Abhinav Jha 30-08-2020

## Problem Statement:

Train your own custom object detector for detecting the given two objects using python,

1.Laptop

2. Street light.

(Expected output will be like, if we pass video/frames as an input it will detect the given trained objects from the video/frames and display the bounding box for that detected object)

## Prerequisite:

Python, OpenCV, Tensorflow, Pandas

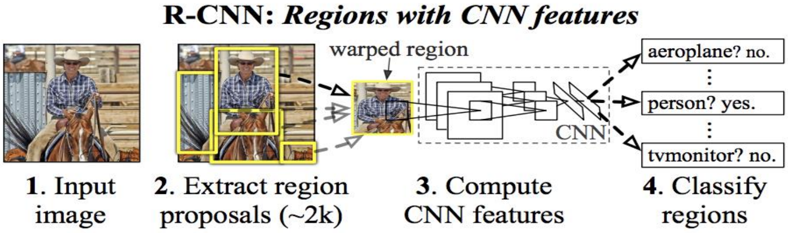
## Software and Algorithm used:

### Software:

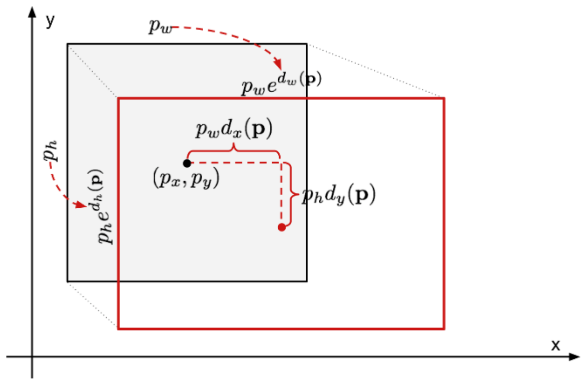
1) LabelImg:
   a. LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface.
   b. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet.

### Algorithm:

1) RCNN:
   a. R-CNN is short for "Region-based Convolutional Neural Networks".
   b. The main idea is composed of two steps. First, using selective search, it identifies a manageable number of bounding-box object region candidates ("region of interest" or "RoI"). And then it extracts CNN features from each region independently for classification.



2) Bounding Box Regression:
   a. Given a predicted bounding box coordinate p=(px,py,pw,ph) (centre coordinate, width, height) and its corresponding ground truth box coordinates g=(gx,gy,gw,gh) , the regressor is configured to learn scale-invariant transformation between two centres and log-scale transformation between widths and heights.
   b. All the transformation functions take p as input.



## Procedure:

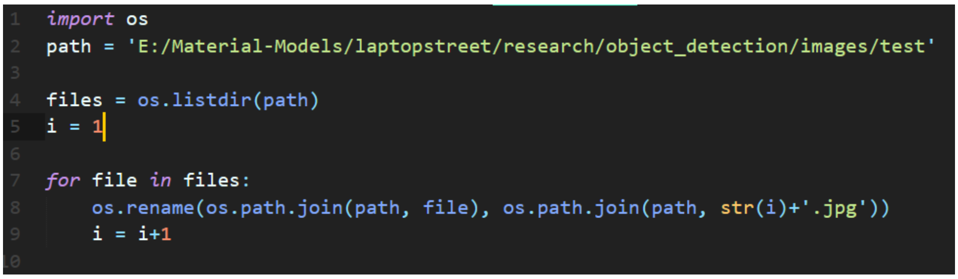
1) Gather a dataset:

The dataset is self-made.

Images are downloaded from google.

2) Renaming the dataset:

a. The self-designed python program is created to get all images in one format and proper name structure.

