Summer Bootcamp 2021

Applied Machine Learning
Classification: Project #4

Ngan Le

thile@uark.edu

Identifying Cards in a Video

- Input: video and a pre-trained model
- Output: Bounding box around the cars in each frames

Video: Frame by Frame

Image Is too big?

- RGB Image
- 12 megapixel image has ? features.
- How to reduce?

Image with PIL and OpenCV

Image Processing with PIL

Import

```
from PIL import Image
```

Create

```
img = Image.new(‘RBG’, (600,400), ‘yellow’)
```

Open

```
img = Image.open(‘existing.png’)
```

Save

```
img.save(‘myimage.png’)
```

Show

```
img.show()
```

Resize

```
img.resize((100,100), Image.ANTIALIAS)
```

Blur (ImageFilter)

```
img.filter.(ImageFilter.BLUR)
```

Blend 2 images together

```
img = Image.blend(Image.open(‘image1.png’,’image2.png’, 0.5))
```

Write a text

```
draw = ImageDraw.Draw(img)
draw.text(0,0,’This text goes on top of the image’)
```

Image Processing with OpenCV

Import

```
import cv2
```

Read an image

```
image = cv2.imread("./Path/To/Image.extension")
```

Show an image

```
cv2.imshow("Image", image)
```

Change colorspace

```
rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Rotate

```
(h, w, d) = image.shape
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, 180, 1.0)
rotated = cv2.warpAffine(image, M, (w, h))
```

Blurring

```
blurred = cv2.GaussianBlur(image, (51, 51), 0)
```

Drawing a bounding box/line

```
output = image.copy()
cv2.rectangle/line(output, (2600, 800), (4100, 2400), (0, 255, 255), 10)
```

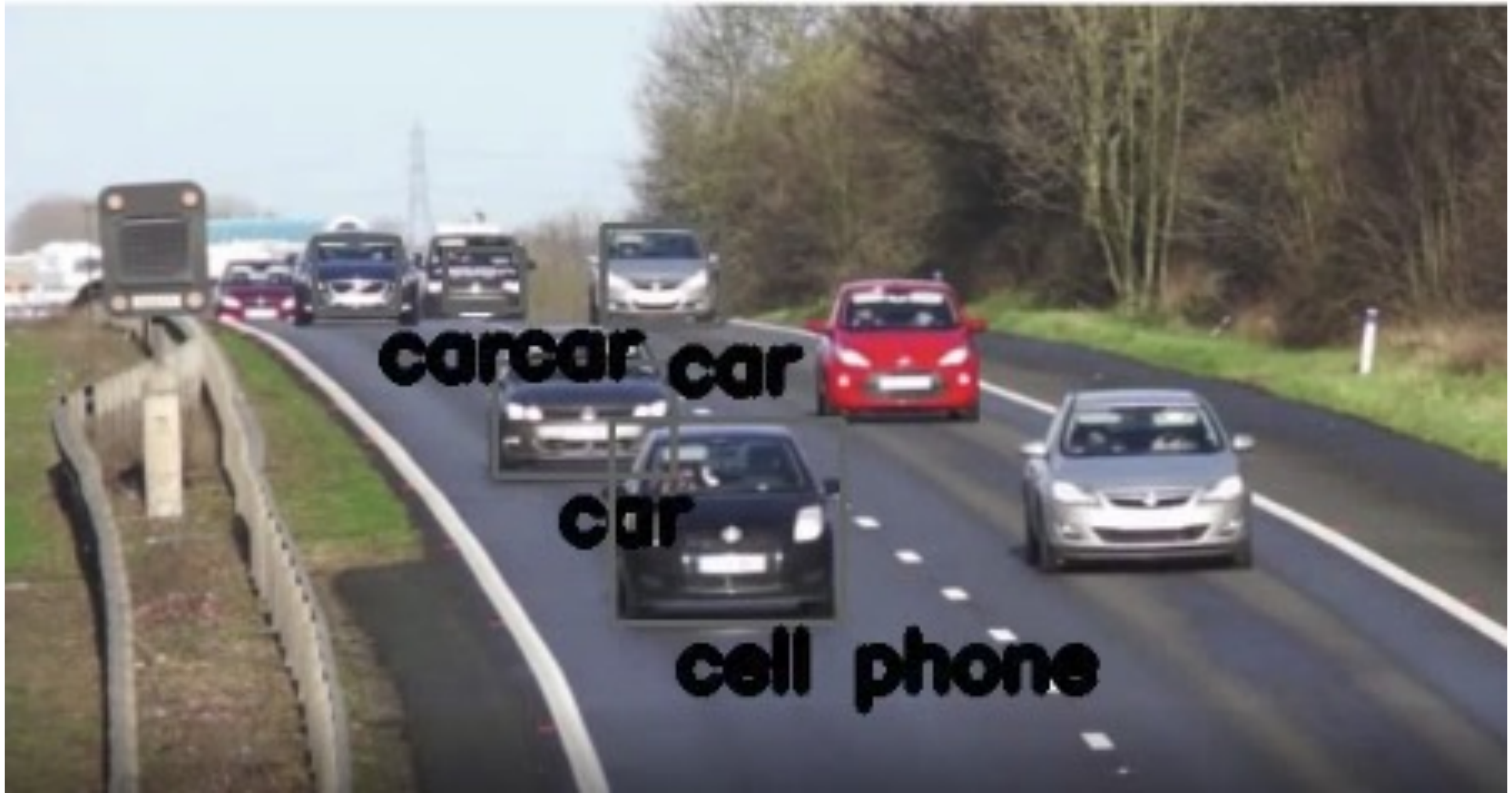
Write a text

