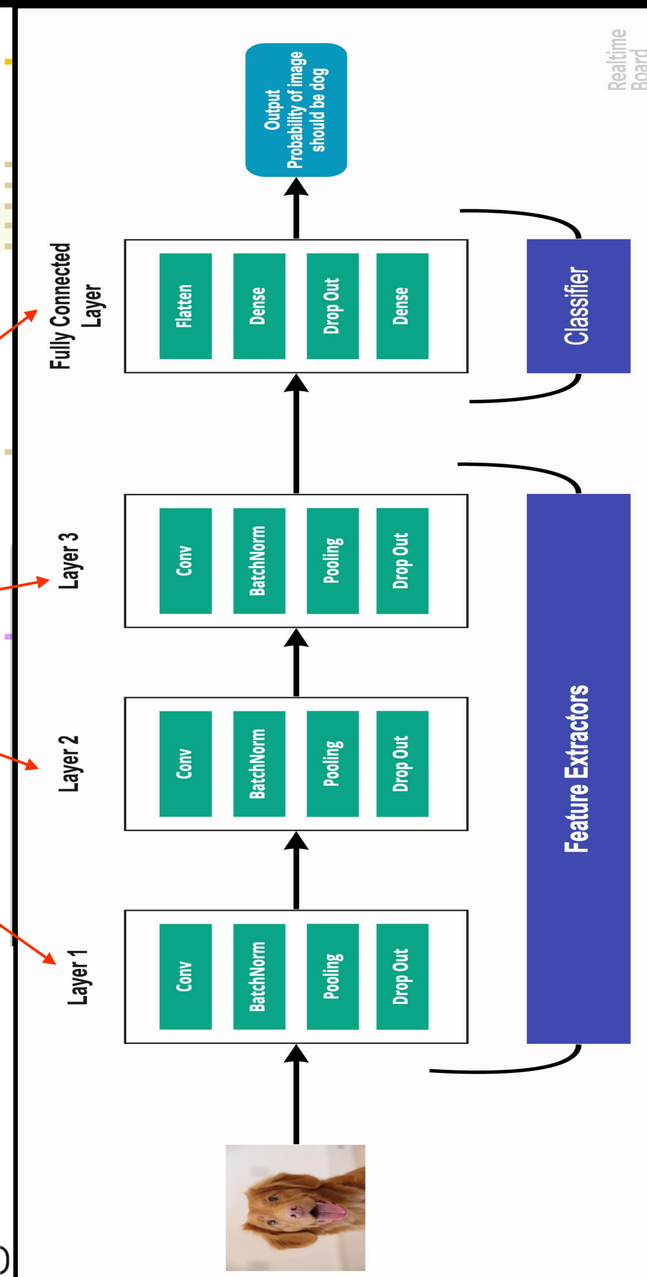
2) DEFININDO O MODELO DA REDE NEURAL



- **Layer 1(Input):** representa os dados de imagem de entrada. Reformata cada imagem para vetor unidimensional. Ex: A imagem tem 64 x 64 = 4096, e é convertida em um vetor (4096,1).
- **Layer 2 (Convulational):** Extrai características (features) da imagem.
- **Layer 3 (Pooling):** Reduz o volume espacial da imagem após a convulação.
- **Fully Connected Layer:** Faz a classificação.
- **Output Layer:** Apresenta como saída a probabilidade de ser a imagem de um cachorro.

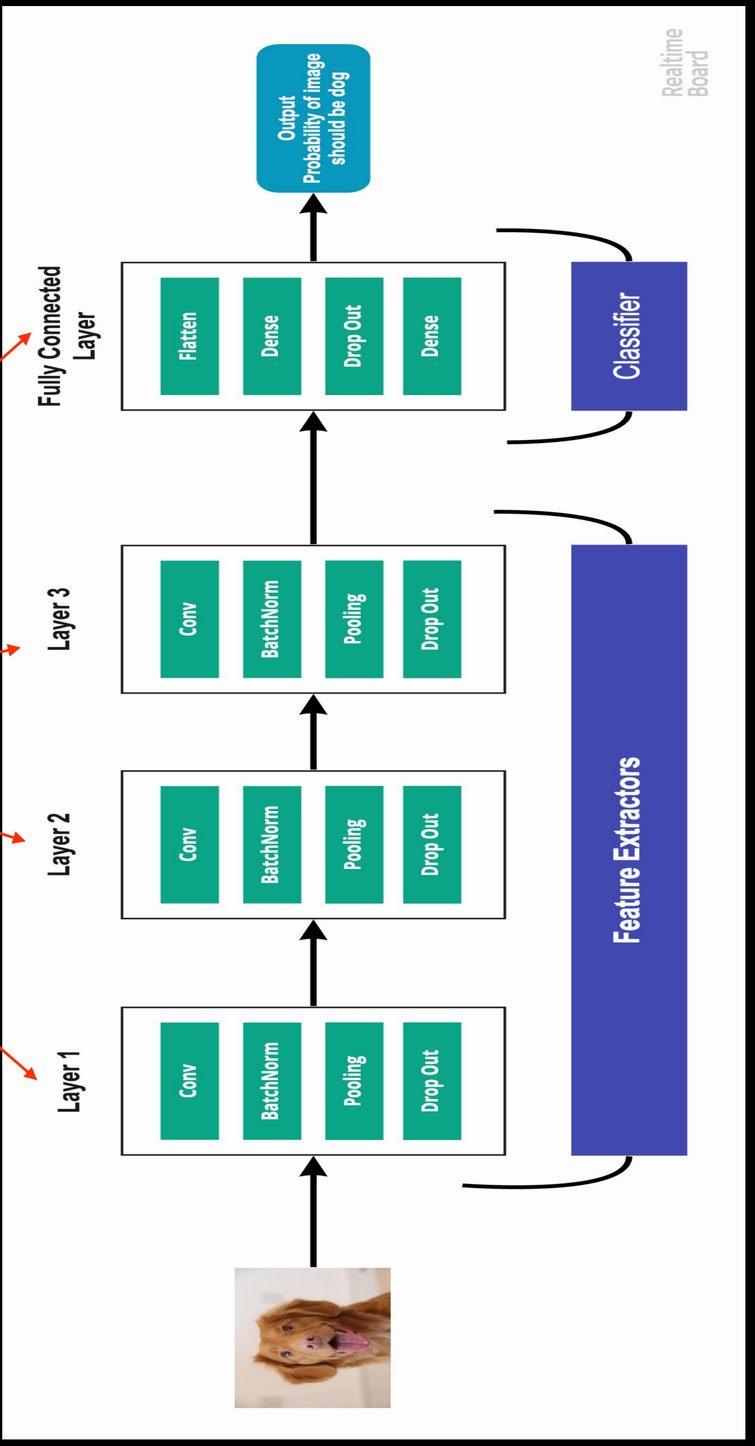
2) DEFININDO O MODELO DA REDE NEURAL

```
28      # define the model
29      model = Sequential()
30
31      model.add(Conv2D(32, (3, 3), activation='relu',
32                       input_shape=(IMAGE_WIDTH, IMAGE_HEIGHT, IMAGE_CHANNELS)))
33      model.add(BatchNormalization())
34      model.add(MaxPooling2D(pool_size=(2, 2)))
35      model.add(Dropout(0.25))
36
37      model.add(Conv2D(64, (3, 3), activation='relu'))
