

- Запустите детектор (ssdMobile_v2 или faster_rcnn, или любой другой детектор) для своей картинки и попробуйте найти 10 объектов, 100 объектов.

```
from __future__ import absolute_import, division, print_function, unicode_literals

import tensorflow as tf

# For running inference on the TF-Hub module.
import tensorflow as tf

import tensorflow_hub as hub

# For downloading the image.
import matplotlib.pyplot as plt
import tempfile
from six.moves.urllib.request import urlopen
from six import BytesIO

# For drawing onto the image.
import numpy as np
from PIL import Image
from PIL import ImageColor
from PIL import ImageDraw
from PIL import ImageFont
from PIL import ImageOps

# For measuring the inference time.
import time
%matplotlib inline

# Check available GPU devices.
print("The following GPU devices are available: %s" % tf.test.gpu_device_name())

The following GPU devices are available:
```

The following GPU devices are available:

```
from google.colab import files  
import cv2 as cv  
  
files.upload()
```

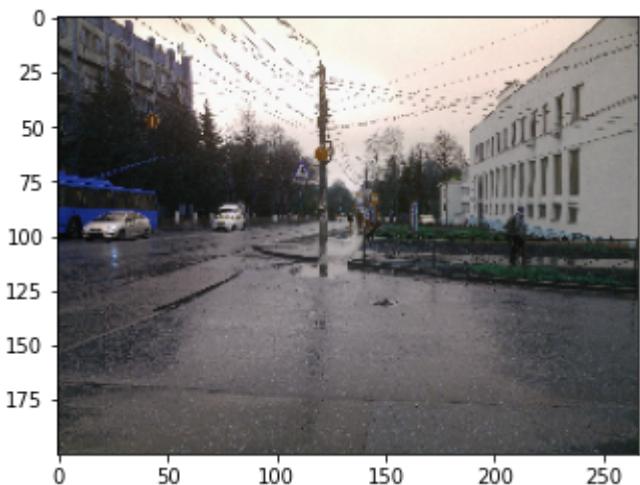
Выбрать файлы img7.jpg

▼ OpenCV - Selectiv Search

```
#Читаем картинку
im = cv.imread('/content/img7.jpg')

# размеры изображения
newH = 200 #высота
newW = int(im.shape[1]*200/im.shape[0]) # ширина
#сжатая картинка
im = cv.resize(im, (newW, newH))
#Выводим

plt.imshow(im)
plt.show()
```



```
#Настройка модели
cv.setUseOptimized(True) #режим оптимизации
cv.setNumThreads(6)      #количество потоков

# Объект Selective Search Segmentation
sss = cv.ximgproc.segmentation.createSelectiveSearchSegmentation()

# запускаем с картинкой
sss.setBaseImage(im)

# Выбираем точный метод
sss.switchToSelectiveSearchQuality()

# Делаем selective search segmentation
rects = sss.process()
#вернёт массив прямоугольников, где возможно что-то есть
print(rects.shape) #число найденных объектов

(1510, 4)
```