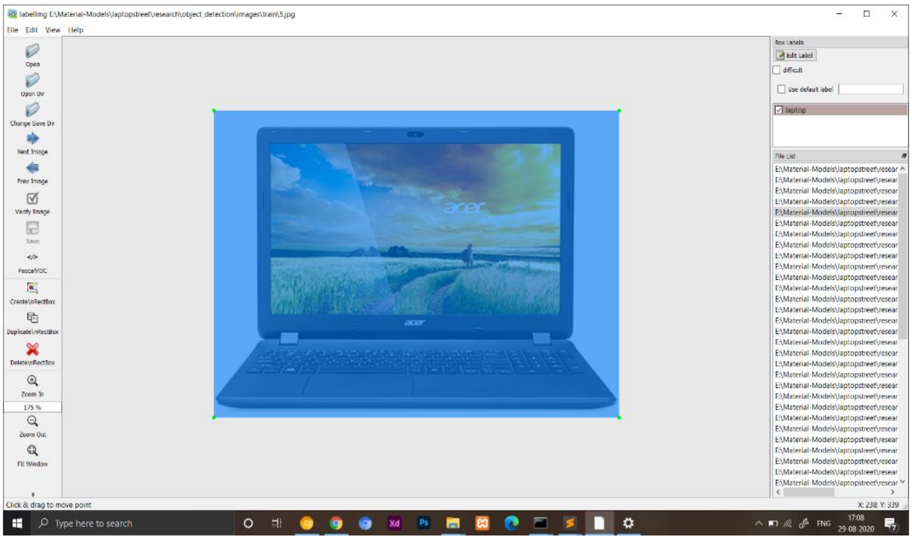


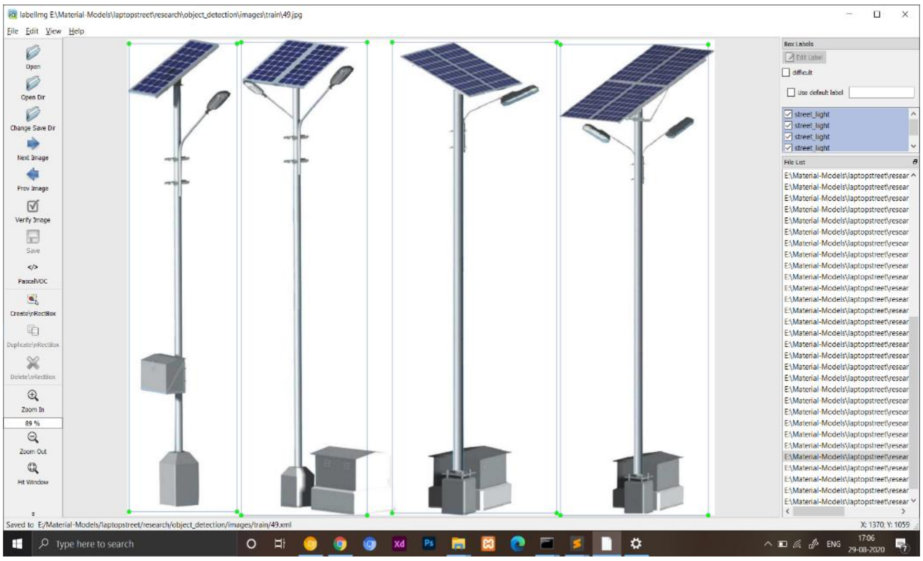
```python
import os
path = 'E:/Material-Models/laptopstreet/research/object_detection/images/test'

files = os.listdir(path)
i = 1

for file in files:
    os.rename(os.path.join(path, file), os.path.join(path, str(i)+'.jpg'))
    i = i+1

```

3) Labeling the image:

We need the height, width and class of each image to train our object detection model. This includes the associated xmin, xmax, ymin, and ymax bounding boxes.

I used the labelImg software written in python an open source program that saves an XML label for each image.