38      model.add(BatchNormalization())
39      model.add(MaxPooling2D(pool_size=(2, 2)))
40      model.add(Dropout(0.25))
41
42      model.add(Conv2D(128, (3, 3), activation='relu'))
43      model.add(BatchNormalization())
44      model.add(MaxPooling2D(pool_size=(2, 2)))
45      model.add(Dropout(0.25))
46
47      model.add(Flatten())
48      model.add(Dense(512, activation='relu'))
49      model.add(BatchNormalization())
50      model.add(Dropout(0.5))
51      model.add(Dense(1, activation='sigmoid'))
52
53      model.compile(loss='binary_crossentropy', optimizer=RMSprop(lr=0.001), metrics=['accuracy'])
```



SUMÁRIO DO MODELO

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
batch_normalization_1 (Batch (None, 148, 148, 32)      128
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 72, 72, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 72, 72, 64)        256
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 36, 36, 64)        0
_________________________________________________________________
dropout_2 (Dropout)          (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 34, 34, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 34, 34, 128)       512
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 17, 17, 128)       0
_________________________________________________________________
dropout_3 (Dropout)          (None, 17, 17, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 36992)             0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               18940416
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 19,037,121
Trainable params: 19,035,649
Non-trainable params: 1,472
```



3) PREPARANDO OS DADOS

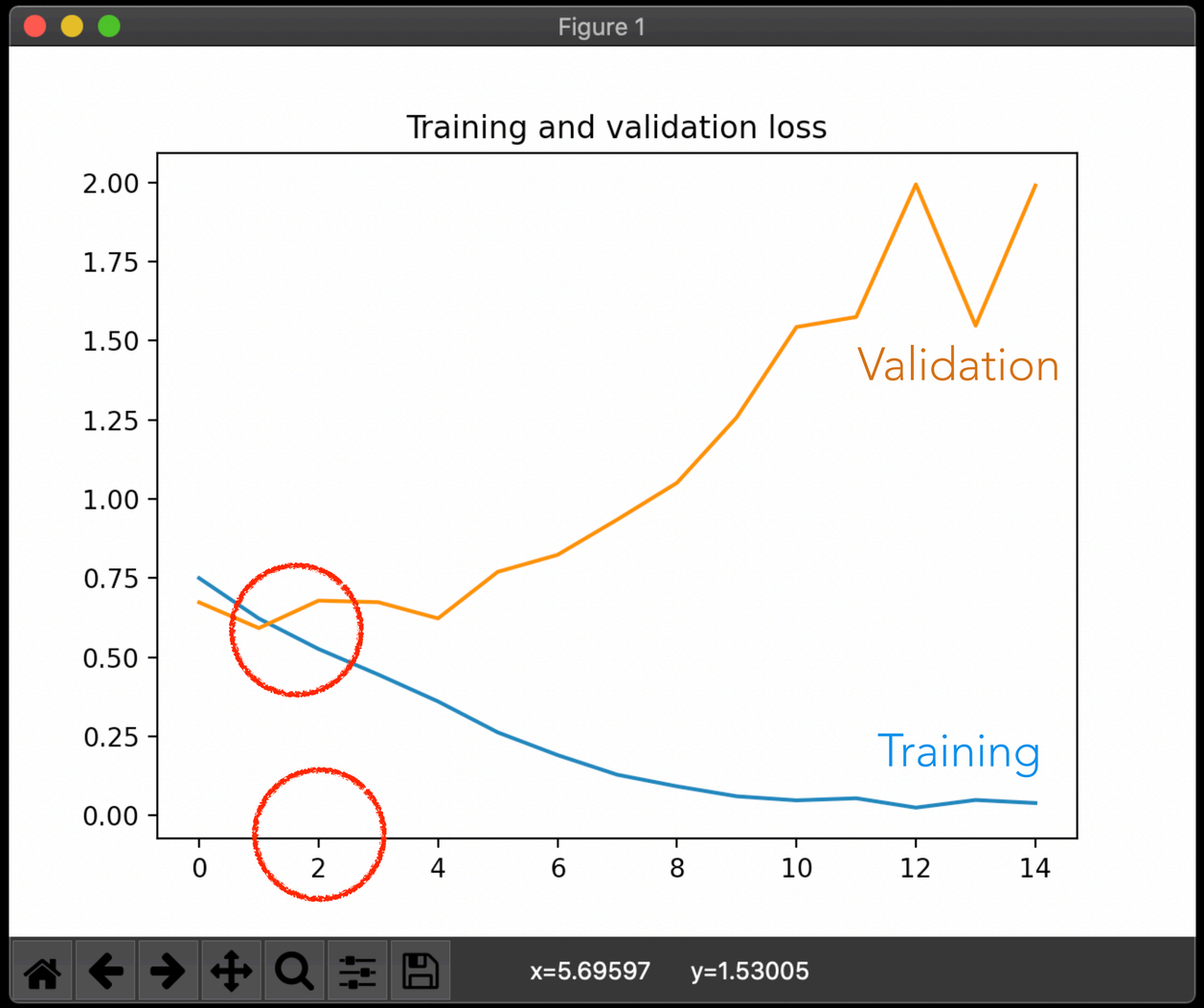
ex3_prepare_data.py

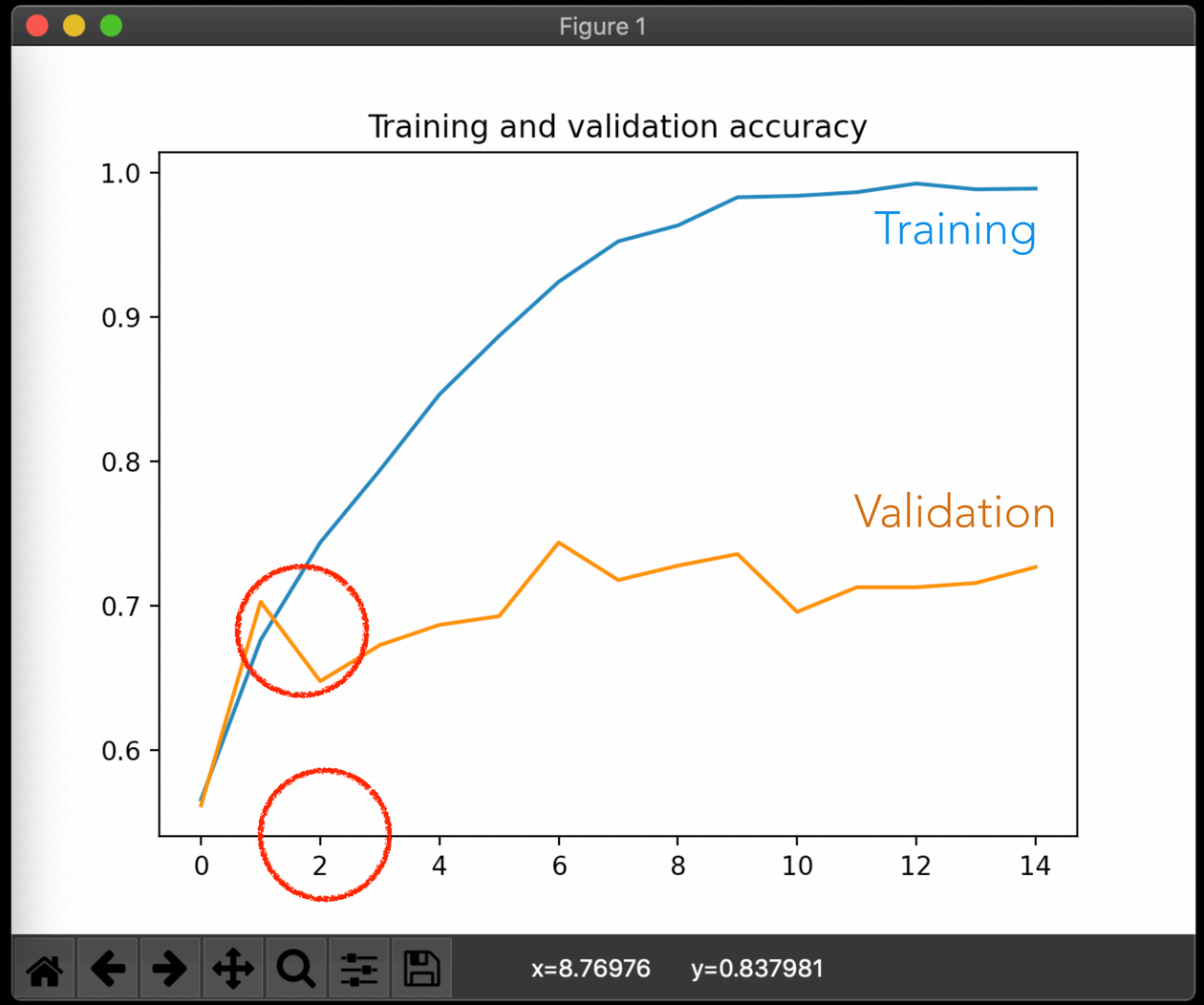
```
70  # 3 - preparing data (preprocessing)
71  # All images will be rescaled by 1./255.
72  train_datagen = ImageDataGenerator(rescale=1.0 / 255.)
73  test_datagen = ImageDataGenerator(rescale=1.0 / 255.)