#Построим области

```

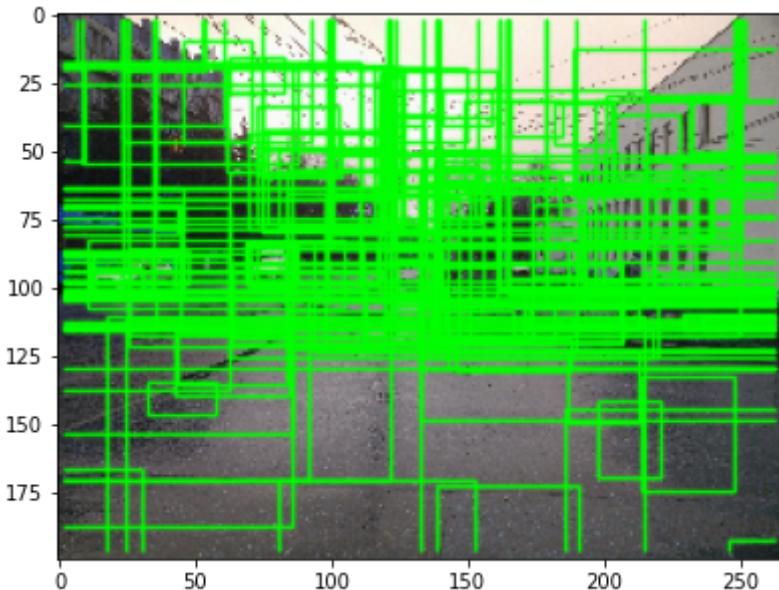
num_Rects = 200 #число выводимых region proposals (прямоугольников)

#создаем копию исходной картинки
im_ = im.copy()

for i, rect in enumerate(rects):
    if (i < num_Rects): #пока не больше
        # x,y - координаты левого верхнего угла; w, h - ширина и высота области
        x, y, w, h = rect
        # рисуем рамку
        cv.rectangle(im_, (x, y), (x+w, y+h), (0, 255, 0), 1, cv.LINE_AA)
    else:
        break

# Рисуем на экран
plt.figure(figsize=(16,5))
plt.imshow(im_)
plt.show()

```



Запишем маршрут, где есть что-нибудь для чтения (последовательность из нескольких фотографий)

```
#change dir to your project folder
root_path = '/content/drive/My Drive/Colab Notebooks/data/'
```

▼ Формирование и обработка результатов детектирования

Запишем основные функции из примера TensorFlow

(https://github.com/tensorflow/hub/blob/master/examples/colab/object_detection.ipynb) и адаптируем их для своих целей

```
# выводит картинку
```

<https://colab.research.google.com/drive/1cv7JEiorpDHZ4Uht-s5XUtR92fbLOMrM#scrollTo=DbG0NgdwzFt9&printMode=true>

```
def display_image(image):
    fig = plt.figure(figsize=(20, 15))
    plt.grid(False)
    plt.imshow(image)

# Грузим и обрабатываем картинку
def download_and_resize_image(url, new_width=256, new_height=256,
                               display=False):
    _, filename = tempfile.mkstemp(suffix=".jpg")
    response = urlopen(url)
    image_data = response.read()
    #read image
    image_data = BytesIO(image_data)
    pil_image = Image.open(image_data)
    pil_image = ImageOps.fit(pil_image, (new_width, new_height), Image.ANTIALIAS)
    pil_image_rgb = pil_image.convert("RGB")
    pil_image_rgb.save(filename, format="JPEG", quality=90)
    #print("Image downloaded to %s." % filename)
    if display:
        display_image(pil_image)
    return filename

# Рисуем рамку на изображении
def draw_bounding_box_on_image(image,
                               ymin,
                               xmin,
                               ymax,
                               xmax,
                               color,
                               font,
                               thickness=4,
                               display_str_list=()):
    """Adds a bounding box to an image."""
    draw = ImageDraw.Draw(image)
    im_width, im_height = image.size
    (left, right, top, bottom) = (xmin * im_width, xmax * im_width,
                                  ymin * im_height, ymax * im_height)
    draw.line([(left, top), (left, bottom), (right, bottom), (right, top),
               (left, top)],
              width=thickness,
              fill=color)

# If the total height of the display strings added to the top of the bounding
# box exceeds the top of the image, stack the strings below the bounding box
# instead of above.
display_str_heights = [font.getsize(ds)[1] for ds in display_str_list]
# Each display_str has a top and bottom margin of 0.05x.
total_display_str_height = (1 + 2 * 0.05) * sum(display_str_heights)

if top > total_display_str_height:
    text_bottom = top
else:
    text_bottom = bottom + total_display_str_height
# Reverse list and print from bottom to top.
for display_str in display_str_list[::-1]:
```

```

text_width, text_height = font.getsize(display_str)
margin = np.ceil(0.05 * text_height)
draw.rectangle([(left, text_bottom - text_height - 2 * margin),
                (left + text_width, text_bottom)],
               fill=color)
draw.text((left + margin, text_bottom - text_height - margin),
          display_str,
          fill="black",
          font=font)
text_bottom -= text_height - 2 * margin

# Рисуем все рамки
# max_boxes - максимальное число объектов для отображения, min_score=0.01 - минимакльная т
# можно менять

def draw_boxes(image, boxes, class_names, scores, max_boxes=3, min_score=0.01):
    """Overlay labeled boxes on an image with formatted scores and label names."""
    colors = list(ImageColor.colormap.values())

try:
    font = ImageFont.truetype("/usr/share/fonts/truetype/liberation/LiberationSansNarrow-R
                               25")
except IOError:
    print("Font not found, using default font.")
    font = ImageFont.load_default()

for i in range(min(boxes.shape[0], max_boxes)):
    if scores[i] >= min_score:
        ymin, xmin, ymax, xmax = tuple(boxes[i])
        display_str = "{}: {}".format(class_names[i].decode("ascii"),
                                       int(100 * scores[i]))
        color = colors[hash(class_names[i]) % len(colors)]
        image_pil = Image.fromarray(np.uint8(image)).convert("RGB")
        draw_bounding_box_on_image(
            image_pil,
            ymin,
            xmin,
            ymax,
            xmax,
            color,
            font,
            display_str_list=[display_str])
        np.copyto(image, np.array(image_pil))
return image

