T2 - Detecção objetos TF

Referências

O código usado nesse trabalho é baseado no Notebook do TensorFlow disponível em Link

Coloque o seu nome aqui:

Nome: Bruno Rodrigues Silva

1 Objetivos

O objetivo desse trabalho é realizar um treinamento de detecção de objetos customizado usando uma RNA pré-treinada.

O conjunto de dados consiste de imagens que mostram dois tipos de objetos: 1) cangurus e 2) pessoas. Assim, tem-se um problema de detecção de objetos com duas classes.

A tarefa principal desse trabalho é preparar os dados para treinamento, ou seja, realizar as anotações das imagens. Para realizar essas anotações você vai usar a função `colab_utils.annotate` que usamos na aula.

Para desenvolver esse trabalho, você vai ter que realizar algumas modificações no progrma usado para essa finalidade visto em aula.

2 Instalação de bilbiotecas e da API de detecção de objetos

2.1 Clonar o repostório de modelos do TensorFlow Hub.

[1]:
```
# Importa bibliotecas do sistema operacional
import os
import pathlib

# Clonagem do repositório de modelos do TensorFlow se isso ainda não foi␣
 ↪realizado
if "models" in pathlib.Path.cwd().parts:
    while "models" in pathlib.Path.cwd().parts:
        os.chdir('..')
elif not pathlib.Path('models').exists():
```

```
!git clone --depth 1 https://github.com/tensorflow/models
```

2.2 Importar bibliotecas

[2]:
```
import matplotlib
import matplotlib.pyplot as plt

import random
import io
import imageio
import glob
import scipy.misc
import numpy as np
from six import BytesIO
from PIL import Image, ImageDraw, ImageFont
from IPython.display import display, Javascript
from IPython.display import Image as IPyImage

import tensorflow as tf

%matplotlib
%matplotlib inline
```

```
Using matplotlib backend: agg
```