74  # --------------------
75  # Flow training images in batches of 20 using train_datagen generator
76  # --------------------
77  train_generator = train_datagen.flow_from_directory(train_dir,
78                                                      batch_size=batch_size,
79                                                      class_mode='binary',
80                                                      target_size=(IMAGE_WIDTH, IMAGE_HEIGHT))
81  # --------------------
82  # Flow validation images in batches of 20 using test_datagen generator
83  # --------------------
84  validation_generator = test_datagen.flow_from_directory(validation_dir,
85                                                          batch_size=batch_size,
86                                                          class_mode='binary',
87                                                          target_size=(IMAGE_WIDTH, IMAGE_HEIGHT))
```

4) TREINANDO A REDE

```
105     # 4 - train the network
106     history = model.fit(train_generator,
107                         validation_data=validation_generator,
108                         steps_per_epoch=100,
109                         epochs=15,
110                         validation_steps=50,
111                         verbose=2)
112     print(history)
113     # ------------------------------------------------------------
114     # Retrieve a list of list results on training and test data
115     # sets for each training epoch
116     # ------------------------------------------------------------
117     acc = history.history['accuracy']
118     val_acc = history.history['val_accuracy']
119     loss = history.history['loss']
120     val_loss = history.history['val_loss']
121     epochs = range(len(acc))  # Get number of epochs
122     # ------------------------------------------------
123     # Plot training and validation accuracy per epoch
124     # ------------------------------------------------
125     plt.plot(epochs, acc, color='b')
126     plt.plot(epochs, val_acc, color='r')
127     plt.title('Training and validation accuracy')
128     plt.show()
129     # ------------------------------------------------
130     # Plot training and validation loss per epoch
131     # ------------------------------------------------
132     plt.plot(epochs, loss, color='b')
133     plt.plot(epochs, val_loss, color='r')
134     plt.title('Training and validation loss')
135     plt.show()
```

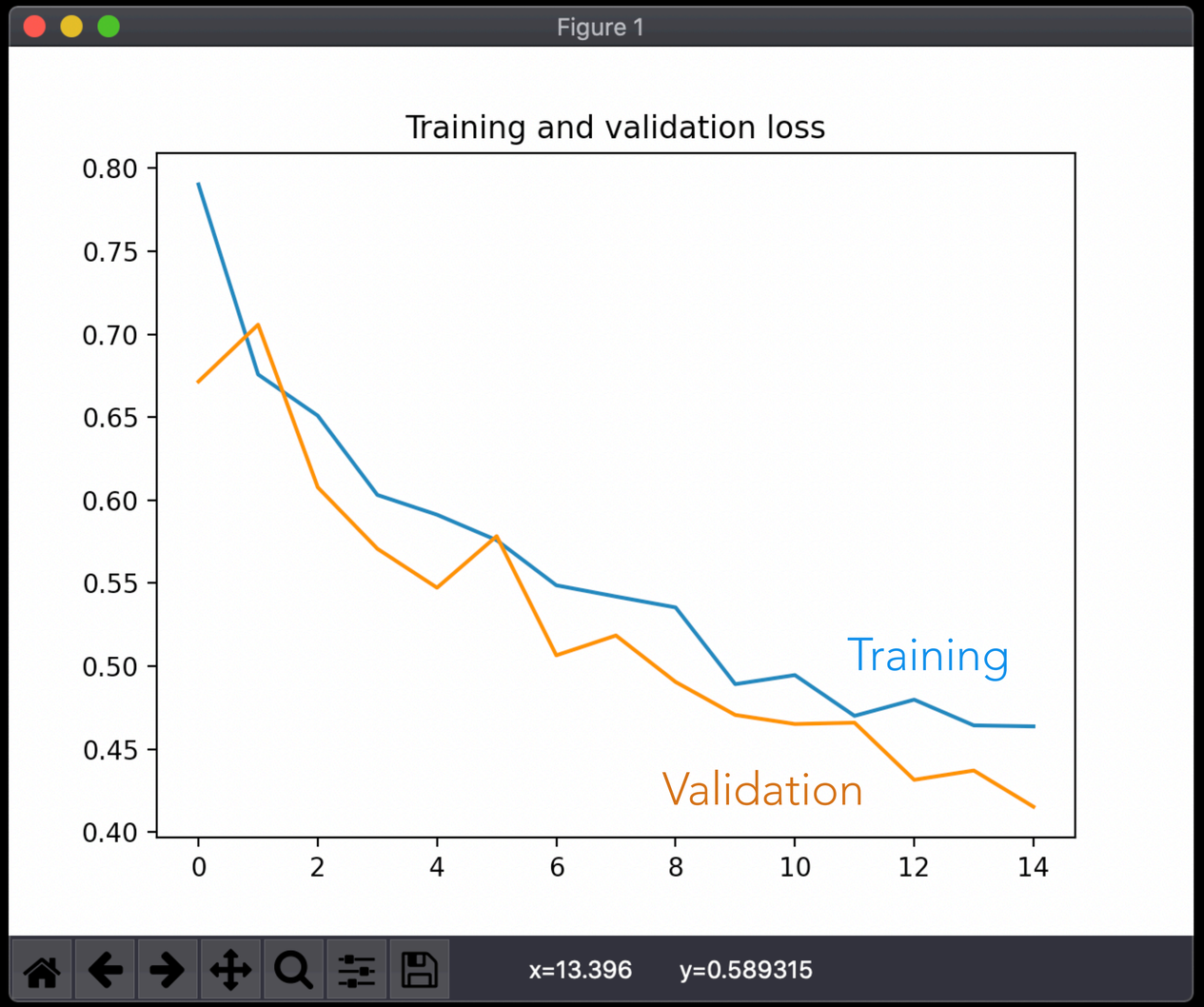
7) RESULTADOS

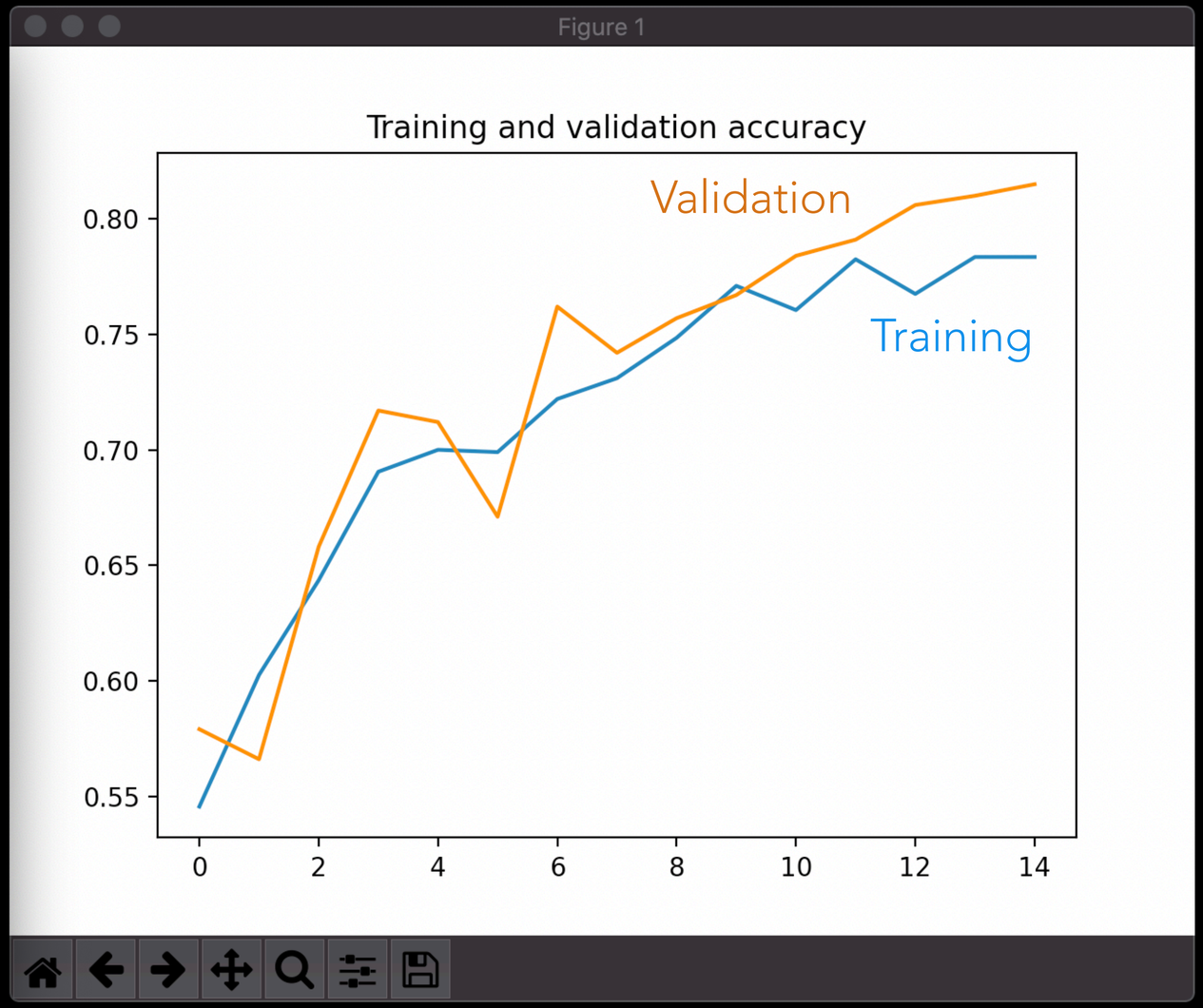




A partir de duas épocas, não aprendemos nada mais (overtraining).

E SE USARMOS UMA BASE DADOS MAIOR?