```
output = image.copy()
cv2.putText(output, ”Car", (1500, 3600),cv2.FONT_HERSHEY_SIMPLEX, 15, (30, 105, 210), 40)
```

Image Processing

- PIL
- OpenCV
- scikit-image
- Pillow

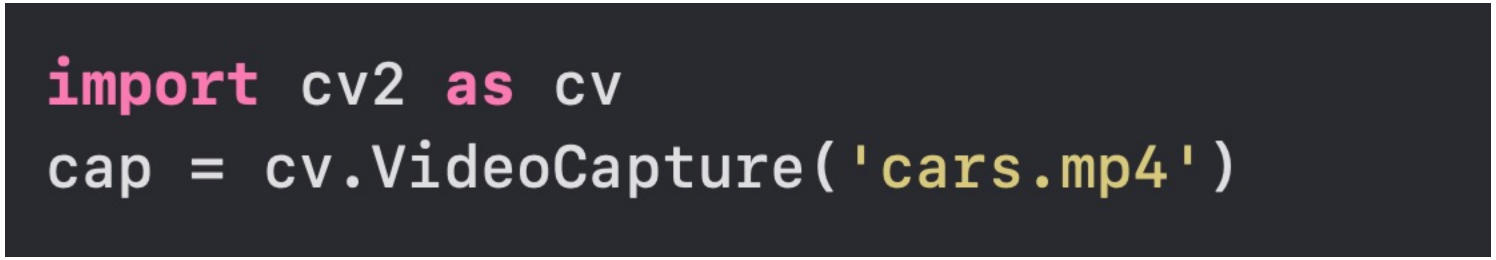
Project



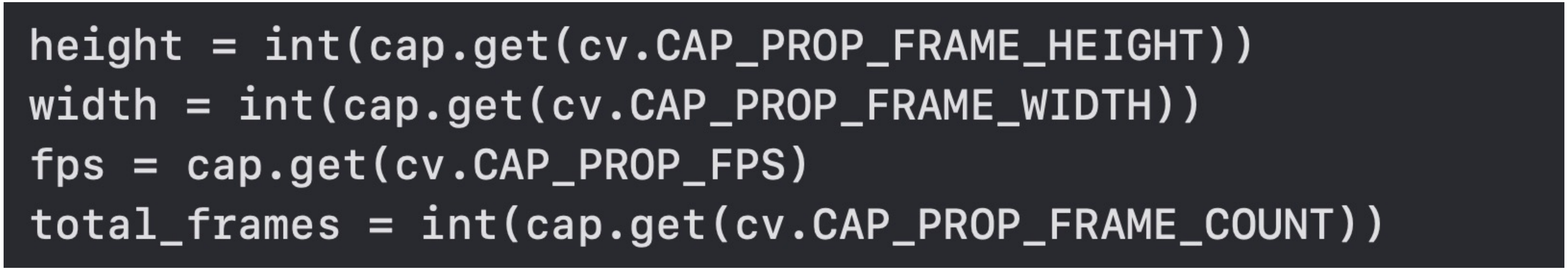
Your program should:

- Read in a video file
- Load the TensorFlow model
- Loop over each frame of the video
- Scale the frame down to a size the model expects
- Feed the frame to the model
- Loop over detections made by the model
- If the detection score is above some threshold, draw a bounding box onto the frame and put a label in or near the box
- Write the frame back to a new video

- Load a video & get video properties:

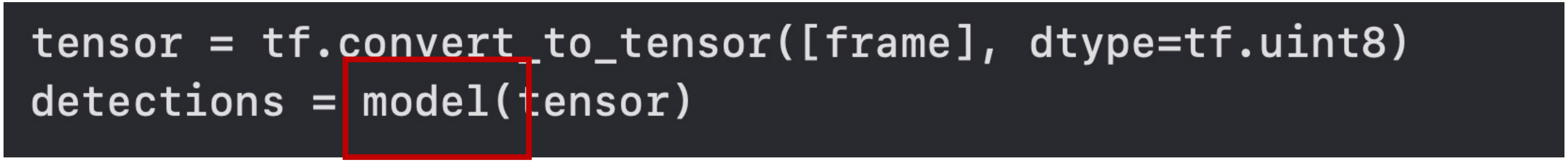