2.3 Instalar a API de detecção de objetos

[3]:
```
# Instalar API de detecção de objetos se isso ainda não foi realizado
%%bash
cd models/research/
protoc object_detection/protos/*.proto --python_out=.
cp object_detection/packages/tf2/setup.py .
python -m pip install .
```

```
Processing /content/models/research
Requirement already satisfied: avro-python3 in /usr/local/lib/python3.7/dist-
packages (from object-detection==0.1) (1.10.2)
Requirement already satisfied: apache-beam in /usr/local/lib/python3.7/dist-
packages (from object-detection==0.1) (2.28.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages
(from object-detection==0.1) (7.0.0)
Requirement already satisfied: lxml in /usr/local/lib/python3.7/dist-packages
(from object-detection==0.1) (4.2.6)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-
packages (from object-detection==0.1) (3.2.2)
Requirement already satisfied: Cython in /usr/local/lib/python3.7/dist-packages
(from object-detection==0.1) (0.29.22)
```

```
Requirement already satisfied: contextlib2 in /usr/local/lib/python3.7/dist-
packages (from object-detection==0.1) (0.5.5)
Requirement already satisfied: tf-slim in /usr/local/lib/python3.7/dist-packages
(from object-detection==0.1) (1.1.0)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from object-detection==0.1) (1.15.0)
Requirement already satisfied: pycocotools in /usr/local/lib/python3.7/dist-
packages (from object-detection==0.1) (2.0.2)
Requirement already satisfied: lvis in /usr/local/lib/python3.7/dist-packages
(from object-detection==0.1) (0.5.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
(from object-detection==0.1) (1.4.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages
(from object-detection==0.1) (1.1.5)
Requirement already satisfied: tf-models-official in
/usr/local/lib/python3.7/dist-packages (from object-detection==0.1) (2.4.0)
Requirement already satisfied: pymongo<4.0.0,>=3.8.0 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(3.11.3)
Requirement already satisfied: dill<0.3.2,>=0.3.1.1 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(0.3.1.1)
Requirement already satisfied: mock<3.0.0,>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(2.0.0)
Requirement already satisfied: protobuf<4,>=3.12.2 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(3.12.4)
Requirement already satisfied: pydot<2,>=1.2.0 in /usr/local/lib/python3.7/dist-
packages (from apache-beam->object-detection==0.1) (1.3.0)
Requirement already satisfied: typing-extensions<3.8.0,>=3.7.0 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(3.7.4.3)
Requirement already satisfied: future<1.0.0,>=0.18.2 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(0.18.2)
Requirement already satisfied: fastavro<2,>=0.21.4 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(1.3.3)
Requirement already satisfied: pytz>=2018.3 in /usr/local/lib/python3.7/dist-
packages (from apache-beam->object-detection==0.1) (2018.9)
Requirement already satisfied: grpcio<2,>=1.29.0 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(1.32.0)
Requirement already satisfied: requests<3.0.0,>=2.24.0 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(2.25.1)
Requirement already satisfied: oauth2client<5,>=2.0.1 in
```

```
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(4.1.3)
Requirement already satisfied: numpy<1.20.0,>=1.14.3 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(1.19.5)
Requirement already satisfied: httplib2<0.18.0,>=0.8 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(0.17.4)
Requirement already satisfied: pyarrow<3.0.0,>=0.15.1 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(2.0.0)
Requirement already satisfied: hdfs<3.0.0,>=2.1.0 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(2.6.0)
Requirement already satisfied: crcmod<2.0,>=1.7 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(1.7)
Requirement already satisfied: python-dateutil<3,>=2.8.0 in
/usr/local/lib/python3.7/dist-packages (from apache-beam->object-detection==0.1)
(2.8.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-
packages (from matplotlib->object-detection==0.1) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->object-detection==0.1)
(2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib->object-detection==0.1)
(1.3.1)
Requirement already satisfied: absl-py>=0.2.2 in /usr/local/lib/python3.7/dist-
packages (from tf-slim->object-detection==0.1) (0.10.0)
Requirement already satisfied: setuptools>=18.0 in
/usr/local/lib/python3.7/dist-packages (from pycocotools->object-detection==0.1)
(54.1.2)
Requirement already satisfied: opencv-python>=4.1.0.25 in
/usr/local/lib/python3.7/dist-packages (from lvis->object-detection==0.1)
(4.1.2.30)
Requirement already satisfied: seqeval in /usr/local/lib/python3.7/dist-packages
(from tf-models-official->object-detection==0.1) (1.2.2)
Requirement already satisfied: tensorflow>=2.4.0 in
/usr/local/lib/python3.7/dist-packages (from tf-models-official->object-
detection==0.1) (2.4.1)
Requirement already satisfied: sentencepiece in /usr/local/lib/python3.7/dist-
packages (from tf-models-official->object-detection==0.1) (0.1.95)
Requirement already satisfied: psutil>=5.4.3 in /usr/local/lib/python3.7/dist-
packages (from tf-models-official->object-detection==0.1) (5.4.8)
Requirement already satisfied: opencv-python-headless in
/usr/local/lib/python3.7/dist-packages (from tf-models-official->object-
detection==0.1) (4.5.1.48)
```

```
Requirement already satisfied: tensorflow-addons in
/usr/local/lib/python3.7/dist-packages (from tf-models-official->object-
detection==0.1) (0.12.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-
packages (from tf-models-official->object-detection==0.1) (5.4.1)
Requirement already satisfied: kaggle>=1.3.9 in /usr/local/lib/python3.7/dist-
packages (from tf-models-official->object-detection==0.1) (1.5.10)
Requirement already satisfied: google-cloud-bigquery>=0.31.0 in
/usr/local/lib/python3.7/dist-packages (from tf-models-official->object-
detection==0.1) (1.21.0)
Requirement already satisfied: tensorflow-hub>=0.6.0 in
/usr/local/lib/python3.7/dist-packages (from tf-models-official->object-
detection==0.1) (0.11.0)
Requirement already satisfied: gin-config in /usr/local/lib/python3.7/dist-
packages (from tf-models-official->object-detection==0.1) (0.4.0)
Requirement already satisfied: google-api-python-client>=1.6.7 in
/usr/local/lib/python3.7/dist-packages (from tf-models-official->object-
detection==0.1) (1.12.8)
Requirement already satisfied: tensorflow-datasets in
/usr/local/lib/python3.7/dist-packages (from tf-models-official->object-
detection==0.1) (4.0.1)
Requirement already satisfied: py-cpuinfo>=3.3.0 in
/usr/local/lib/python3.7/dist-packages (from tf-models-official->object-
detection==0.1) (7.0.0)
Requirement already satisfied: tensorflow-model-optimization>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from tf-models-official->object-
detection==0.1) (0.5.0)
Requirement already satisfied: dataclasses in /usr/local/lib/python3.7/dist-
packages (from tf-models-official->object-detection==0.1) (0.6)
Requirement already satisfied: pbr>=0.11 in /usr/local/lib/python3.7/dist-
packages (from mock<3.0.0,>=1.0.1->apache-beam->object-detection==0.1) (5.5.1)
Requirement already satisfied: chardet<5,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.24.0->apache-
beam->object-detection==0.1) (3.0.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.24.0->apache-
beam->object-detection==0.1) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests<3.0.0,>=2.24.0->apache-beam->object-detection==0.1)
(2.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests<3.0.0,>=2.24.0->apache-
beam->object-detection==0.1) (2020.12.5)
Requirement already satisfied: rsa>=3.1.4 in /usr/local/lib/python3.7/dist-
packages (from oauth2client<5,>=2.0.1->apache-beam->object-detection==0.1)
(4.7.2)
Requirement already satisfied: pyasn1-modules>=0.0.5 in
/usr/local/lib/python3.7/dist-packages (from oauth2client<5,>=2.0.1->apache-
```

```
beam->object-detection==0.1) (0.2.8)
Requirement already satisfied: pyasn1>=0.1.7 in /usr/local/lib/python3.7/dist-
packages (from oauth2client<5,>=2.0.1->apache-beam->object-detection==0.1)
(0.4.8)
Requirement already satisfied: docopt in /usr/local/lib/python3.7/dist-packages
(from hdfs<3.0.0,>=2.1.0->apache-beam->object-detection==0.1) (0.6.2)
Requirement already satisfied: scikit-learn>=0.21.3 in
/usr/local/lib/python3.7/dist-packages (from seqeval->tf-models-
official->object-detection==0.1) (0.22.2.post1)
Requirement already satisfied: h5py~=2.10.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.4.0->tf-models-official->object-detection==0.1)
(2.10.0)
Requirement already satisfied: termcolor~=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (1.1.0)
Requirement already satisfied: opt-einsum~=3.3.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (3.3.0)
Requirement already satisfied: wheel~=0.35 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.4.0->tf-models-official->object-detection==0.1)
(0.36.2)
Requirement already satisfied: wrapt~=1.12.1 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.4.0->tf-models-official->object-detection==0.1)
(1.12.1)
Requirement already satisfied: keras-preprocessing~=1.1.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (1.1.2)
Requirement already satisfied: tensorboard~=2.4 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (2.4.1)
Requirement already satisfied: tensorflow-estimator<2.5.0,>=2.4.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (2.4.0)
Requirement already satisfied: astunparse~=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (1.6.3)
Requirement already satisfied: gast==0.3.3 in /usr/local/lib/python3.7/dist-
packages (from tensorflow>=2.4.0->tf-models-official->object-detection==0.1)
(0.3.3)
Requirement already satisfied: flatbuffers~=1.12.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (1.12)
Requirement already satisfied: google-pasta~=0.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (0.2.0)
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python3.7/dist-
packages (from tensorflow-addons->tf-models-official->object-detection==0.1)
(2.7.1)
```

```
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-
packages (from kaggle>=1.3.9->tf-models-official->object-detection==0.1) (4.0.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages
(from kaggle>=1.3.9->tf-models-official->object-detection==0.1) (4.41.1)
Requirement already satisfied: google-cloud-core<2.0dev,>=1.0.3 in
/usr/local/lib/python3.7/dist-packages (from google-cloud-bigquery>=0.31.0->tf-
models-official->object-detection==0.1) (1.0.3)
Requirement already satisfied: google-resumable-media!=0.4.0,<0.5.0dev,>=0.3.1
in /usr/local/lib/python3.7/dist-packages (from google-cloud-
bigquery>=0.31.0->tf-models-official->object-detection==0.1) (0.4.1)
Requirement already satisfied: google-auth-httplib2>=0.0.3 in
/usr/local/lib/python3.7/dist-packages (from google-api-python-
client>=1.6.7->tf-models-official->object-detection==0.1) (0.0.4)
Requirement already satisfied: google-api-core<2dev,>=1.21.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-python-
client>=1.6.7->tf-models-official->object-detection==0.1) (1.26.1)
Requirement already satisfied: google-auth>=1.16.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-python-
client>=1.6.7->tf-models-official->object-detection==0.1) (1.27.1)
Requirement already satisfied: uritemplate<4dev,>=3.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-python-
client>=1.6.7->tf-models-official->object-detection==0.1) (3.0.1)
Requirement already satisfied: dm-tree in /usr/local/lib/python3.7/dist-packages
(from tensorflow-datasets->tf-models-official->object-detection==0.1) (0.1.5)
Requirement already satisfied: promise in /usr/local/lib/python3.7/dist-packages
(from tensorflow-datasets->tf-models-official->object-detection==0.1) (2.3)
Requirement already satisfied: attrs>=18.1.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow-datasets->tf-models-official->object-detection==0.1)
(20.3.0)
Requirement already satisfied: importlib-resources; python_version < "3.9" in
/usr/local/lib/python3.7/dist-packages (from tensorflow-datasets->tf-models-
official->object-detection==0.1) (5.1.2)
Requirement already satisfied: tensorflow-metadata in
/usr/local/lib/python3.7/dist-packages (from tensorflow-datasets->tf-models-
official->object-detection==0.1) (0.28.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn>=0.21.3->seqeval->tf-models-official->object-
detection==0.1) (1.0.1)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.4->tensorflow>=2.4.0->tf-models-official->object-detection==0.1)
(1.0.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.4->tensorflow>=2.4.0->tf-models-official->object-detection==0.1)
(0.4.3)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from
```

```
tensorboard~=2.4->tensorflow>=2.4.0->tf-models-official->object-detection==0.1)
(1.8.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-
packages (from tensorboard~=2.4->tensorflow>=2.4.0->tf-models-official->object-
detection==0.1) (3.3.4)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.7/dist-packages (from python-slugify->kaggle>=1.3.9->tf-
models-official->object-detection==0.1) (1.3)
Requirement already satisfied: googleapis-common-protos<2.0dev,>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from google-api-
core<2dev,>=1.21.0->google-api-python-client>=1.6.7->tf-models-official->object-
detection==0.1) (1.53.0)
Requirement already satisfied: packaging>=14.3 in /usr/local/lib/python3.7/dist-
packages (from google-api-core<2dev,>=1.21.0->google-api-python-
client>=1.6.7->tf-models-official->object-detection==0.1) (20.9)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth>=1.16.0->google-api-
python-client>=1.6.7->tf-models-official->object-detection==0.1) (4.2.1)
Requirement already satisfied: zipp>=0.4; python_version < "3.8" in
/usr/local/lib/python3.7/dist-packages (from importlib-resources; python_version
< "3.9"->tensorflow-datasets->tf-models-official->object-detection==0.1) (3.4.1)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.4->tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (1.3.0)
Requirement already satisfied: importlib-metadata; python_version < "3.8" in
/usr/local/lib/python3.7/dist-packages (from
markdown>=2.6.8->tensorboard~=2.4->tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (3.7.2)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-
packages (from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.4->tensorflow>=2.4.0->tf-models-
official->object-detection==0.1) (3.1.0)
Building wheels for collected packages: object-detection
  Building wheel for object-detection (setup.py): started
  Building wheel for object-detection (setup.py): finished with status 'done'
  Created wheel for object-detection: filename=object_detection-0.1-cp37-none-
any.whl size=1628562
sha256=68c2c213053b9fe6d30c96e73e72e8e8bdce72b4bf7b2b8febd4b61c94777251
  Stored in directory: /tmp/pip-ephem-wheel-cache-5r2gfvbb/wheels/94/49/4b/39b05
1683087a22ef7e80ec52152a27249d1a644ccf4e442ea
Successfully built object-detection
Installing collected packages: object-detection
  Found existing installation: object-detection 0.1
    Uninstalling object-detection-0.1:
      Successfully uninstalled object-detection-0.1
Successfully installed object-detection-0.1
```

2.4 2.4 Importar funções da API de detecção de objetos.