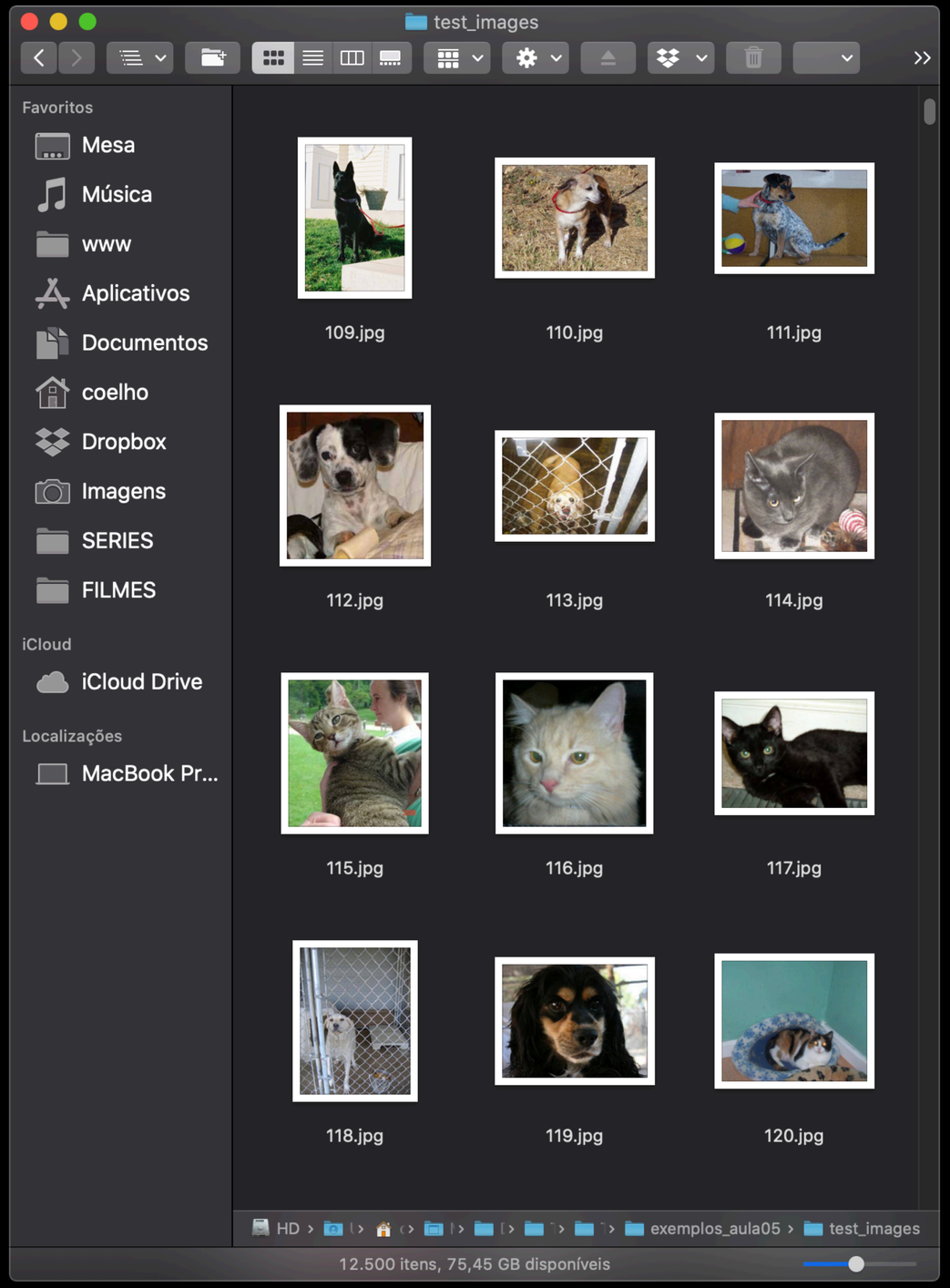




Usando os dados totais do BD de gatos e cachorros do Kagle (25.000 imagens)

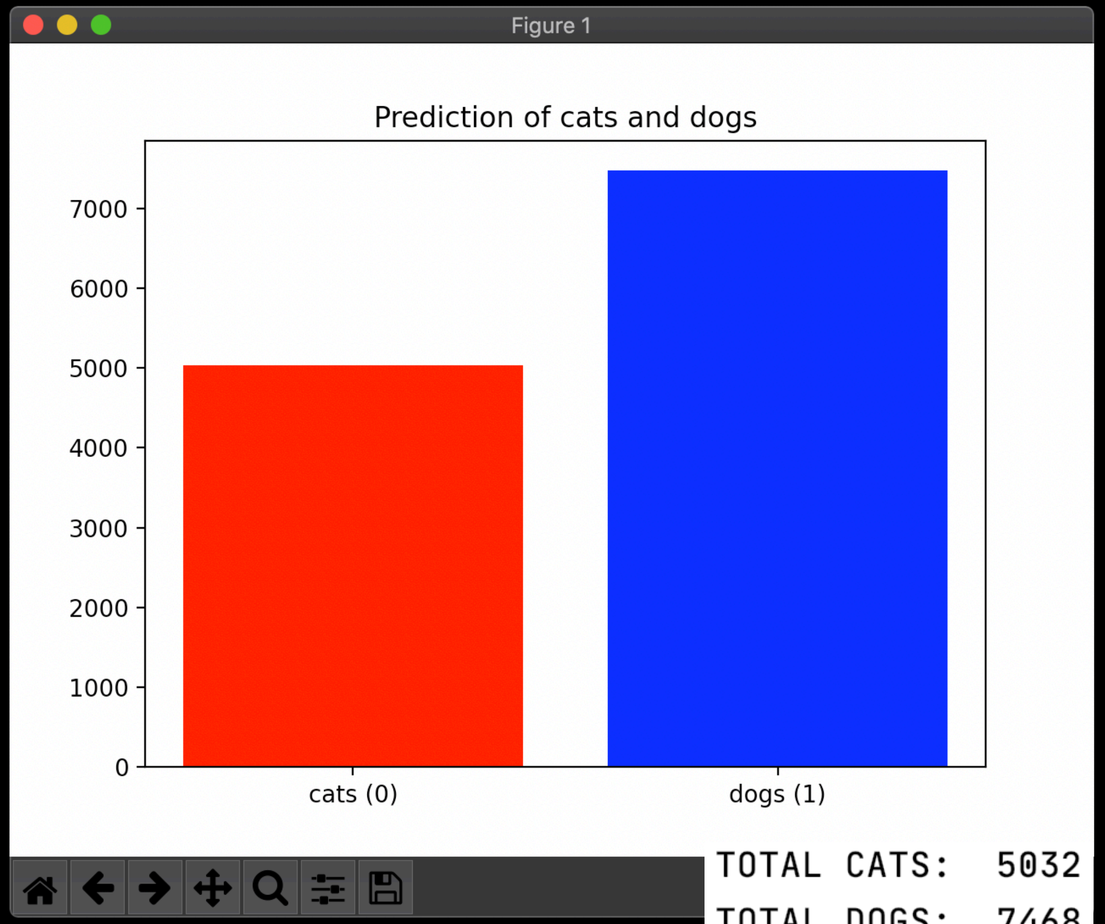
5) PREDIÇÃO

```
    # 5 - predict
    files = [os.path.join(dp, f)
             for dp, dn, filenames in os.walk(test_dir)
             for f in filenames
             if os.path.splitext(f)[1] == '.jpg']
    print(files)
    cats = 0
    dogs = 0
    index = 0
    plt.figure(figsize=(12, 24))
    for fn in files:
        img = load_img(fn, target_size=image_size)
        y = image.img_to_array(img)
        x = np.expand_dims(y, axis=0)
        images = np.vstack([x])
        classes = model.predict(images, batch_size=10)
        print(classes[0])
        category = "dog"
        if classes[0] > 0.5:
            print(fn + " is a dog")
            dogs += 1
        else:
            category = "cat"
            print(fn + " is a cat")
            cats += 1
        plt.subplot(6, 3, index + 1)
        plt.imshow(img)
        plt.xlabel(fn + '(' + "{}".format(category) + ')')
        index += 1
    print('TOTAL CATS: ', cats)