```python
import cv2 as cv
cap = cv.VideoCapture('cars.mp4')
```



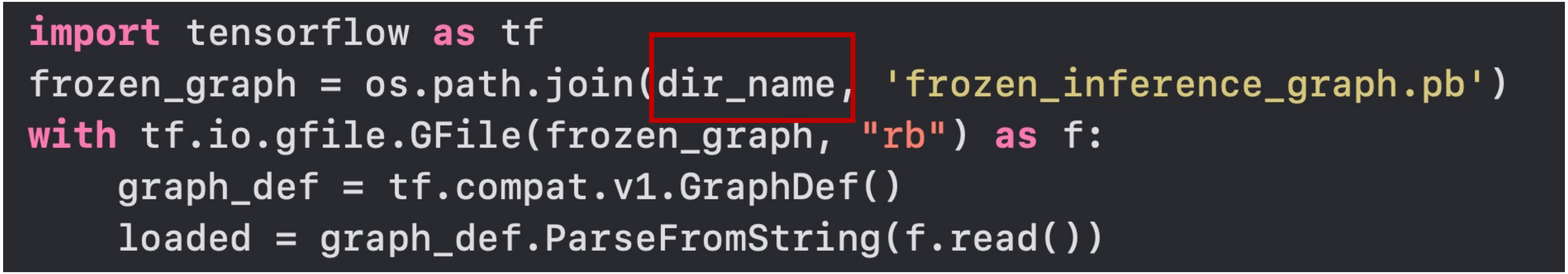
```python
height = int(cap.get(cv.CAP_PROP_FRAME_HEIGHT))
width = int(cap.get(cv.CAP_PROP_FRAME_WIDTH))
fps = cap.get(cv.CAP_PROP_FPS)
total_frames = int(cap.get(cv.CAP_PROP_FRAME_COUNT))
```

- For each frame:

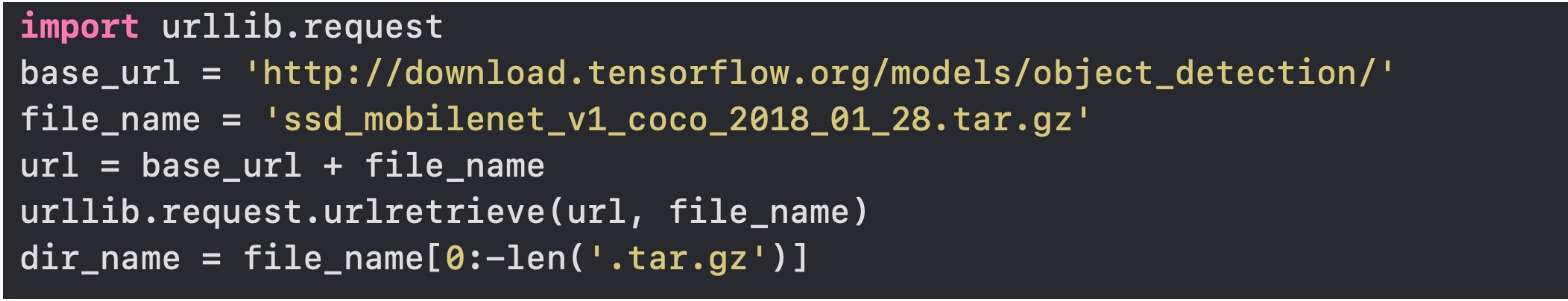