Após a instalação da API, tem-se disponível bilbiotecas de detecção de objetos com classes e funções que devem ser importadas para o programa para poderem ser usadas.

[4]:
```
# Importa bilbiotecas da API de detecção de objetos
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.utils import config_util
from object_detection.utils import colab_utils
from object_detection.builders import model_builder
```

- `viz_utils` contém métodos para desenhar caixas delimitadoras.
- `label_map_util` é uma função que associa números para os nomes das classes de objetos. Observa-se que os modelos geram números para as classes e com essa função é possível associar esses números com os nomes das classes.
- `config_util` função para criar modelos a partir do arquivo de configuração
- `colab_utils` função para anotar imagens
- `model_builder` função para construir modelo e carregar parâmetros

2.5 2.5 Funções auxiliares carregar imagens e desenhar caixas delimitadoras

Na célula abaixo são definidas duas funções, uma para carregar imagens de uma pasta e transformar em tensor Numpy e outra para desenhar as caixas delimitadoras identificadas sobrepostas na imagem. Para visualizar a imagem com as caixas é usado o método `viz_utils.visualize_boxes_and_labels_on_image_array()` da API de detecção de objetos.

1. `load_img_into_numpy_array()` carrega imagem de uma pasta e a transfroma em tensor Numpy
2. `plot_detection()` mostra uma imagem com as caixas delimitadoras sobrepostas

[5]:
```
# Define função para carregar imagem e transformar em tensor Numpy
def load_image_into_numpy_array(path):
    """Load an image from file into a numpy array.

    Puts image into numpy array to feed into tensorflow graph.
    Note that by convention we put it into a numpy array with shape
    (height, width, channels), where channels=3 for RGB.

    Args:
        path: a file path.

    Returns:
        uint8 numpy array with shape (img_height, img_width, 3)
    """
    img_data = tf.io.gfile.GFile(path, 'rb').read()
```

```
    image = Image.open(BytesIO(img_data))
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape((im_height, im_width, 3)).astype(np.
 ↪uint8)

# Define função para mostrar imagem com caixas delimitadoras sobrepostas
def plot_detections(image_np,
                    boxes,
                    classes,
                    scores,
                    category_index,
                    figsize=(12, 16),
                    image_name=None):
    """Wrapper function to visualize detections.

    Args:
        image_np: uint8 numpy array with shape (img_height, img_width, 3)
        boxes: a numpy array of shape [N, 4]
        classes: a numpy array of shape [N]. Note that class indices are 1-based,
                 and match the keys in the label map.
        scores: a numpy array of shape [N] or None.  If scores=None, then
                this function assumes that the boxes to be plotted are␣
 ↪groundtruth
        boxes and plot all boxes as black with no classes or scores.
        category_index: a dict containing category dictionaries (each holding
                        category index `id` and category name `name`) keyed by␣
 ↪category indices.
        figsize: size for the figure.
        image_name: a name for the image file.
    """
    image_np_with_annotations = image_np.copy()
    viz_utils.visualize_boxes_and_labels_on_image_array(
          image_np_with_annotations,
          boxes,
          classes,
          scores,
          category_index,
          use_normalized_coordinates=True,
          min_score_thresh=0.8)
    if image_name:
        plt.imsave(image_name, image_np_with_annotations)
    else:
        plt.imshow(image_np_with_annotations)
```

3 Conjunto de dados

Vamos usar o conjunto de dados originalmente utilizado para detectar cangurus. Porém como muitas imagens desse conjunto mostram também pessoas, vamos utilizá-lo para detectar cangurus e pessoas.

As imagens desse conjunto de dados etá no arquivo compactado `cangurus_pessoas.zip`. Para carregar esse conjutno de dados primeiramente você que que importá-lo para o ambiente do Colab.

Após importar o arquivo deve descompactá-lo executando a célula abaixo.

[6]:
```
!unzip cangurus_pessoas.zip
```

```
Archive:  cangurus_pessoas.zip
replace cangurus_pessoas/test/00004.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00019.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00024.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00045.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00060.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00098.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00100.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00103.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00105.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00111.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00117.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00119.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00123.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00124.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00125.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00129.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00136.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00145.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00146.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00152.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00157.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00159.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00162.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace cangurus_pessoas/test/00166.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: N
```

A descompactação desse arquivo gera uma pasta de nome “cangurus_pessoas” com duas sub pastas. Uma de nome “train”, com as imagens usadas para treinamento, e outra de nome “test” com as imagens usadas para teste.

Verifique no seu ambiente do Colab onde está a pasta “cangurus_pessoas”. Para isso execute a célula abaixo.

[7]:
```
! ls cangurus_pessoas
```

```
test  train
```

3.1 Carregar imagens de treinamento

Para carregar as imagens de treinamento devemos gerar uma lista com os arquivos presentes na pasta "train".

3.2 Exercício #1: Gerar lista de arquivos das imagens

Para carregar as imagens de treinamemto crie um código na célula abaixo para gerar um alista dos arquivos das imagens.

[8]:
```
# Para você fazer: carregar imagens de treinamento

# Importa bilbliotecas glob
from glob import glob

# Define diretório onde se encontram as imagens
train_dir = "cangurus_pessoas/train"

# Escolhe tipos de arquivos desejados
glob_imgs = "cangurus_pessoas/train"

# Cria lista dos nomes dos arquivos
img_paths = [os.path.join(train_dir, fn) for fn in os.listdir(train_dir)]

# Mostra 5 primeiros arquivos da lista
print(img_paths[:5])
```

```
['cangurus_pessoas/train/00077.jpg', 'cangurus_pessoas/train/00012.jpg',
'cangurus_pessoas/train/00032.jpg', 'cangurus_pessoas/train/00040.jpg',
'cangurus_pessoas/train/00101.jpg']
```

3.3 Exercício #2: Carregar imagens de treinamento

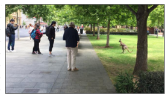
As imagens de treinamento devem carregas, transformadas em tensores Numpy e colocadas na lista `train_images_np`. Para carregar e transfromar as imagens em tensores Numpy use a função `load_image_into_numpy_array` defina na Seção 2.5.

[13]:
```
%matplotlib inline
```

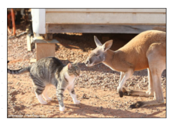
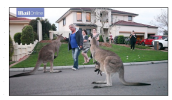
[15]:
```
# Para você fazer: Carregar e visualizar imagens de treinamento

# Iniciliza lista de imagens
train_images_np = [load_image_into_numpy_array(img_path) for img_path in␣
 ↪img_paths]

# Carrega imagens, tranforma em tensores Numpy e inclui na lista
# Insira seu código aqui
#
```

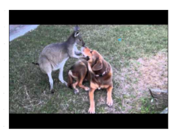
```
# Visualização das imagens
plt.rcParams['axes.grid'] = False
plt.rcParams['xtick.labelsize'] = False
plt.rcParams['ytick.labelsize'] = False
plt.rcParams['xtick.top'] = False
plt.rcParams['xtick.bottom'] = False
plt.rcParams['ytick.left'] = False
plt.rcParams['ytick.right'] = False
plt.rcParams['figure.figsize'] = [16, 60]

for idx, train_image_np in enumerate(train_images_np):
    plt.subplot(7, 3, idx+1)
    plt.imshow(train_image_np)
plt.show()
```

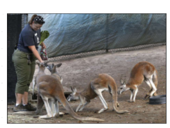


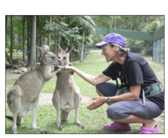
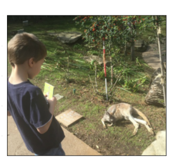








- Observa-se que existem 19 imagens no conjunto de treinamento. Esse número é muito pequeno para realizar um treinamento efetivo, porém a anotação dessas 19 imagens é bastante trabalhosa.