    print('TOTAL DOGS: ', dogs)
    # 6 - plot results
    plt.tight_layout()
    plt.show()  # show previous figure
    plt.title("Prediction of cats and dogs")
    plt.bar(['cats (0)'], cats, color='r')
    plt.bar(['dogs (1)'], dogs, color='b')
    plt.show()
```



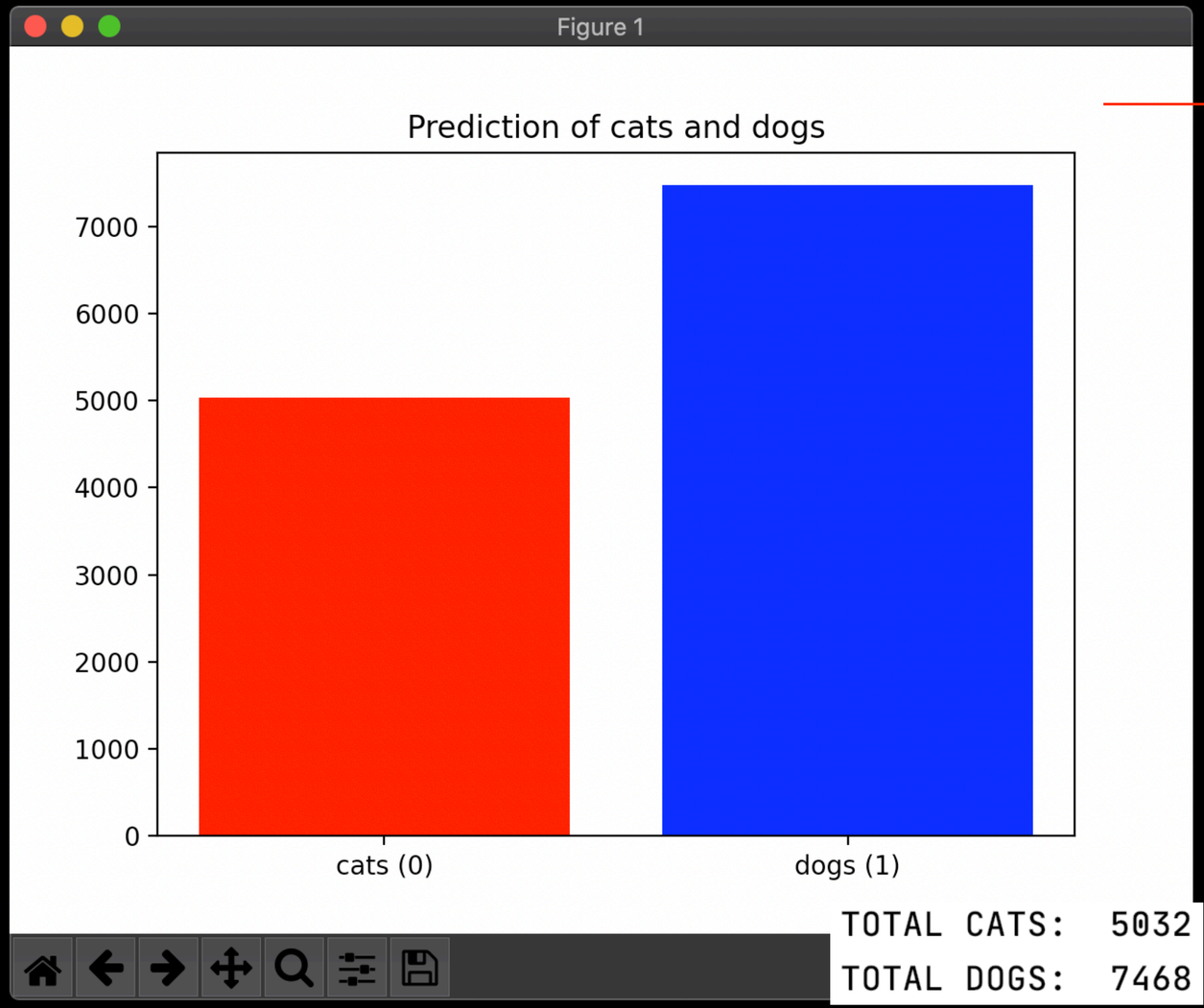
6) RESULTADOS

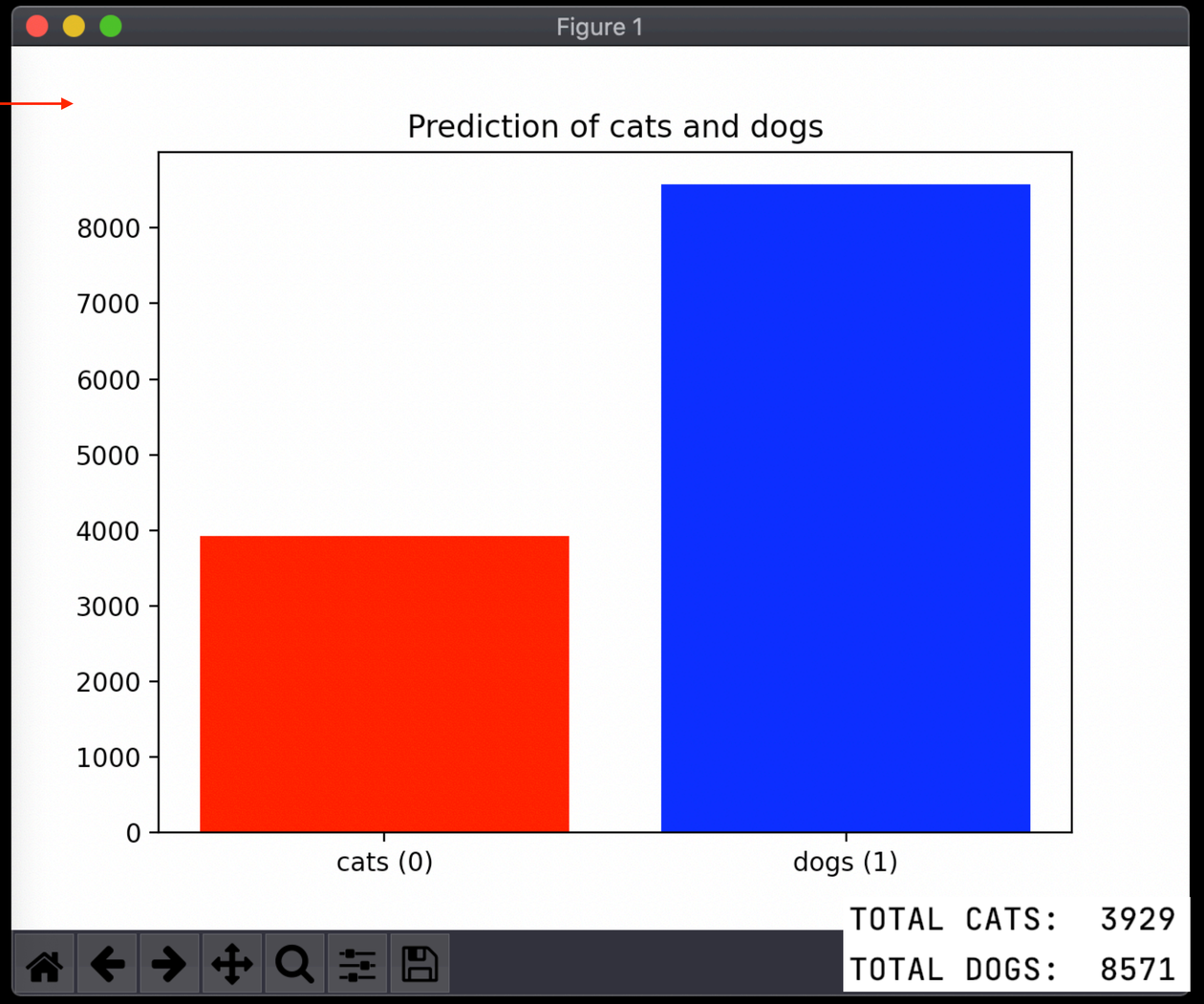
```
    # 5 - predict
    files = [os.path.join(dp, f)
             for dp, dn, filenames in os.walk(test_dir)
             for f in filenames
             if os.path.splitext(f)[1] == '.jpg']
    print(files)
    cats = 0
    dogs = 0
    index = 0
    plt.figure(figsize=(12, 24))
    for fn in files:
        img = load_img(fn, target_size=image_size)
        y = image.img_to_array(img)
        x = np.expand_dims(y, axis=0)
        images = np.vstack([x])
        classes = model.predict(images, batch_size=10)
        print(classes[0])
        category = "dog"
        if classes[0] > 0.5:
            print(fn + " is a dog")
            dogs += 1
        else:
            category = "cat"