```
tensor = tf.convert_to_tensor([frame], dtype=tf.uint8)
detections = model(tensor)
```

Load model & wrap

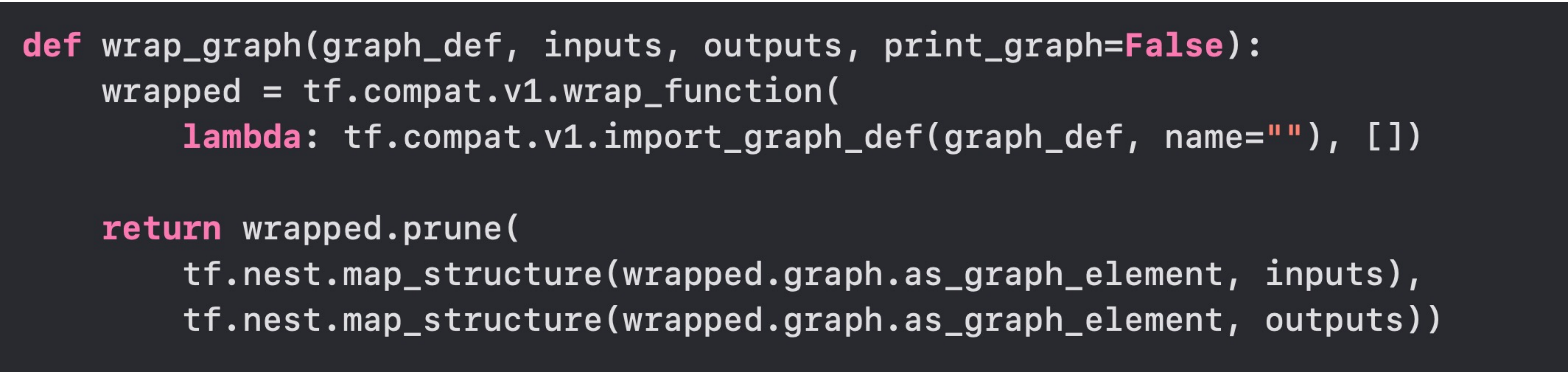


```python
import tensorflow as tf
frozen_graph = os.path.join(dir_name, 'frozen_inference_graph.pb')
with tf.io.gfile.GFile(frozen_graph, "rb") as f:
    graph_def = tf.compat.v1.GraphDef()
    loaded = graph_def.ParseFromString(f.read())
```