```

Загрузка детектора

```

# справа в списке выбора можно поменять архитектуру на SSD или F
module_handle = "https://tfhub.dev/google/openimages_v4/ssd/mobilenet_v2/1" #@param ["http
detector = hub.load(module_handle).signatures['default']

```

INFO:tensorflow:Saver not created because there are no variables in the graph to rest

INFO:tensorflow:Saver not created because there are no variables in the graph to rest

Формируем тензор

```
def load_img(path):
    img = tf.io.read_file(path)
    img = tf.image.decode_jpeg(img, channels=3) # алгоритм кодирования JPEG
    return img
```

Выполняем детектор на 10 рамок

```
max_frames = 10
def run_detector(detector, path):
    img = load_img(path)
    # превращаем изображение из целого в вещественное
    converted_img = tf.image.convert_image_dtype(img, tf.float32)[tf.newaxis, ...]
    start_time = time.time()
    result = detector(converted_img)
    end_time = time.time()
    #print(result['detection_scores'])
    result = {key:value.numpy() for key,value in result.items()}

    # визуализация. Ограничиваем количество рамок 100
    image_with_boxes = draw_boxes(
        img.numpy(), result["detection_boxes"],
        result["detection_class_entities"], result["detection_scores"], max_boxes=max_frames)

    display_image(image_with_boxes)
    return result

# Найдем рамку
# max_boxes - максимальное число объектов для отображения, min_score=0.0 - минимакльная то
# можно менять
def get_bounding_box_on_image(image, boxes, class_names, scores, max_boxes=max_frames, min_
    """Overlay labeled boxes on an image with formatted scores and label names."""
if len(image.shape)>3:
    image=image.reshape(image.shape[-3],image.shape[-2],image.shape[-1])
I_shape=np.uint8(image).shape
#print(I_shape)
im_array=[]
for i in range(min(boxes.shape[0], max_boxes)):
    #print(tuple(boxes[i]))
    if scores[i] >= min_score:
        ymin, xmin, ymax, xmax = tuple(boxes[i])
```

```
        yminI = np.int(ymin*I_shape[0])
        ymaxI = np.int(ymax*I_shape[0])
        xminI = np.int(xmin*I_shape[1])
        xmaxI = np.int(xmax*I_shape[1])
        im_array.append(np.array([yminI, xminI, ymaxI, xmaxI]))
```

```
imf = np.array(image)[ymin:ymax, xmin:xmax, :]
#print(imf.shape, imf[0,0,:])

im_array.append(imf.copy())
return im_array

# Вернем рамки
def get_boxes(image, boxes, class_names, scores, max_boxes=max_frames, min_score=0.1):
    """Overlay labeled boxes on an image with formatted scores and label names."""

    image_pil = np.array(image)
    for i in range(min(boxes.shape[0], max_boxes)):
        if scores[i] >= min_score:
            ymin, xmin, ymax, xmax = tuple(boxes[i])

    imf= get_bounding_box_on_image(image_pil, boxes, class_names, scores, max_boxes=max_

return imf

import os, fnmatch
listOfFiles = os.listdir(root_path)
pattern = '*.jpg'
List_name=[]
for entry in listOfFiles:
    if fnmatch.fnmatch(entry, pattern):
        List_name.append(np.array(root_path+entry))

image_to_process = List_name[6]
```