            print(fn + " is a cat")
            cats += 1
        plt.subplot(6, 3, index + 1)
        plt.imshow(img)
        plt.xlabel(fn + '(' + "{}".format(category) + ')')
        index += 1
    print('TOTAL CATS: ', cats)
    print('TOTAL DOGS: ', dogs)
    # 6 - plot results
    plt.tight_layout()
    plt.show()  # show previous figure
    plt.title("Prediction of cats and dogs")
    plt.bar(['cats (0)'], cats, color='r')
    plt.bar(['dogs (1)'], dogs, color='b')
    plt.show()
```




```
TOTAL CATS:   5032
TOTAL DOGS:   7468
```

6) RESULTADOS





Usando os dados totais do BD de gatos e cachorros do Kagle (25.000 imagens)

LINKS ÚTEIS

- https://www.kaggle.com/c/dogs-vs-cats
- https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification
- https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/Course%202%20-%20Part%202%20-%20Lesson%202%20-%20Notebook.ipynb#scrollTo=H4XHh2xSfgie

EXERCÍCIO

Tente usar as 25.000 imagens!!

https://ojqugrhayuzyvbkearhjak.coursera-apps.org/notebooks/week1/Exercise_1_Cats_vs_Dogs_Question-FINAL.ipynb

DÚVIDAS?

RAFAELVC2@GMAIL.COM