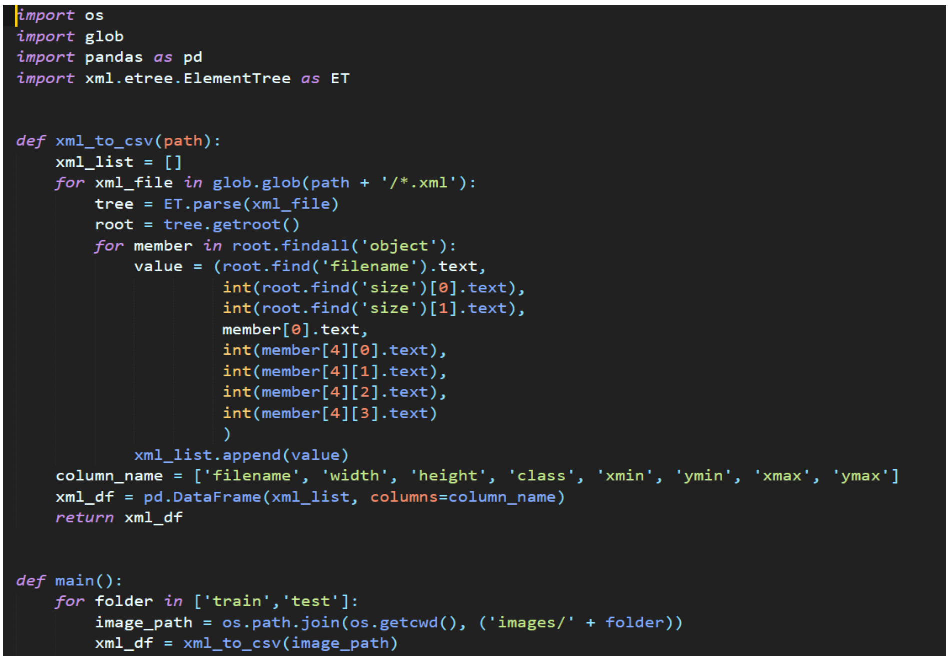




Saved to E:/Material-Models/laptopstreet/research/object_detection/images/train/49.xml

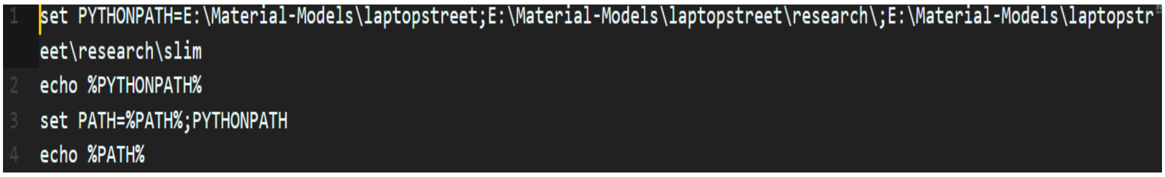
## 4) Creating TFRecords:

TFRecords is an input data to the TensorFlow training model create `.csv` files from `.xml` files



```python
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET


def xml_to_csv(path):
    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        for member in root.findall('object'):
            value = (root.find('filename').text,
                     int(root.find('size')[0].text),
                     int(root.find('size')[1].text),
                     member[0].text,
                     int(member[4][0].text),
                     int(member[4][1].text),
                     int(member[4][2].text),
                     int(member[4][3].text)
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def main():
    for folder in ['train','test']:
        image_path = os.path.join(os.getcwd(), ('images/' + folder))
        xml_df = xml_to_csv(image_path)
```

5) Configure environment variable:

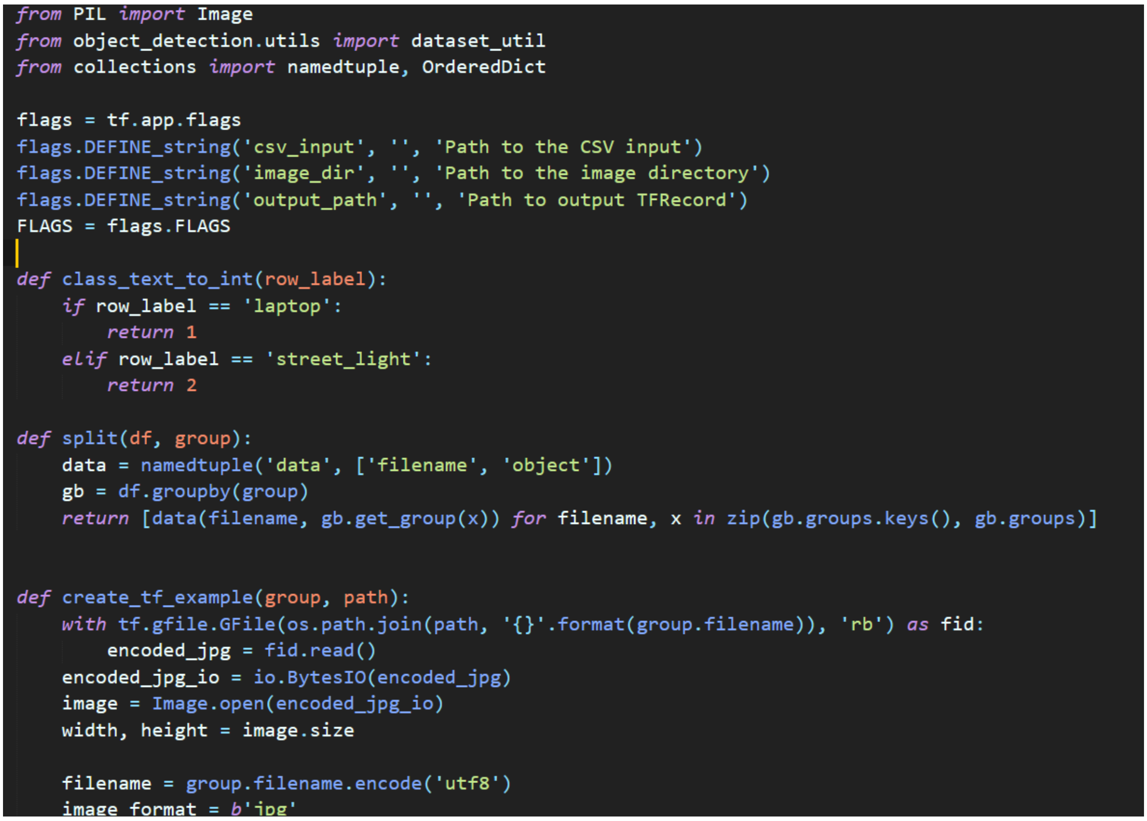


```
set PYTHONPATH=E:\Material-Models\laptopstreet;E:\Material-Models\laptopstreet\research\;E:\Material-Models\laptopstreet\research\slim
echo %PYTHONPATH%
set PATH=%PATH%;PYTHONPATH
echo %PATH%
```