```python
import urllib.request
base_url = 'http://download.tensorflow.org/models/object_detection/'
file_name = 'ssd_mobilenet_v1_coco_2018_01_28.tar.gz'
url = base_url + file_name
urllib.request.urlretrieve(url, file_name)
dir_name = file_name[0:-len('.tar.gz')]
```

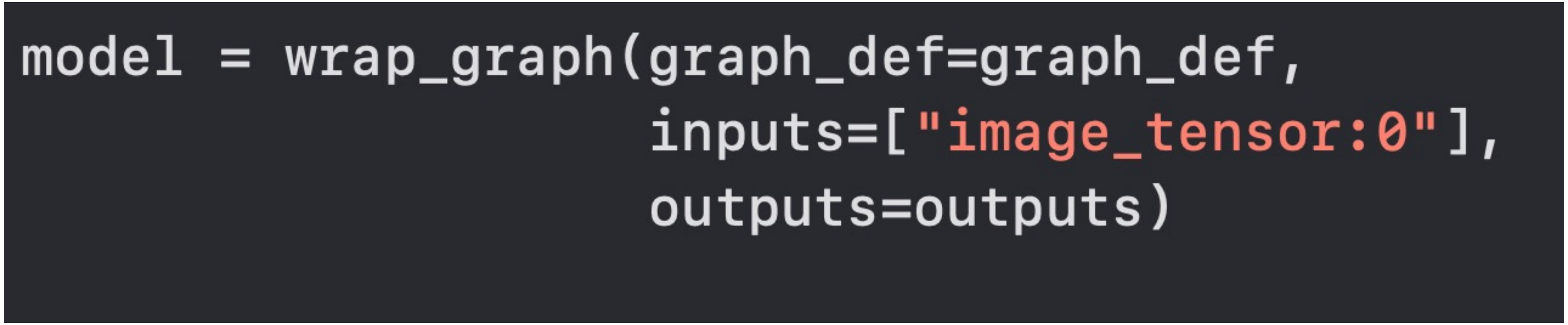
Load model & warp



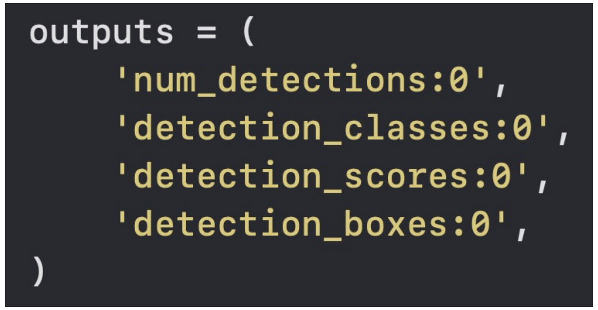
```python
def wrap_graph(graph_def, inputs, outputs, print_graph=False):
    wrapped = tf.compat.v1.wrap_function(
        lambda: tf.compat.v1.import_graph_def(graph_def, name=""), [])

    return wrapped.prune(
        tf.nest.map_structure(wrapped.graph.as_graph_element, inputs),
        tf.nest.map_structure(wrapped.graph.as_graph_element, outputs))
```

Load model & warp

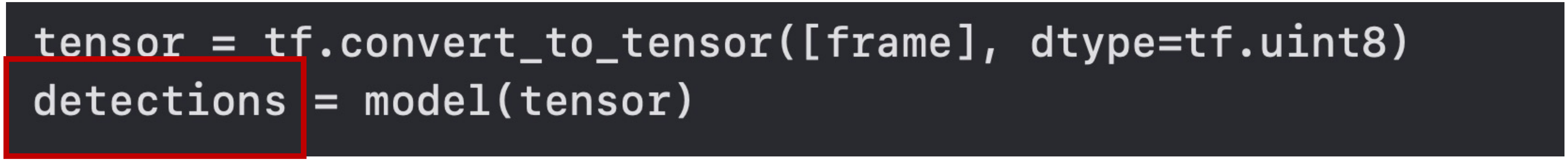


```
model = wrap_graph(graph_def=graph_def,
                   inputs=["image_tensor:0"],
                   outputs=outputs)
```