Проверяем детектор

```
result_10 = run_detector(detector, image_to_process)
```



машины, деревья, здания, но нет людей

▼ Выполняем детектор на 100 рамок

```
max_frames = 100
def run_detector(detector, path):
    img = load_img(path)
    # превращаем изображение из целого в вещественное
    converted_img = tf.image.convert_image_dtype(img, tf.float32)[tf.newaxis, ...]
    start_time = time.time()
    result = detector(converted_img)
    end_time = time.time()
    #print(result['detection_scores'])
    result = {key:value.numpy() for key,value in result.items()}

    # визуализация. Ограничиваем количество рамок 10
    image_with_boxes = draw_boxes(
        img.numpy(), result["detection_boxes"],
        result["detection_class_entities"], result["detection_scores"], max_boxes=max_frames

    display_image(image_with_boxes)
    return result

# Найдем рамку
# max_boxes - максимальное число объектов для отображения, min_score=0.0 - минимакльная то
```

<https://colab.research.google.com/drive/1cv7JEiorpDHZ4Uht-s5XUtR92fbLOMrM#scrollTo=DbG0NgdwzFt9&printMode=true>

```
# можно менять
def get_bounding_box_on_image(image, boxes, class_names, scores, max_boxes=max_frames, min_
    """Overlay labeled boxes on an image with formatted scores and label names."""
    if len(image.shape)>3:
        image=image.reshape(image.shape[-3],image.shape[-2],image.shape[-1])
    I_shape=np.uint8(image).shape
    #print(I_shape)
    im_array=[]
    for i in range(min(boxes.shape[0], max_boxes)):
        #print(tuple(boxes[i]))
        if scores[i] >= min_score:
            ymin, xmin, ymax, xmax = tuple(boxes[i])

            yminI = np.int(ymin*I_shape[0])
            ymaxI = np.int(ymax*I_shape[0])
            xminI = np.int(xmin*I_shape[1])
            xmaxI = np.int(xmax*I_shape[1])
            imf = np.array(image)[yminI:ymaxI,xminI:xmaxI,:]
            #print(imf.shape,imf[0,0,:])

            im_array.append(imf.copy())
    return im_array

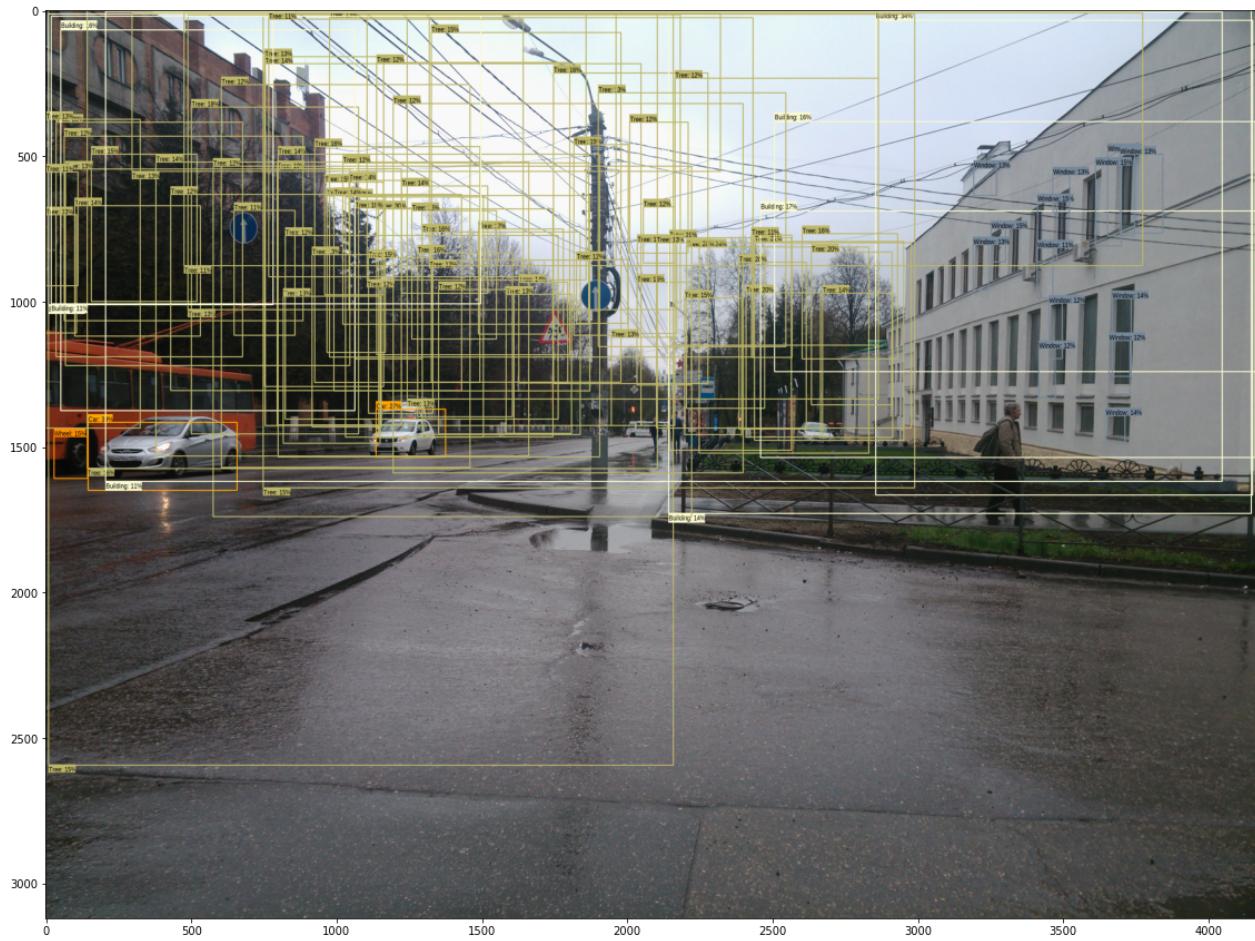
# Вернем рамки
def get_boxes(image, boxes, class_names, scores, max_boxes=max_frames, min_score=0.1):
    """Overlay labeled boxes on an image with formatted scores and label names."""

    image_pil = np.array(image)
    for i in range(min(boxes.shape[0], max_boxes)):
        if scores[i] >= min_score:
            ymin, xmin, ymax, xmax = tuple(boxes[i])

            imf= get_bounding_box_on_image(image_pil, boxes, class_names, scores, max_boxes=max_

    return imf

result_100 = run_detector(detector, image_to_process)
```



Добавились окна и даже колесо, но людей по-прежнему нет. И одна машина теперь принята за здание

✓ 20 сек. выполнено в 02:29