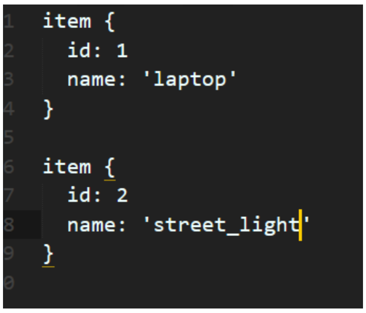
6) Compile Protobufs:

   a. Protobuf (Protocol Buffers) libraries must be compiled, it used by TensorFlow to configure model and training parameters.

7) Generating records and creating classes.



```
from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('image_dir', '', 'Path to the image directory')
flags.DEFINE_string('output_path', '', 'Path to output TFRecord')
FLAGS = flags.FLAGS

def class_text_to_int(row_label):
    if row_label == 'laptop':
        return 1
    elif row_label == 'street_light':
        return 2

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]


def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
```

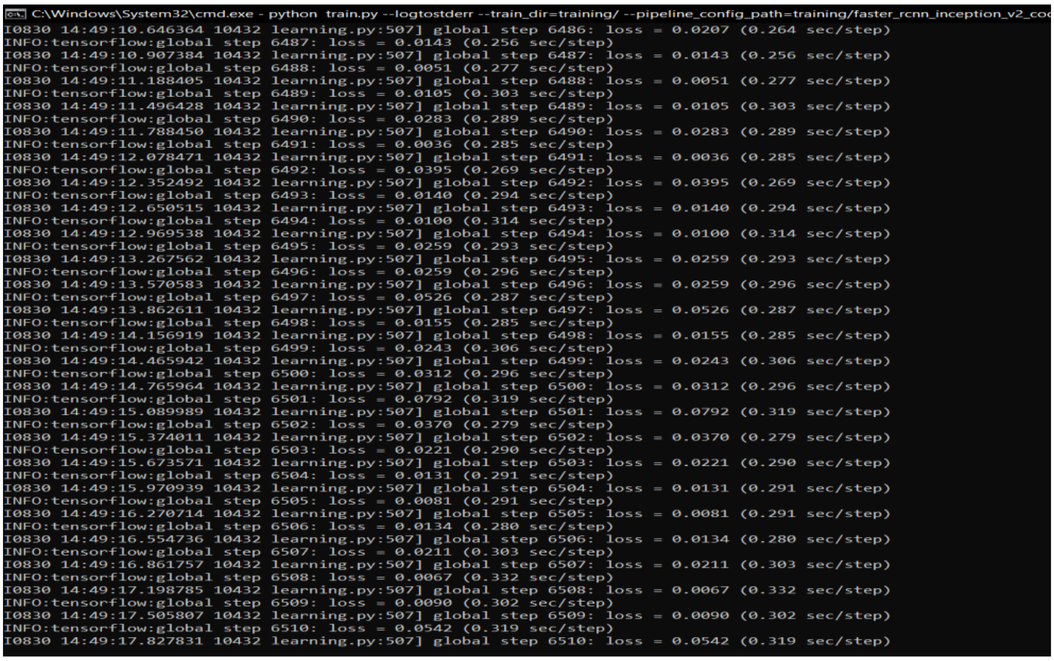


```
item {
  id: 1
  name: 'laptop'
}

item {
  id: 2
  name: 'street_light'
}
```