```
outputs = (
    'num_detections:0',
    'detection_classes:0',
    'detection_scores:0',
    'detection_boxes:0',
)
```

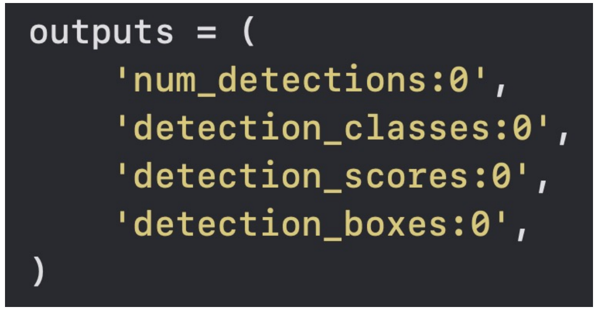
- For each frame:



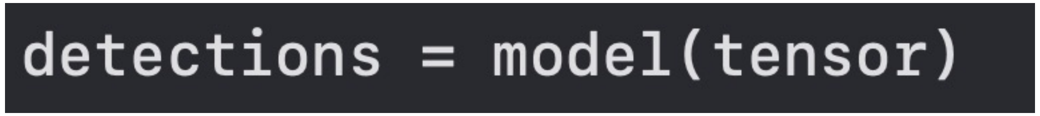
```
tensor = tf.convert_to_tensor([frame], dtype=tf.uint8)
detections = model(tensor)
```

Visualize the results & Compare

- adds bounding boxes and labels to the detected objects in a frame.



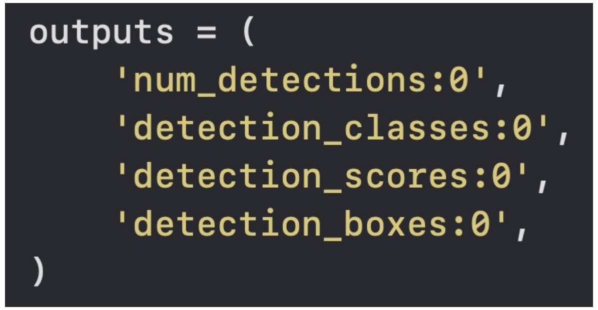
```
outputs = (
    'num_detections:0',
    'detection_classes:0',
    'detection_scores:0',
    'detection_boxes:0',
)
```



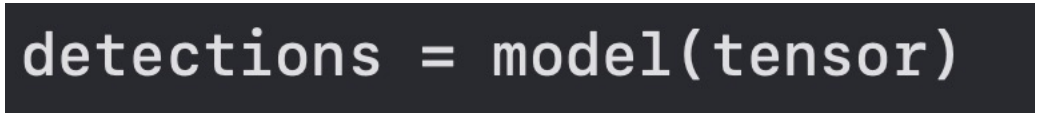
```
detections = model(tensor)
```

Visualize the results & Compare

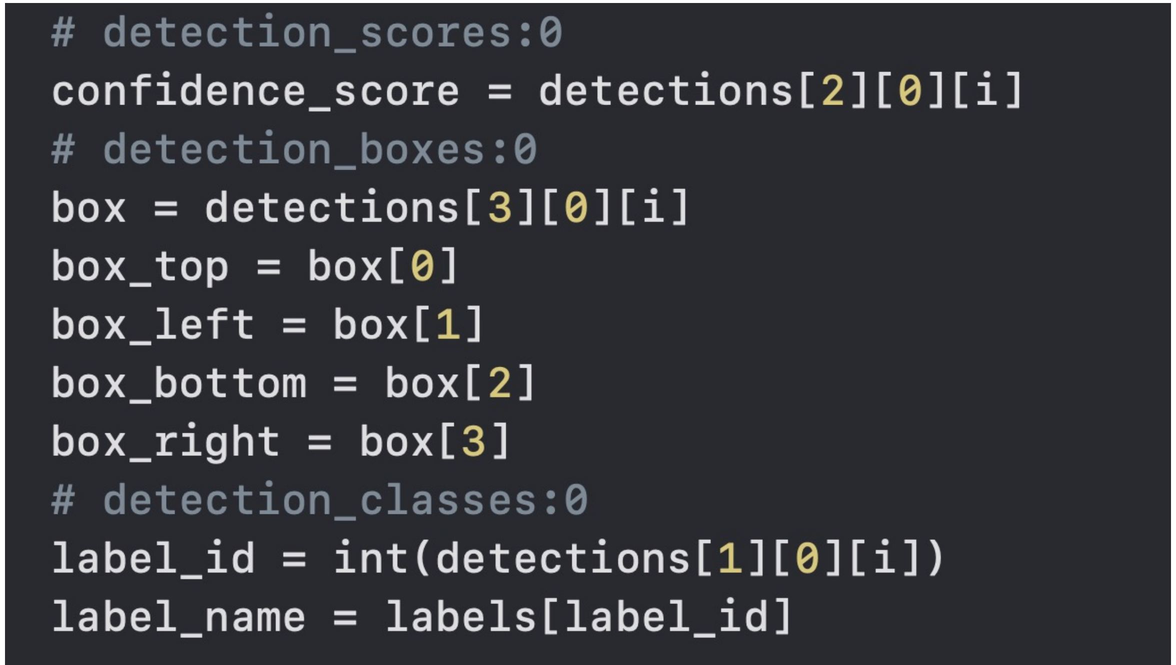
- adds bounding boxes and labels to the detected objects in a frame.