3.4 Anotação das imagens com as caixas delimitadoras

Você tem que anotar as imagens identificando os cangurus e as pessoas. Com visto, isso representa desenhar uma caixa ao redor de cada canguru e cada pessoa presentes nas imagens. Ao fazer isso, são criadas as caixas delimitadoras desejadas para cada imagem, que representam as saídas desejadas.

A API de detecção de objetos disponibiliza a função `colab_utilis.annotate()` para auxiliar a execução dessa tarefa. Infelizmente essa função é bastante limitada e não permite definir as classes de cada caixa, portanto, isso terá que ser realizado após serem definidas todas as caixas de todas as imagens.

Oberva que existem muitos softwares para facilitar a anotação de imagens, um bstante completo e de uso livre para poucas iamgens é o Roboflow (https://roboflow.com/).

Importante: a qualidade do seu modelo de detecção de objetos depende muito da realização de um trabalho de anotação das imagens. Dessa forma, tente ser o mais preciso possível nessa tarefa.

3.5 Exercício #3; Anotação das imagens de treinamento

Execute a célula abaixo para realizar as anotações das 19 imagens de treinamento.

Dicas:

1. Utlize classe 0 para os cangurus e classe 1 para as pessoas.
2. Para facilitar a definição das classes de cada caixa de cada imagem, para cada imagem defina primeiro todas as caixas dos cangurus para depois definir as caixas das pessoas. Isso vai facilitar associar para cada caixa de cada imagem a sua classe correspondente.
3. Após criar a lista de classes é interessante copiá-la para poder usá-l posteriormenet e assim, evitar de ter que repetir essa tarefa de anotação das imagens toda vez que tiver que refazer o trabalho.

[16]:
```
# Para você fazer: Anotação das imagens

# Inicializar a lista de caixas
gt_boxes = []

# Anotação manual das imagens para criar caixas
# Insira seu código aqui
#
colab_utils.annotate(train_images_np, box_storage_pointer=gt_boxes)
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

'--boxes array populated--'

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

'--boxes array populated--'

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

<IPython.core.display.Javascript object>

'--boxes array populated--'

<IPython.core.display.Javascript object>
```

A lista de caixas identificadas nas imagens é criada na variável `gt_boxes` e é apresentada executando a célula abaixo.

[63]:
```
# Apresenta lista de caixas geradas na anotação das imagens
gt_boxes
```

[63]:
```
[array([[0.35166667, 0.        , 0.95166667, 0.444     ]]),
 array([[0.37166667, 0.51818182, 0.845     , 0.99454545],
        [0.11833333, 0.35090909, 0.825     , 0.52727273]]),
 array([[0.05333333, 0.10208333, 0.98833333, 0.53333333]]),
 array([[0.38      , 0.70461538, 0.55666667, 0.80923077],
        [0.17166667, 0.36153846, 0.775     , 0.48      ],
        [0.14666667, 0.25538462, 0.59666667, 0.33076923],
        [0.165     , 0.14923077, 0.57333333, 0.25230769],
        [0.12166667, 0.00307692, 0.52      , 0.10461538]]),
 array([[0.17166667, 0.30875   , 0.92166667, 0.69      ],
        [0.07666667, 0.20375   , 0.96666667, 0.37375   ]]),
 array([[0.16833333, 0.47929936, 0.89833333, 0.88694268]]),
 array([[0.18166667, 0.33679834, 0.75666667, 0.997921  ]]),
 array([[0.18833333, 0.47634069, 0.925     , 1.        ]]),
 array([[0.28666667, 0.        , 0.82      , 0.35377358],
        [0.29333333, 0.50314465, 0.90666667, 0.89937107],
        [0.18166667, 0.3663522 , 0.71166667, 0.45440252],
        [0.24333333, 0.44339623, 0.63      , 0.50628931],
        [0.225     , 0.51886792, 0.36666667, 0.57232704],
```

```
       [0.25      , 0.75157233, 0.49166667, 0.8081761 ]]),
array([[0.14      , 0.49636364, 0.825     , 0.70363636],
       [0.37666667, 0.        , 1.        , 0.49272727],
       [0.13166667, 0.14909091, 0.73333333, 0.48909091],
       [0.265     , 0.76909091, 1.        , 0.96727273]]),
array([[0.31      , 0.28444444, 0.82666667, 0.56      ],
       [0.27833333, 0.5       , 0.81833333, 0.76888889],
       [0.15833333, 0.12888889, 0.83166667, 0.28      ],
       [0.23      , 0.72444444, 0.79666667, 0.89111111]]),
array([[0.14333333, 0.14791667, 0.68833333, 0.48541667]]),
array([[0.12833333, 0.0046875 , 0.79166667, 0.6625    ],
       [0.18166667, 0.6390625 , 0.70666667, 0.925     ]]),
array([[0.39833333, 0.17391304, 0.98333333, 0.43913043],
       [0.40333333, 0.25543478, 0.76166667, 0.36413043],
       [0.52333333, 0.37173913, 0.95333333, 0.76304348],
       [0.41666667, 0.70434783, 0.855     , 0.96304348],
       [0.01666667, 0.025     , 0.87666667, 0.24347826]]),
array([[0.40166667, 0.5186722 , 0.85833333, 0.96127248],
       [0.39      , 0.18672199, 0.71166667, 0.31258645],
       [0.38833333, 0.04979253, 0.70333333, 0.18533887]]),
array([[0.21      , 0.30666667, 0.96333333, 0.47333333],
       [0.23666667, 0.        , 0.985     , 0.37333333],
       [0.13333333, 0.38833333, 1.        , 1.        ]]),
array([[0.59333333, 0.59238095, 0.86166667, 0.93333333],
       [0.08833333, 0.0152381 , 0.98833333, 0.36380952]]),
array([[0.3       , 0.0745098 , 0.87833333, 0.46078431]]),
array([[0.135     , 0.41542289, 0.885     , 1.        ],
       [0.125     , 0.        , 0.84166667, 0.43034826]]),
array([[0.52666667, 0.435     , 0.96666667, 0.93      ],
       [0.2       , 0.21833333, 0.86      , 0.51833333]])]
```

3.6 Exercíco #4: Definição das classes das caixas

Você tem que associar cada caixa presente nas 19 imagens com a sua classe e criar uma lista com as classes de todas as imagens.

As classes são representadas por números inteiros. No caso de se ter duas classes, pode-se definir por exemplo:

- classe = 0, para canguru;
- classe = 1, para pessoa.

Para cada imagem deve ser associada um vetor com as classes de cada caixa. Por exemplo, se a imagem 1 tem três caixas, sendo duas de cangurus e uma de pessoa, para essa imagem o vetor de classes deve ser o seguinte: `[0, 0, 1]`.

A lista de classes é obtida simplesmente unindo os vetores de classes de cada imagem em uma variável tipo lista usando o método `append`.