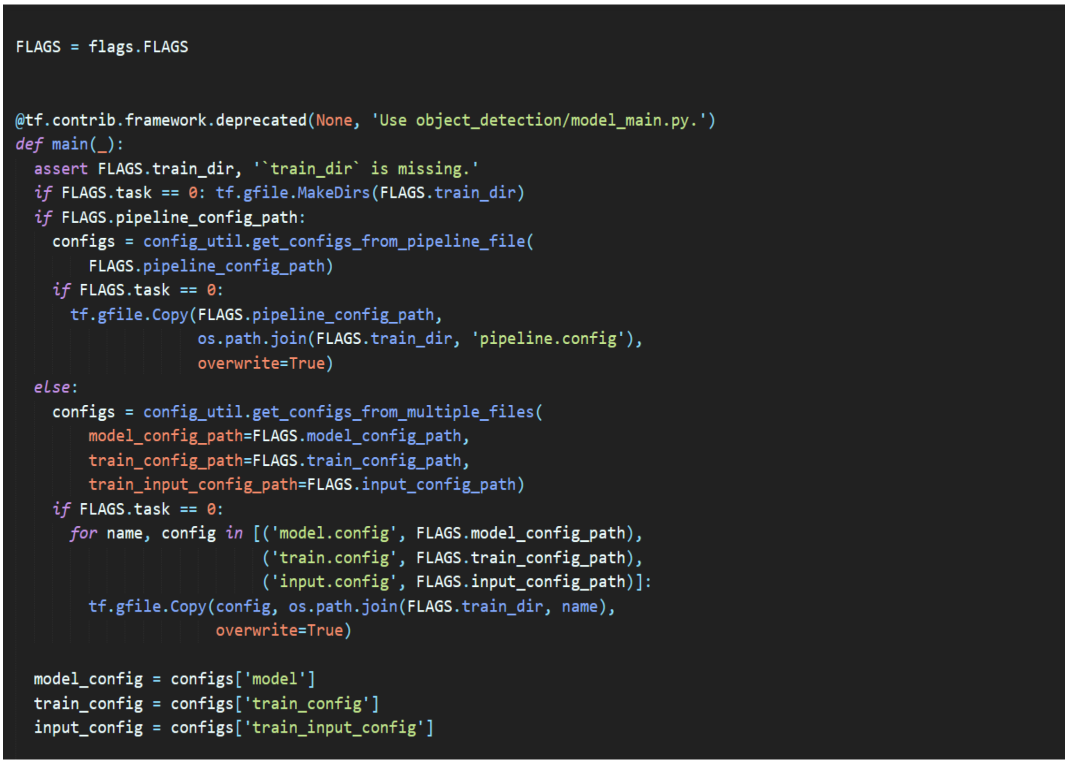
8) Creating a check point.

9) Training the model with 70 percent train data and 30 percent test data.