```
outputs = (
    'num_detections:0',
    'detection_classes:0',
    'detection_scores:0',
    'detection_boxes:0',
)
```



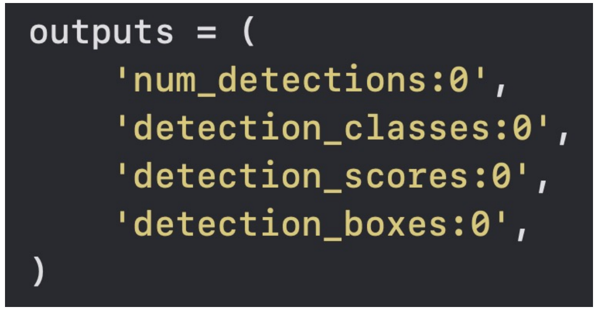
```
detections = model(tensor)
```



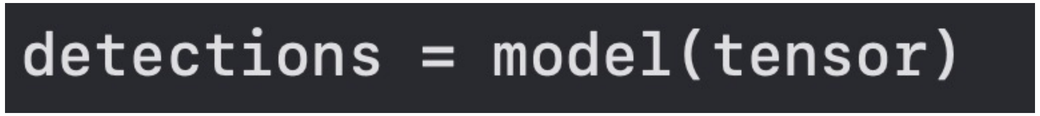
```
# detection_scores:0
confidence_score = detections[2][0][i]
# detection_boxes:0
box = detections[3][0][i]
box_top = box[0]
box_left = box[1]
box_bottom = box[2]
box_right = box[3]
# detection_classes:0
label_id = int(detections[1][0][i])
label_name = labels[label_id]
```

Visualize the results & Compare

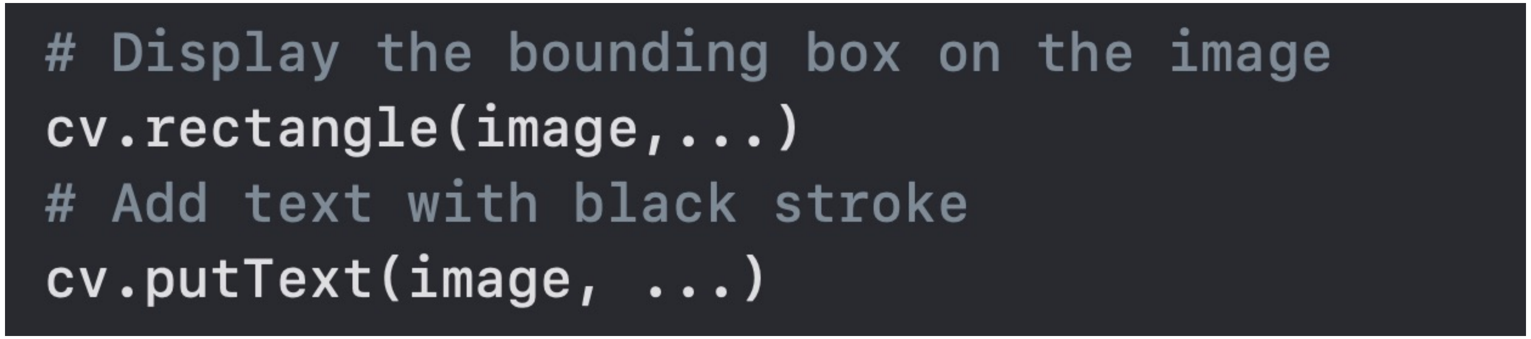
- adds bounding boxes and labels to the detected objects in a frame.