[52]:
```
# Para você fazer: Definição das classes das caixas

# Incicializar lista de classes das imagens
classes=[]

# Definir vetor de classe de cada imagem e incluir na lista de classes
# Insira seu código aqui
#
classes.append([0])
classes.append([0, 1])
classes.append([0])
classes.append([0, 1, 1, 1, 1])
classes.append([0, 1])
classes.append([0])
classes.append([0])
classes.append([0])
classes.append([0, 0, 1, 1,1,1])
classes.append([0,0,0,1])
classes.append([0,0,1,1])
classes.append([0])
classes.append([0,1])
classes.append([0,0,0,0,1])
classes.append([0,1,1])
classes.append([0,0,1])
classes.append([0,1])
classes.append([0])
classes.append([0,1])
classes.append([0,1])
# Mostra lista de classes
print(classes)
```

```
[[0], [0, 1], [0], [0, 1, 1, 1, 1], [0, 1], [0], [0], [0], [0, 0, 1, 1, 1, 1],
[0, 0, 0, 1], [0, 0, 1, 1], [0], [0, 1], [0, 0, 0, 0, 1], [0, 1, 1], [0, 0, 1],
[0, 1], [0], [0, 1], [0, 1]]
```

3.7 3.3 Preparar dados para o treinamento

Os dados de treinamento devem ser preparados. Essa preparação envolve as seguintes etapas:

1. Criar dicionário que associa número das classes com os seus nomes.
2. Converter a imagens para tensor do TensorFlow e em números reais, e incluir eixo dos exemplos. Observa-se que a normalização das imagens é realizada dentro da rede.
3. Converter lista de caixas para tensor do TensorFlow.
4. Converter lista de classes para tensor do TensorFlow.
5. Converter classes para vetores one-hot.

3.8 Exercício #5: Preparar dados

Na célula abaixo execute as etapas de preparação dos dados descritas acima.

Dicas: - Para converter uma imagem ou um vetor em um tensor TF use a função `tf.convert_to_tensor()`. - Ao converter uma imagem ou um vetor em tensor TF deve definir o tipo de dado, para as imagens e caixas usar `tf.float32` e para as classes `tf.int32`. - Para incluir o eixo dos exemplos nas imagens use a função `tf.expand_dims()`.

Observa-se que não precisa usar a variável `label_id_offset`, como usado na aula, se identificou as classes com 0 e 1.

[64]:
```
# Para você fazer: Preparação dos dados de treinamento

# Define número de classes
num_classes = 2

# Define dicionário com números e nomes das classes
# Insira seu código aqui
#
category_index = {0: {'id': 0, 'name': 'canguru'},
                  1: {'id': 1, 'name': 'pessoa'}}

# Os índices das classes devem ser convertidos para vetores one-hot e todas as␣
 ↪variáveis devem ser
# transformadas em tensores.

# Incializa lista de imagens com números reais
train_image_tensors = []

# Inicializa lista de classes codificadas com vetores one-hot
gt_classes_one_hot_tensors = []

# Inicializa lista de caixas
gt_box_tensors = []


# Iterage em todas as imagens de treinamento incluindo as imagens, as caixas e␣
 ↪as classes transfromadas nas listas
for (train_image_np, gt_box_np, classe) in zip(train_images_np, gt_boxes,␣
 ↪classes):

    # Incluiu eixo dos exemplos e transforma imagens em tensores com números␣
 ↪reais (use axis=0)
    train_image_tensors.append(tf.expand_dims(tf.
 ↪convert_to_tensor(train_image_np, dtype=tf.float32), axis=0))

    # Transforma caixas a serem previstas em tensores TF
    gt_box_tensors.append(tf.convert_to_tensor(gt_box_np,dtype=tf.float32))
```

```
    # Transforma caixas reais em tensores TF
    zero_indexed_groundtruth_classes = tf.convert_to_tensor(np.
 ↪ones(shape=[gt_box_np.shape[0]], dtype=np.int32))

    # Codificação one-hot das classes
    gt_classes_one_hot_tensors.append(tf.
 ↪one_hot(zero_indexed_groundtruth_classes, num_classes))

print('Preparação de dados pronta')
```

```
Preparação de dados pronta
```

Visualização das classe codificadas em vetores one-hot.

[65]:
```
print(gt_classes_one_hot_tensors)
x = np.array(classes[0])
print(x, x.shape)
```

```
[<tf.Tensor: shape=(1, 2), dtype=float32, numpy=array([[0., 1.]],
dtype=float32)>, <tf.Tensor: shape=(2, 2), dtype=float32, numpy=
array([[0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(1, 2), dtype=float32,
numpy=array([[0., 1.]], dtype=float32)>, <tf.Tensor: shape=(5, 2),
dtype=float32, numpy=
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(2, 2), dtype=float32,
numpy=
array([[0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(1, 2), dtype=float32,
numpy=array([[0., 1.]], dtype=float32)>, <tf.Tensor: shape=(1, 2),
dtype=float32, numpy=array([[0., 1.]], dtype=float32)>, <tf.Tensor: shape=(1,
2), dtype=float32, numpy=array([[0., 1.]], dtype=float32)>, <tf.Tensor:
shape=(6, 2), dtype=float32, numpy=
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(4, 2), dtype=float32,
numpy=
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(4, 2), dtype=float32,
```

```
numpy=
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(1, 2), dtype=float32,
numpy=array([[0., 1.]], dtype=float32)>, <tf.Tensor: shape=(2, 2),
dtype=float32, numpy=
array([[0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(5, 2), dtype=float32,
numpy=
array([[0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(3, 2), dtype=float32,
numpy=
array([[0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(3, 2), dtype=float32,
numpy=
array([[0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(2, 2), dtype=float32,
numpy=
array([[0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(1, 2), dtype=float32,
numpy=array([[0., 1.]], dtype=float32)>, <tf.Tensor: shape=(2, 2),
dtype=float32, numpy=
array([[0., 1.],
       [0., 1.]], dtype=float32)>, <tf.Tensor: shape=(2, 2), dtype=float32,
numpy=
array([[0., 1.],
       [0., 1.]], dtype=float32)>]
[0] (1,)
```

3.9 Visualização das imagens de treinamento com as caixas sobrepostas

Para verificar se a preparação de dados foi realizada de forma correta, é importante visualizar as imagens com as caixas sobrepostas indentificadas pelas suas classes e pela probabilidade de estarem corretas.

Para realizar essa visualização é necessário primeiramente definir uma lista de probabilidades ("scores") "fake", porque a função de visualização disponibilizada pelo TensorFlow exige esse dado de entrada.

3.10 Exercíco #6: Definição dos "scores"

Para poder visualizar as imagens de treinamento com as caixas sobrepostas deve-se associar cada caixa presente nas 19 imagens de treinamento com a probabilidade da caixa estar identificando

corretamente um objeto. Além disso, deve-se criar uma lista com as probabilidades de todas as caixas das imagens.

O formato dessa lista de probabilidades é o mesmo usado para definir as classes de cada caixa. Contudo, as probabilidades são números reais variando de 0.0 a 1.0. Como essas probabilidades são somente para permitir visualizar as imagens de treinamento com as caixas deve-se definir todas as probabilidades iguais a 1.0.

[66]:
```
# Para você fazer: Defir lista de probabilidades das caixas

# Inicializa lista de probabilidades
dummy_scores=[]

# Define vetor de probabilidade das caixas de cada imagem e une a lista, segundo␣
 ↪o exemplos
#score1 = [1., 1.]
#dummy_scores.append(score1)
# Insira seu código aqui
#
dummy_scores = [[1. for _ in range(len(classes[i]))] for i in␣
 ↪range(len(classes))]


# Apresenta resultados
print(dummy_scores)
```