   a. Model is trained for 6807 steps with average loss = 0.0145.
   b. Accuracy of model is 98%.



```
C:\Windows\System32\cmd.exe - python  train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/faster_rcnn_inception_v2_coc
I0830 14:49:10.646364 10432 learning.py:507] global step 6486: loss = 0.0207 (0.264 sec/step)
INFO:tensorflow:global step 6487: loss = 0.0143 (0.256 sec/step)
I0830 14:49:10.907384 10432 learning.py:507] global step 6487: loss = 0.0143 (0.256 sec/step)
INFO:tensorflow:global step 6488: loss = 0.0051 (0.277 sec/step)
I0830 14:49:11.188405 10432 learning.py:507] global step 6488: loss = 0.0051 (0.277 sec/step)
INFO:tensorflow:global step 6489: loss = 0.0105 (0.303 sec/step)
I0830 14:49:11.496428 10432 learning.py:507] global step 6489: loss = 0.0105 (0.303 sec/step)
INFO:tensorflow:global step 6490: loss = 0.0283 (0.289 sec/step)
I0830 14:49:11.788450 10432 learning.py:507] global step 6490: loss = 0.0283 (0.289 sec/step)
INFO:tensorflow:global step 6491: loss = 0.0036 (0.285 sec/step)
I0830 14:49:12.078471 10432 learning.py:507] global step 6491: loss = 0.0036 (0.285 sec/step)
INFO:tensorflow:global step 6492: loss = 0.0395 (0.269 sec/step)
I0830 14:49:12.352492 10432 learning.py:507] global step 6492: loss = 0.0395 (0.269 sec/step)
INFO:tensorflow:global step 6493: loss = 0.0140 (0.294 sec/step)
I0830 14:49:12.650515 10432 learning.py:507] global step 6493: loss = 0.0140 (0.294 sec/step)
INFO:tensorflow:global step 6494: loss = 0.0100 (0.314 sec/step)
I0830 14:49:12.969538 10432 learning.py:507] global step 6494: loss = 0.0100 (0.314 sec/step)
INFO:tensorflow:global step 6495: loss = 0.0259 (0.293 sec/step)
I0830 14:49:13.267562 10432 learning.py:507] global step 6495: loss = 0.0259 (0.293 sec/step)
INFO:tensorflow:global step 6496: loss = 0.0259 (0.296 sec/step)
I0830 14:49:13.570583 10432 learning.py:507] global step 6496: loss = 0.0259 (0.296 sec/step)
INFO:tensorflow:global step 6497: loss = 0.0526 (0.287 sec/step)
I0830 14:49:13.862611 10432 learning.py:507] global step 6497: loss = 0.0526 (0.287 sec/step)
INFO:tensorflow:global step 6498: loss = 0.0155 (0.285 sec/step)
I0830 14:49:14.156919 10432 learning.py:507] global step 6498: loss = 0.0155 (0.285 sec/step)
INFO:tensorflow:global step 6499: loss = 0.0243 (0.306 sec/step)
I0830 14:49:14.465942 10432 learning.py:507] global step 6499: loss = 0.0243 (0.306 sec/step)
INFO:tensorflow:global step 6500: loss = 0.0312 (0.296 sec/step)
I0830 14:49:14.765964 10432 learning.py:507] global step 6500: loss = 0.0312 (0.296 sec/step)
INFO:tensorflow:global step 6501: loss = 0.0792 (0.319 sec/step)
I0830 14:49:15.089989 10432 learning.py:507] global step 6501: loss = 0.0792 (0.319 sec/step)
INFO:tensorflow:global step 6502: loss = 0.0370 (0.279 sec/step)
I0830 14:49:15.374011 10432 learning.py:507] global step 6502: loss = 0.0370 (0.279 sec/step)
INFO:tensorflow:global step 6503: loss = 0.0221 (0.290 sec/step)
I0830 14:49:15.673571 10432 learning.py:507] global step 6503: loss = 0.0221 (0.290 sec/step)
INFO:tensorflow:global step 6504: loss = 0.0131 (0.291 sec/step)
I0830 14:49:15.970939 10432 learning.py:507] global step 6504: loss = 0.0131 (0.291 sec/step)
INFO:tensorflow:global step 6505: loss = 0.0081 (0.291 sec/step)
I0830 14:49:16.270714 10432 learning.py:507] global step 6505: loss = 0.0081 (0.291 sec/step)
INFO:tensorflow:global step 6506: loss = 0.0134 (0.280 sec/step)
I0830 14:49:16.554736 10432 learning.py:507] global step 6506: loss = 0.0134 (0.280 sec/step)
INFO:tensorflow:global step 6507: loss = 0.0211 (0.303 sec/step)
I0830 14:49:16.861757 10432 learning.py:507] global step 6507: loss = 0.0211 (0.303 sec/step)
INFO:tensorflow:global step 6508: loss = 0.0067 (0.332 sec/step)
I0830 14:49:17.198785 10432 learning.py:507] global step 6508: loss = 0.0067 (0.332 sec/step)
INFO:tensorflow:global step 6509: loss = 0.0090 (0.302 sec/step)
I0830 14:49:17.505807 10432 learning.py:507] global step 6509: loss = 0.0090 (0.302 sec/step)
INFO:tensorflow:global step 6510: loss = 0.0542 (0.319 sec/step)
I0830 14:49:17.827831 10432 learning.py:507] global step 6510: loss = 0.0542 (0.319 sec/step)
```

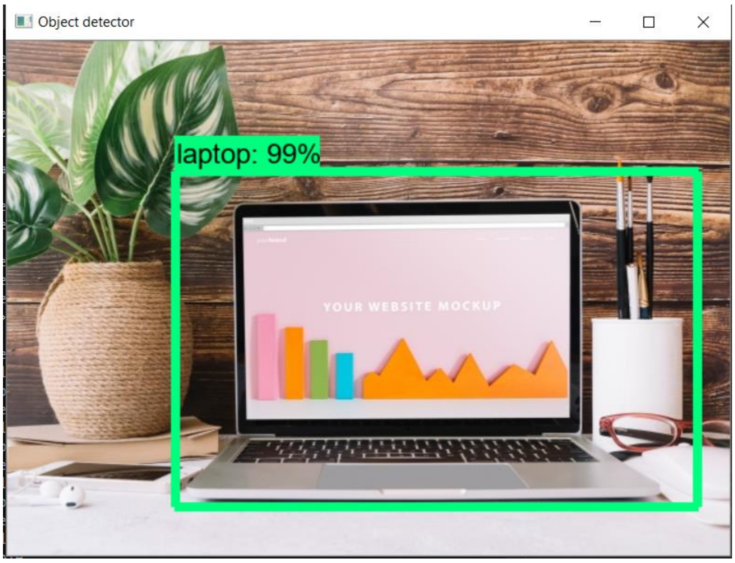


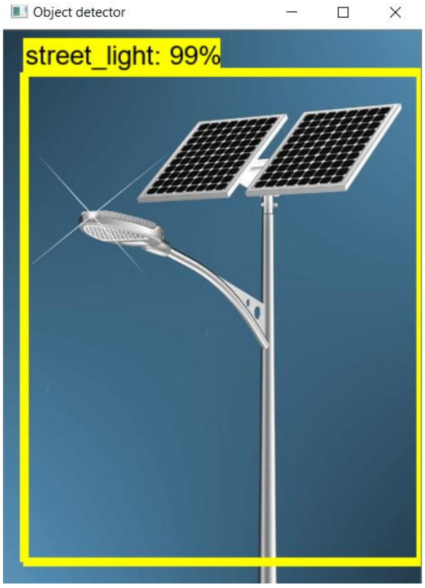
```python
FLAGS = flags.FLAGS


@tf.contrib.framework.deprecated(None, 'Use object_detection/model_main.py.')
def main(_):
  assert FLAGS.train_dir, '`train_dir` is missing.'
  if FLAGS.task == 0: tf.gfile.MakeDirs(FLAGS.train_dir)
  if FLAGS.pipeline_config_path:
    configs = config_util.get_configs_from_pipeline_file(
        FLAGS.pipeline_config_path)
    if FLAGS.task == 0:
      tf.gfile.Copy(FLAGS.pipeline_config_path,
                    os.path.join(FLAGS.train_dir, 'pipeline.config'),
                    overwrite=True)
  else:
    configs = config_util.get_configs_from_multiple_files(
        model_config_path=FLAGS.model_config_path,
        train_config_path=FLAGS.train_config_path,
        train_input_config_path=FLAGS.input_config_path)
    if FLAGS.task == 0:
      for name, config in [('model.config', FLAGS.model_config_path),
                           ('train.config', FLAGS.train_config_path),
                           ('input.config', FLAGS.input_config_path)]:
        tf.gfile.Copy(config, os.path.join(FLAGS.train_dir, name),
                      overwrite=True)

  model_config = configs['model']
  train_config = configs['train_config']
  input_config = configs['train_input_config']
```

10) Exporting the model.

## Outputs Screenshots:

1) Via image.





2) Via video.
   a. Video Link:

https://drive.google.com/file/d/1fXNYZ4lFcrYfC_8DDCffIY1njLK5Ppjw/view?usp=sharing

   b. Demonstration Link:

https://drive.google.com/file/d/1b6BMQqy9yFykf6r-dflTr4cX7UceqNNT/view?usp=sharing

3) Via Webcam.
   a. Demonstration Link:

https://drive.google.com/file/d/1qdJYq_DiC9KryFgRGkZPjcSmbt8NnC7T/view?usp=sharing

## Conclusion:

The model is trained successfully and given good results.