```
outputs = (
    'num_detections:0',
    'detection_classes:0',
    'detection_scores:0',
    'detection_boxes:0',
)
```



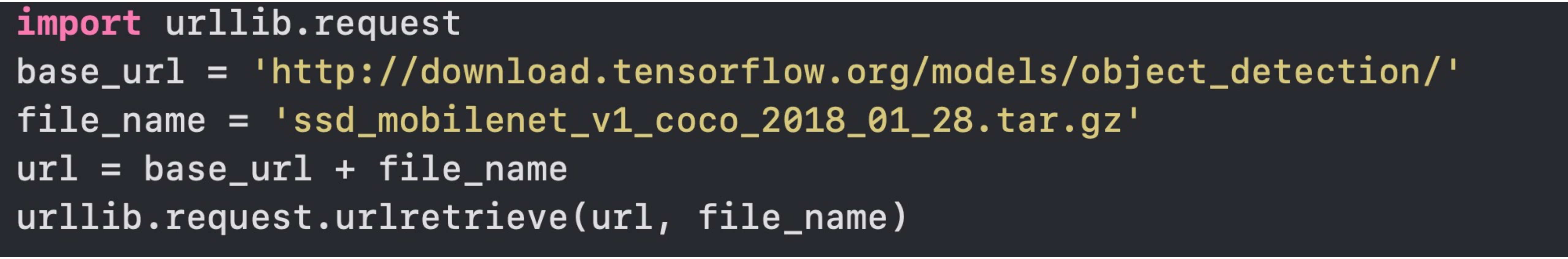
```
detections = model(tensor)
```



```
# Display the bounding box on the image
cv.rectangle(image,...)
# Add text with black stroke
cv.putText(image, ...)
```

Prepare data & model

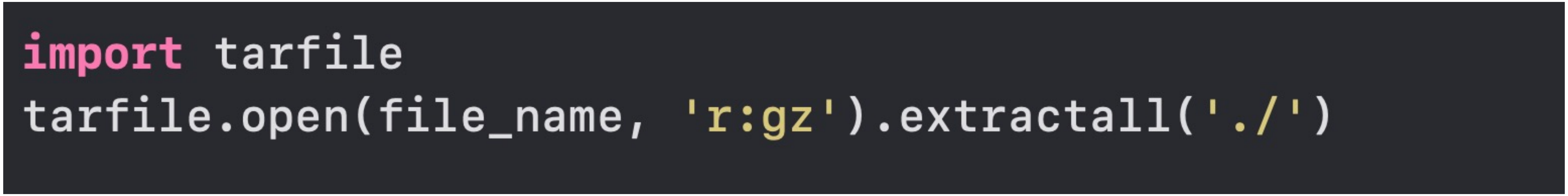
- Download model file



```python
import urllib.request
base_url = 'http://download.tensorflow.org/models/object_detection/'
file_name = 'ssd_mobilenet_v1_coco_2018_01_28.tar.gz'
url = base_url + file_name
urllib.request.urlretrieve(url, file_name)
```

Prepare data & model

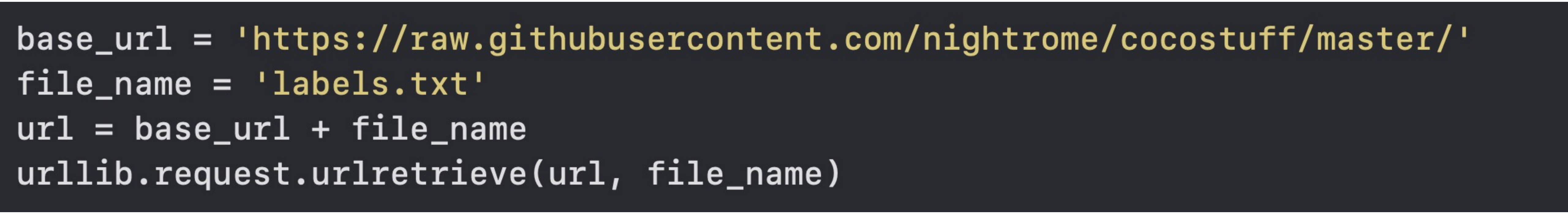
- Unzip a file



```python
import tarfile
tarfile.open(file_name, 'r:gz').extractall('./')
```

Prepare data & model

- Download annotation



```
base_url = 'https://raw.githubusercontent.com/nightrome/cocostuff/master/'
file_name = 'labels.txt'
url = base_url + file_name
urllib.request.urlretrieve(url, file_name)
```