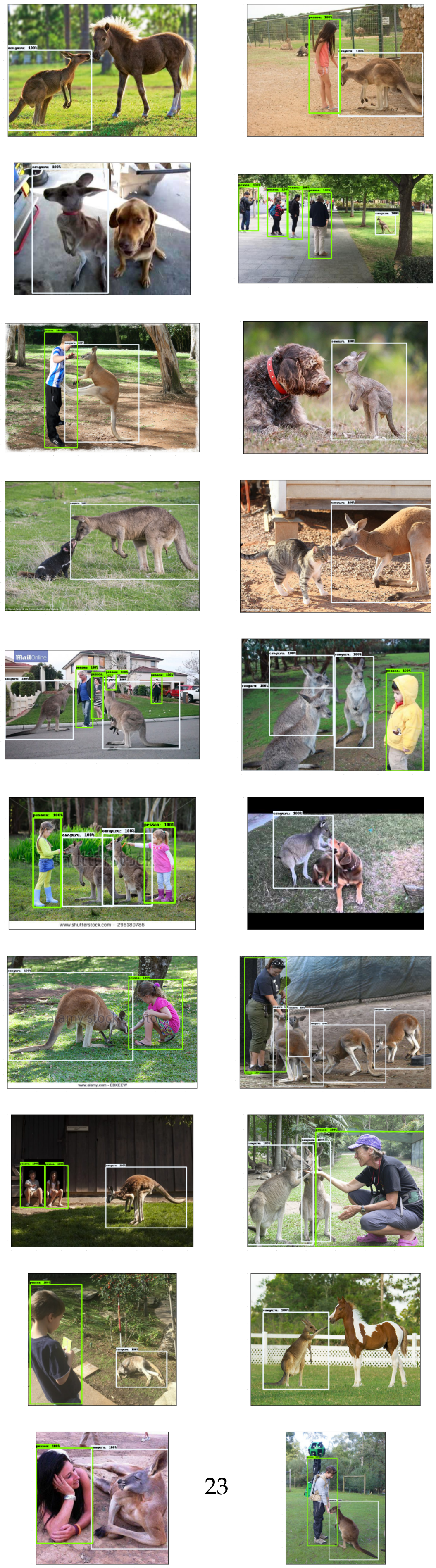
```
[[1.0], [1.0, 1.0], [1.0], [1.0, 1.0, 1.0, 1.0, 1.0], [1.0, 1.0], [1.0], [1.0],
[1.0], [1.0, 1.0, 1.0, 1.0, 1.0, 1.0], [1.0, 1.0, 1.0, 1.0], [1.0, 1.0, 1.0,
1.0], [1.0], [1.0, 1.0], [1.0, 1.0, 1.0, 1.0, 1.0], [1.0, 1.0, 1.0], [1.0, 1.0,
1.0], [1.0, 1.0], [1.0], [1.0, 1.0], [1.0, 1.0]]
```

[67]:
```
len(dummy_scores)
```

[67]: 20

Execute a célula abaixo para visualizar as imagens de treinamento com as caixas sobrepostas.

[70]:
```
plt.figure(figsize=(16, 60))
for idx, train_image_np in enumerate(train_images_np):
    plt.subplot(10, 2, idx+1)
    plot_detections(train_images_np[idx], gt_boxes[idx], classes[idx],␣
 ↪dummy_scores[idx], category_index)
plt.show()
```



4 Criar modelo pré-treinado e restaurar seus parâmetros

Nesta seção deve-se construir o modelo de detecção de objetos do tipo RetinaNet e carregar os seus parâmetros pré-treinados.

Nesse trabalho recomenda-se usar o mesmo modelo usado da aula de detecção de objetos com a API do tensorFlow, ou seja, o modelo SSD-ResNet50 para imagens com dimensão 640x640x3. Esse modelo utiliza o método SSD para detecção dos objejos, usando a rede ResNet-50 para extrair as características das imagens. Se quiser pode tentar usar outro modelo, mas cuidado porque você pode ter dificuldade para configurá-lo e criá-lo.

Execute a célula abaixo para carregar os parâmetros do modelo e os colocá-los na pasta do Colab "content/research/models/detection/test_data".

Observa-se que se você escolheu usar outro modelo é necessário alterar o código das duas células que realizam essas operações de criar o modelo e carregar seus parâmetros.

[71]:
```
# Carrega o arquivo de parâmetros ("checkpoint") e o coloca na pasta
# models/research/object_detection/test_data/SSD_ResNet_50
!wget http://download.tensorflow.org/models/object_detection/tf2/20200711/
 ↪ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz
!tar -xf ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz
!mv ssd_resnet50_v1_fpn_640x640_coco17_tpu-8/checkpoint models/research/
 ↪object_detection/test_data/
```

```
--2021-03-20 03:43:30--  http://download.tensorflow.org/models/object_detection/
tf2/20200711/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz
Resolving download.tensorflow.org (download.tensorflow.org)... 172.217.164.144,
2607:f8b0:4004:814::2010
Connecting to download.tensorflow.org
(download.tensorflow.org)|172.217.164.144|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 244817203 (233M) [application/x-tar]
Saving to: 'ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz'

ssd_resnet50_v1_fpn 100%[===================>] 233.48M   116MB/s    in 2.0s

2021-03-20 03:43:33 (116 MB/s) -
'ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz' saved [244817203/244817203]
```

Para criar o modelo é usado o arquivo de configuração, que define a arquitetura do modelo, e os parâmetros pré-treinados ("checkpoints").

Ao executarmos a célula acima carregamos o arquivo de configuração e os parâmetros pré-treinados do modelo que iremos usar.

- Arquivo de configuração: “models/research/object_detection/configs/tf2/ssd_resnet50_v1_fpn_640x640_8.config”
- Prefixo dos arquivos com parâmetros pré-treinados do modelo: “models/research/object_detection/test_data/checkpoint/ckpt-0”

Se abrir o arquivo de configuração é possível ver a configuração do modelo.

Execute a célula abaixo para criar o modelo.

[72]:
```
tf.keras.backend.clear_session()
print('Construindo modelo e carregando parâmetros pré-treinados...', flush=True)

# Define número de classes do novo conjunto de dados
num_classes = 2

# Define arquivos de configuraçao e prefixo dos arquivos de parâmetros
pipeline_config = 'models/research/object_detection/configs/tf2/
 ↪ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.config'
checkpoint_path = 'models/research/object_detection/test_data/checkpoint/ckpt-0'

# Define diretório para slavar arquivo de configuração e parãmetros do modelo␣
 ↪retreinado
output_directory = 'output/'
output_checkpoint_dir = os.path.join(output_directory, 'checkpoint')

# Alteração do arquivo de configuração
configs = config_util.get_configs_from_pipeline_file(pipeline_config)
model_config = configs['model']
model_config.ssd.num_classes = num_classes
model_config.ssd.freeze_batchnorm = True

# Construção do modelo completo com parâmetros inicializados aleatóriamente
detection_model = model_builder.build(model_config=model_config,␣
 ↪is_training=True)

# Salva novo arquivo de configuração
pipeline_proto = config_util.create_pipeline_proto_from_configs(configs)
config_util.save_pipeline_config(pipeline_proto, output_directory)

# Cria subrede de identificação de caixas usando a subrede do modelo pré-treinado
fake_box_predictor = tf.compat.v2.train.Checkpoint(
    _base_tower_layers_for_heads=detection_model._box_predictor.
 ↪_base_tower_layers_for_heads,
    # Se for desejado deixar também a subrede de classificação, deve-se tirar o␣
 ↪comentário da linha abaixo
    # _prediction_heads=detection_model._box_predictor._prediction_heads,
    _box_prediction_head=detection_model._box_predictor._box_prediction_head,
    )
```

```
# Cria modelo parcial com rede de extração de características e subrede de␣
 ↪identificação das caixas
fake_model = tf.compat.v2.train.Checkpoint(
            _feature_extractor=detection_model._feature_extractor,
            _box_predictor=fake_box_predictor)

# Carrega parâmetros pré-treinados no modelo parcial, ou seja, na rede de␣
 ↪extração
# de característica e na subrede de identificação das caixas
ckpt = tf.compat.v2.train.Checkpoint(model=fake_model)
ckpt.restore(checkpoint_path).expect_partial()

# Transfere modelo parcial com parâmetros pré-treinados para modelo final
exported_ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt_manager = tf.train.CheckpointManager(exported_ckpt, output_checkpoint_dir,␣
 ↪max_to_keep=1)

# Executa modelo com imagem dummy para inicializar variáveis criadas
image, shapes = detection_model.preprocess(tf.zeros([1, 640, 640, 3]))
prediction_dict = detection_model.predict(image, shapes)
_ = detection_model.postprocess(prediction_dict, shapes)

print('Modelo criado!')
```

```
Construindo modelo e carregando parâmetros pré-treinados...
INFO:tensorflow:Writing pipeline config file to output/pipeline.config
Modelo criado!
```

Mostra novo arquivo de configuração para inspeção.

[73]:
```
model_config
```

[73]:
```
ssd {
  num_classes: 2
  image_resizer {
    fixed_shape_resizer {
      height: 640
      width: 640
    }
  }
  feature_extractor {
    type: "ssd_resnet50_v1_fpn_keras"
    depth_multiplier: 1.0
    min_depth: 16
    conv_hyperparams {
      regularizer {
```

```
      l2_regularizer {
        weight: 0.00039999998989515007
      }
    }
    initializer {
      truncated_normal_initializer {
        mean: 0.0
        stddev: 0.029999999329447746
      }
    }
    activation: RELU_6
    batch_norm {
      decay: 0.996999979019165
      scale: true
      epsilon: 0.0010000000474974513
    }
  }
  override_base_feature_extractor_hyperparams: true
  fpn {
    min_level: 3
    max_level: 7
  }
}
box_coder {
  faster_rcnn_box_coder {
    y_scale: 10.0
    x_scale: 10.0
    height_scale: 5.0
    width_scale: 5.0
  }
}
matcher {
  argmax_matcher {
    matched_threshold: 0.5
    unmatched_threshold: 0.5
    ignore_thresholds: false
    negatives_lower_than_unmatched: true
    force_match_for_each_row: true
    use_matmul_gather: true
  }
}
similarity_calculator {
  iou_similarity {
  }
}
box_predictor {
  weight_shared_convolutional_box_predictor {
```

```
    conv_hyperparams {
      regularizer {
        l2_regularizer {
          weight: 0.00039999998989515007
        }
      }
      initializer {
        random_normal_initializer {
          mean: 0.0
          stddev: 0.009999999776482582
        }
      }
      activation: RELU_6
      batch_norm {
        decay: 0.996999979019165
        scale: true
        epsilon: 0.0010000000474974513
      }
    }
    depth: 256
    num_layers_before_predictor: 4
    kernel_size: 3
    class_prediction_bias_init: -4.599999904632568
  }
}
anchor_generator {
  multiscale_anchor_generator {
    min_level: 3
    max_level: 7
    anchor_scale: 4.0
    aspect_ratios: 1.0
    aspect_ratios: 2.0
    aspect_ratios: 0.5
    scales_per_octave: 2
  }
}
post_processing {
  batch_non_max_suppression {
    score_threshold: 9.99999993922529e-09
    iou_threshold: 0.6000000238418579
    max_detections_per_class: 100
    max_total_detections: 100
  }
  score_converter: SIGMOID
}
normalize_loss_by_num_matches: true
loss {
```

```
    localization_loss {
      weighted_smooth_l1 {
      }
    }
    classification_loss {
      weighted_sigmoid_focal {
        gamma: 2.0
        alpha: 0.25
      }
    }
    classification_weight: 1.0
    localization_weight: 1.0
  }
  encode_background_as_zeros: true
  normalize_loc_loss_by_codesize: true
  inplace_batchnorm_update: true
  freeze_batchnorm: true
}
```

5 Treinamento do modelo

A célula abaixo define os parâmetros do modelo que será retreinado e executa o treinamento.

5.1 Exercício #7: Seleção dos parâmetros de treinamento

Na ceúla abaixo você deve definir os seguintes parâmetros de treinamento:

- `batch_size` - define tamanho do lote;
- `learning_rate` - define taxa de aprendizado;
- `num_batches` - define quantos lotes são utilizados no treinamento;
- Otimizador a ser usado.

Observe que o lopp de treinamento se repete em função do número de lotes. Por exemplo, se existirem 100 exemplos de treinamento e os lotes (`batch_size`) forem de 25 exemplos, tem-se 4 lotes de exemplos. Assim, se o número de lotes (`num_batches`) for igual a 200, então cada lote de dados será utlizado no treinamemto 50 vezes.

Após definir esses parâmetros basta executar a célula para realizar o treinamento.

[74]:
```
# Para você fazer: Seleção de parâmetros de treinamemto

tf.keras.backend.set_learning_phase(True)

# Define tamanho do lote de treinamento
batch_size = 10

# Define taxa de treinamento
learning_rate = 0.01
```

```
# Define número de lotes usados no treinamento
num_batches = 100


# Define parâmetros do modelo que são alterados no treinamento
trainable_variables = detection_model.trainable_variables
to_fine_tune = []
prefixes_to_train = [
    'WeightSharedConvolutionalBoxPredictor/WeightSharedConvolutionalBoxHead',
    'WeightSharedConvolutionalBoxPredictor/WeightSharedConvolutionalClassHead']
for var in trainable_variables:
    if any([var.name.startswith(prefix) for prefix in prefixes_to_train]):
        to_fine_tune.append(var)


# Define função para carregar modelo e otimizador para treinamento
def get_model_train_step_function(model, optimizer, vars_to_fine_tune):
    """Get a tf.function for training step."""

    # Define função para executar um passo de treinamento que processa um lote␣
 ↪de dados
    # Função é criada na forma tf.function para aumentar velocidade de computação
    # Pode-se comentar a linha @tf.function para desabilitar essa opção
    #@tf.function
    def train_step_fn(image_tensors,
                      groundtruth_boxes_list,
                      groundtruth_classes_list):
        """A single training iteration.

        Args:
        image_tensors: A list of [1, height, width, 3] Tensor of type tf.float32.
             Note that the height and width can vary across images, as they are
             reshaped within this function to be 640x640.
        groundtruth_boxes_list: A list of Tensors of shape [N_i, 4] with type
             tf.float32 representing groundtruth boxes for each image in the␣
 ↪batch.
        groundtruth_classes_list: A list of Tensors of shape [N_i, num_classes]
             with type tf.float32 representing groundtruth boxes for each image␣
 ↪in
             the batch.

        Returns:
            A scalar tensor representing the total loss for the input batch.
        """
        # Define dimensão do tensor com dados e entrada
        shapes = tf.constant(batch_size * [[640, 640, 3]], dtype=tf.int32)
```

```
        # Obtém saídas reais
        model.provide_groundtruth(
              groundtruth_boxes_list=groundtruth_boxes_list,
              groundtruth_classes_list=groundtruth_classes_list)

        # Calcula gradiente da função de custo em relação aos parâmetros
        with tf.GradientTape() as tape:
            # Pré-processa imagens do lote
            preprocessed_images = tf.concat([detection_model.
 ↪preprocess(image_tensor)[0]
                                             for image_tensor in image_tensors],␣
 ↪axis=0)

            # Executa modelo para obter previsões das classes e caixas
            prediction_dict = model.predict(preprocessed_images, shapes)

            # Calcula função de custo
            losses_dict = model.loss(prediction_dict, shapes)
            total_loss = losses_dict['Loss/localization_loss'] +␣
 ↪losses_dict['Loss/classification_loss']

            # Calcula gradiente da função de custo em relação aos parâmetros␣
 ↪treináveis
            gradients = tape.gradient(total_loss, vars_to_fine_tune)

            # Aplica otimizador para atualizar parâmetros
            optimizer.apply_gradients(zip(gradients, vars_to_fine_tune))

        return total_loss
    return train_step_fn

# Define otimizador (SGD com momento)
# Insira seu código aqui
optimizer = tf.keras.optimizers.SGD(learning_rate=learning_rate)

# Inicializa função de treinamento passando modelo, otimizador e lista de␣
 ↪parâmetros treináveis
train_step_fn = get_model_train_step_function(detection_model, optimizer,␣
 ↪to_fine_tune)

print('Inicio do treinamento!', flush=True)

# Executa loop de treinamento
for idx in range(num_batches):
    # Embaralha dados aleatóriamente
```

```
    all_keys = list(range(len(train_images_np)))
    random.shuffle(all_keys)
    example_keys = all_keys[:batch_size]

    # Obtém dados de treinamenteo
    gt_boxes_list = [gt_box_tensors[key] for key in example_keys]
    gt_classes_list = [gt_classes_one_hot_tensors[key] for key in example_keys]
    image_tensors = [train_image_tensors[key] for key in example_keys]

    # Realiza um passo de treinamento
    total_loss = train_step_fn(image_tensors, gt_boxes_list, gt_classes_list)

    if idx % 10 == 0:
        print('batch ' + str(idx) + ' of ' + str(num_batches) + ', loss=' + ␣
 ↪str(total_loss.numpy()), flush=True)

# Salva novos parâmetros do modelo
ckpt_manager.save()

print('Término do treinamento!')
```

```
Inicio do treinamento!
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/backend.py:434:
UserWarning: `tf.keras.backend.set_learning_phase` is deprecated and will be
removed after 2020-10-11. To update it, simply pass a True/False value to the
`training` argument of the `__call__` method of your layer or model.
  warnings.warn('`tf.keras.backend.set_learning_phase` is deprecated and '
```

```
batch 0 of 100, loss=1.2901722
batch 10 of 100, loss=0.5222959
batch 20 of 100, loss=0.4116385
batch 30 of 100, loss=0.32264596
batch 40 of 100, loss=0.3110628
batch 50 of 100, loss=0.23505315
batch 60 of 100, loss=0.2476586
batch 70 of 100, loss=0.27782196
batch 80 of 100, loss=0.27647865
batch 90 of 100, loss=0.25119516
Término do treinamento!
```

6 teste do modelo

Para testar o modelo retreinado com o novo conjunto de dados vamos usar as imagens de teste.

6.1 Exercício #8: Gerar lista de arquivos das imagens de teste

Para carregar as imagens de treinamento deve-se gerar uma lista com os arquivos presentes na pasta "cangurus_pessoas/test". Crie um código na célula abaixo para gerar uma lista dos arquivos das imagens de teste.

[76]:
```
a = [0, 1, 2]
a[-2:]
```

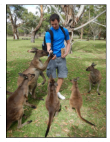
[76]:
```
[1, 2]
```

[78]:
```
# Para você fazer: Criar lista de arquivos das imagens de teste

# Define diretório onde se encontram as imagens
test_dir = "cangurus_pessoas/test"

# Escolhe tipos de arquivos desejados
glob_imgs = 'jpg'

# Cria lista dos nomes dos arquivos
img_paths = [os.path.join(test_dir, fn) for fn in os.listdir(test_dir)]

# Número de imagens de teste
print(len(img_paths))

# Inicializa lista de imagens de teste
test_images_np = [load_image_into_numpy_array(img_path) for img_path in␣
 ↪img_paths]

# Le arquivos da imagens e inclui dados na lista de imagens
# Insira seu código aqui
#


# Mostra imagens de teste
plt.rcParams['axes.grid'] = False
plt.rcParams['xtick.labelsize'] = False
plt.rcParams['ytick.labelsize'] = False
plt.rcParams['xtick.top'] = False
plt.rcParams['xtick.bottom'] = False
plt.rcParams['ytick.left'] = False
plt.rcParams['ytick.right'] = False
plt.rcParams['figure.figsize'] = [16, 60]

for idx, test_image_np in enumerate(test_images_np):
    plt.subplot(6, 5, idx+1)
    plt.imshow(test_image_np)
plt.show()
```





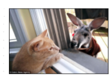
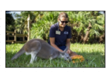
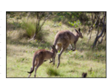




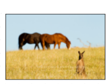
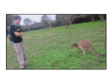


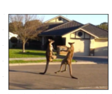









Execute a célula abaixo para criar a a função `detect()`, que recebe uma imagem na forma de tensor TF e calcula as previsões de objetos presentes na imagem.

[79]:
```
# Cria função para calcular previsões do modelo
# Função é criada na forma tf.function para aumentar velocidade de computação
# Pode-se comentar a linha @tf.function para desabilitar essa opção
@tf.function
def detect(input_tensor):
    """Run detection on an input image.

    Args:
        input_tensor: A [1, height, width, 3] Tensor of type tf.float32.
        Note that height and width can be anything since the image will be
        immediately resized according to the needs of the model within this
        function.

    Returns:
       A dict containing 3 Tensors (`detection_boxes`, `detection_classes`, and␣
 ↪`detection_scores`).
    """
    # Pré-processa imagens
    preprocessed_image, shapes = detection_model.preprocess(input_tensor)
    # calcula previsões do modelo
    prediction_dict = detection_model.predict(preprocessed_image, shapes)
    # Retorna previsões pós-processadas
    return detection_model.postprocess(prediction_dict, shapes)
```

6.2 Exercício #9: Calculo das pervisões do modelo

Complete a célula abaixo e depois a execute para calcular as previsões do seu modelo e mostrar as imagens com as caixas detectadas sobrepostas.

Note que a saída da função `detect()` é um dicionário que contém as seguintes `keys`:

```
detections['detection_boxes'];
detections['detection_classes'];
detections['detection_scores'].
```

[101]:
```
# Para você fazer: Preparação das imagens de cálculo e cálculo das previsões

# Tamanho de apresentação das imagens
plt.figure(figsize=(15, 60))

# Loop para executar previsões do modelo nas imagens de teste
for i in range(len(test_images_np)):
```

```
    # Transforma imagens em tensor do TF de números reais
    input_tensor = tf.convert_to_tensor(test_images_np[i], dtype=tf.float32)

    # Inclui eixo dos exemplos nas imagens
    input_tensor = tf.expand_dims(input_tensor, axis=0)

    # Calcula previsões
    detections = detect(input_tensor)

    # Chama função plot_detections() para criar imagens com  as caixas
    plt.subplot(9, 3, i+1)
    # Insira seu código aqui
    #
    plot_detections(
        test_images_np[i],
        detections['detection_boxes'][0].numpy(),
        detections['detection_classes'][0].numpy().astype(np.uint32),
        detections['detection_scores'][0].numpy(),
        category_index)

plt.show()
```

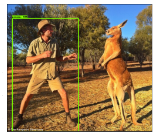
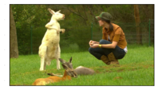
```
WARNING:tensorflow:5 out of the last 29 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:6 out of the last 30 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:7 out of the last 31 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
```

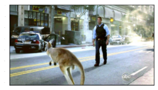
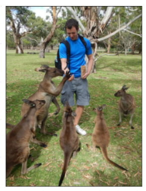
```
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:8 out of the last 32 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:9 out of the last 33 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 34 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:11 out of the last 35 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
```

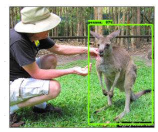
```
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 11 calls to <function detect at
```

```
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:10 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:11 out of the last 12 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
```

```
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:11 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:11 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
WARNING:tensorflow:11 out of the last 11 calls to <function detect at
0x7f68bc4d7830> triggered tf.function retracing. Tracing is expensive and the
excessive number of tracings could be due to (1) creating @tf.function
repeatedly in a loop, (2) passing tensors with different shapes, (3) passing
Python objects instead of tensors. For (1), please define your @tf.function
outside of the loop. For (2), @tf.function has experimental_relax_shapes=True
option that relaxes argument shapes that can avoid unnecessary retracing. For
(3), please refer to
https://www.tensorflow.org/guide/function#controlling_retracing and
https://www.tensorflow.org/api_docs/python/tf/function for  more details.
```

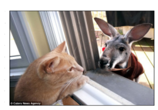
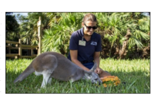
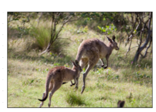




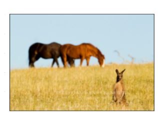
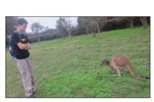
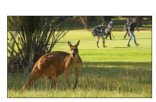
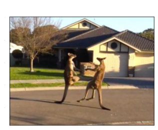
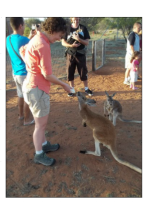









7 Utilização do modelo treinado

Como visto em aula, o modelo criado utiliza camadas customizadas e o método de programação em "graphos" do TensorFlow de forma que para ser utilizado após ter sido salvo ele tem que ser reconstruído. O modelo salvo é reconstruído a partir do arquivo de configuração e dos parâmetros salvos. Assim, se quiser reutilizar esse modelo você deve seguir os passos apresentados em aula para fazer isso.

[ ]: