LECTURE SERIES FOR DIGITAL SURVEILLANCE SYSTEMS AND APPLICATION

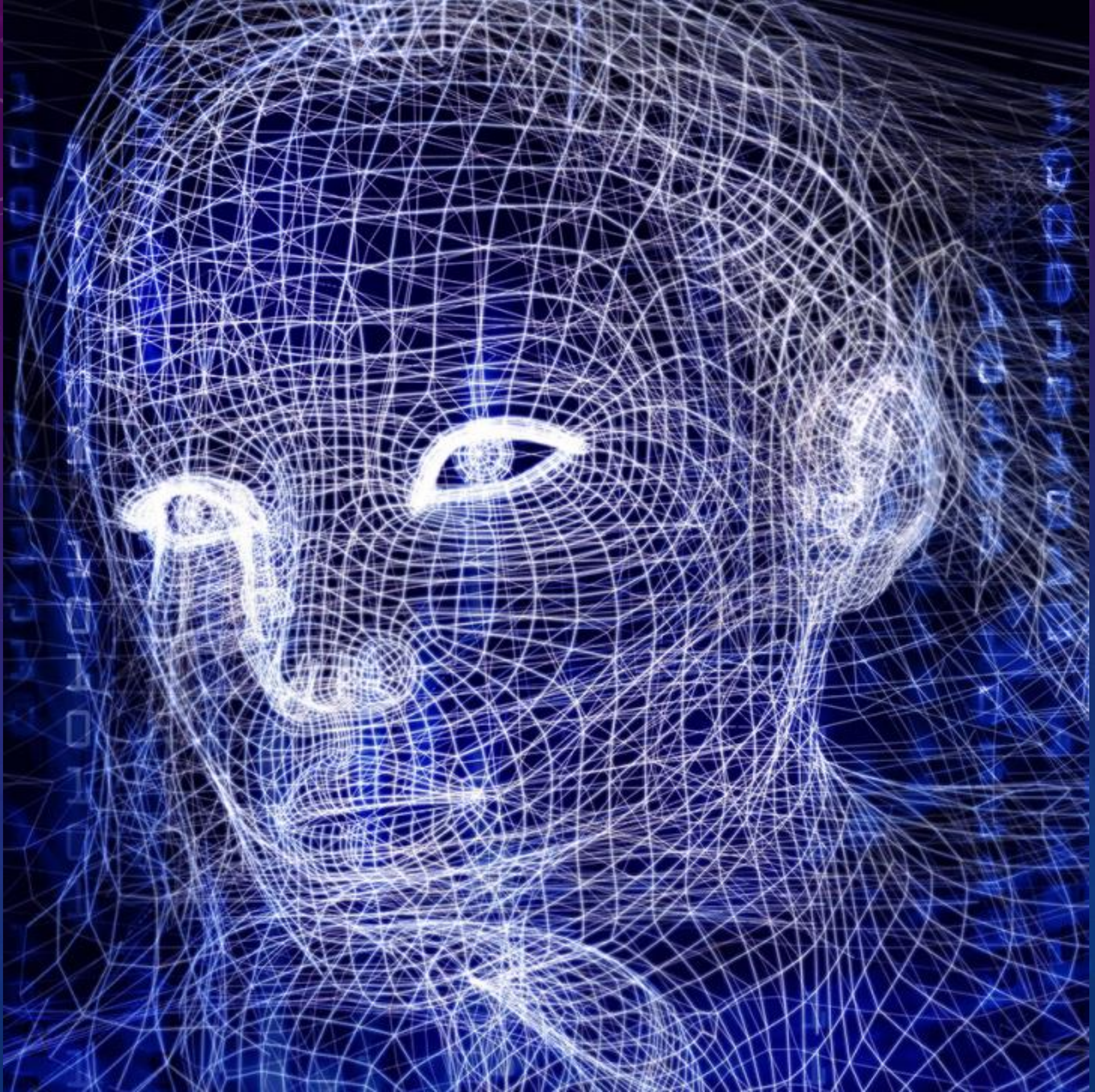
Chapter 6

Coding Manual

徐繼聖

Gee-Sern Jison Hsu

National Taiwan University of Science and Technology



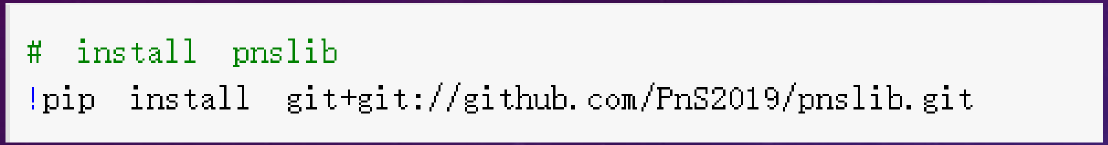
Face Detection

Example 6-1 Face Detection

- This example will show the commonly used basic package OpenCV and the MTCNN of the deep learning method to compare face detection.

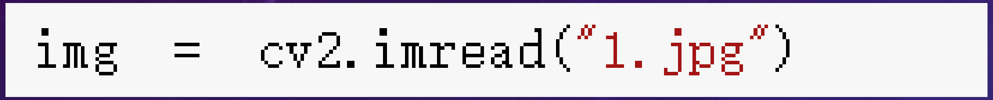
Example 6-1 Face Detection_OpenCV

- Prepare the code and install



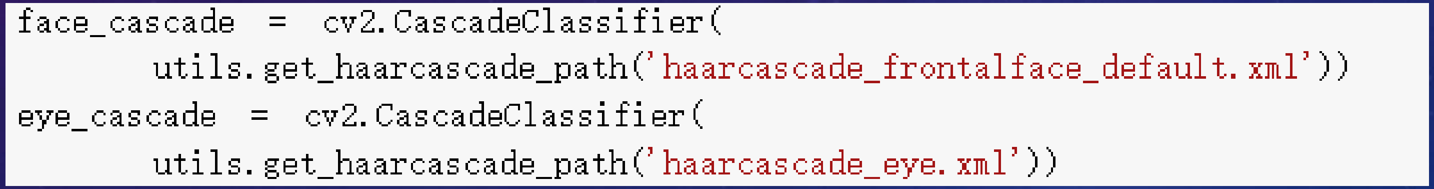
```
# install pnslib
!pip install git+git://github.com/PnS2019/pnslib.git
```

- Read image



```
img = cv2.imread(“1.jpg”)
```

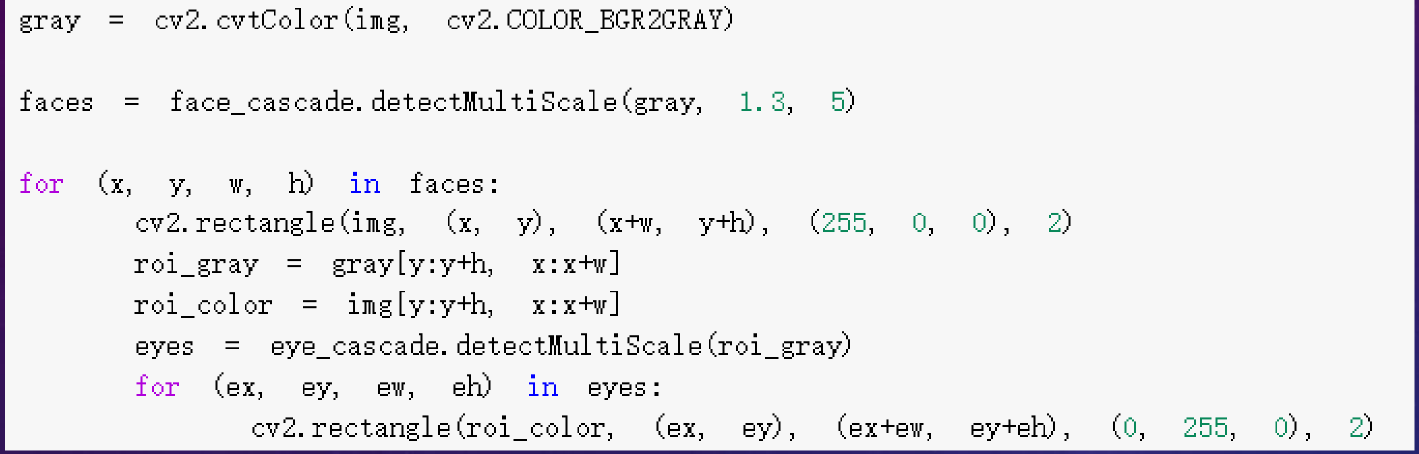
- Load face cascade and eye cascade



```
face_cascade = cv2.CascadeClassifier(
       utils.get_haarcascade_path(’haarcascade_frontalface_default.xml’))
eye_cascade = cv2.CascadeClassifier(
       utils.get_haarcascade_path(’haarcascade_eye.xml’))
```

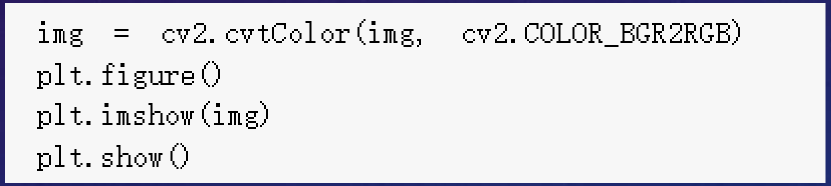
Example 6-1 Face Detection_OpenCV

- Search face



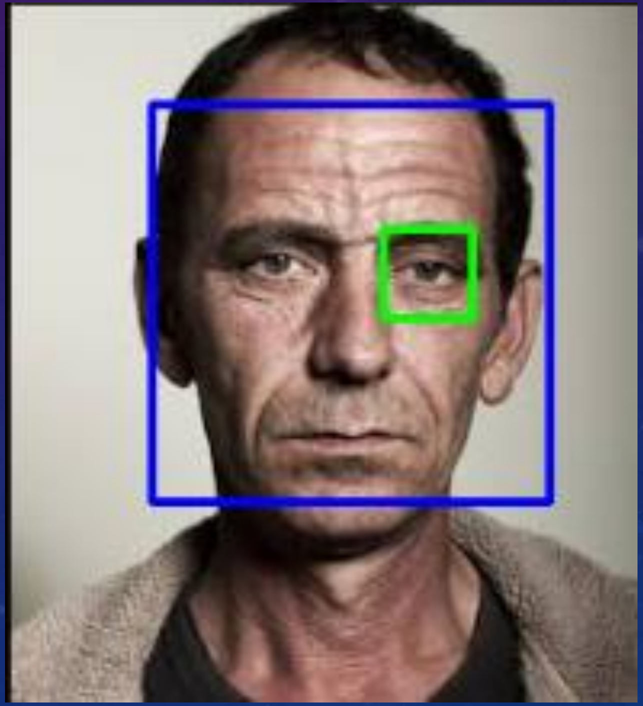
```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade.detectMultiScale(gray, 1.3, 5)

for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
    roi_gray = gray[y:y+h, x:x+w]
    roi_color = img[y:y+h, x:x+w]
    eyes = eye_cascade.detectMultiScale(roi_gray)
    for (ex, ey, ew, eh) in eyes:
        cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
```

- Show image



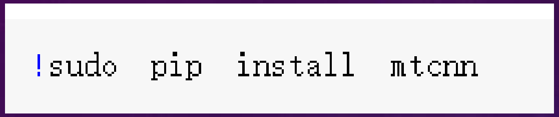
```
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
plt.figure()
plt.imshow(img)
plt.show()
```

- Result



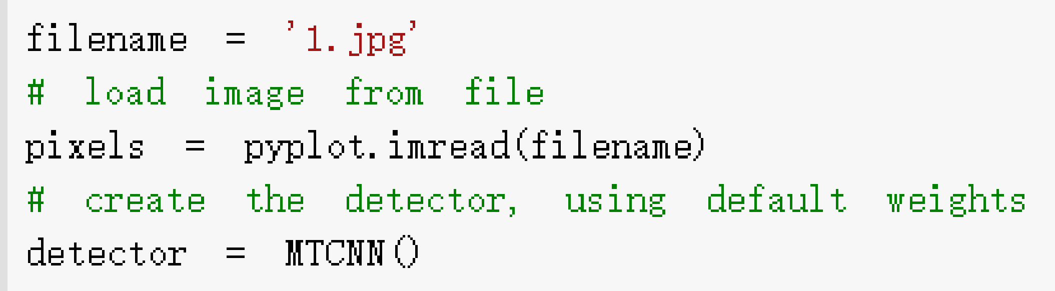
Example 6-1 Face Detection_MTCNN

- Install the MTCNN



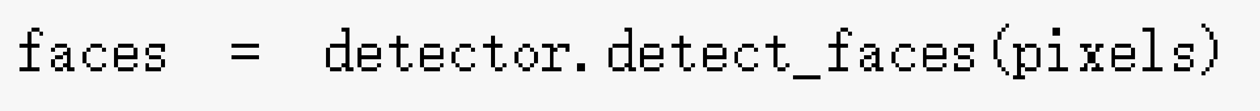
```
!sudo pip install mtcnn
```

- Load image and create detector, using default weights



```
filename = '1.jpg'
# load image from file
pixels = pyplot.imread(filename)
# create the detector, using default weights
detector = MTCNN()
```

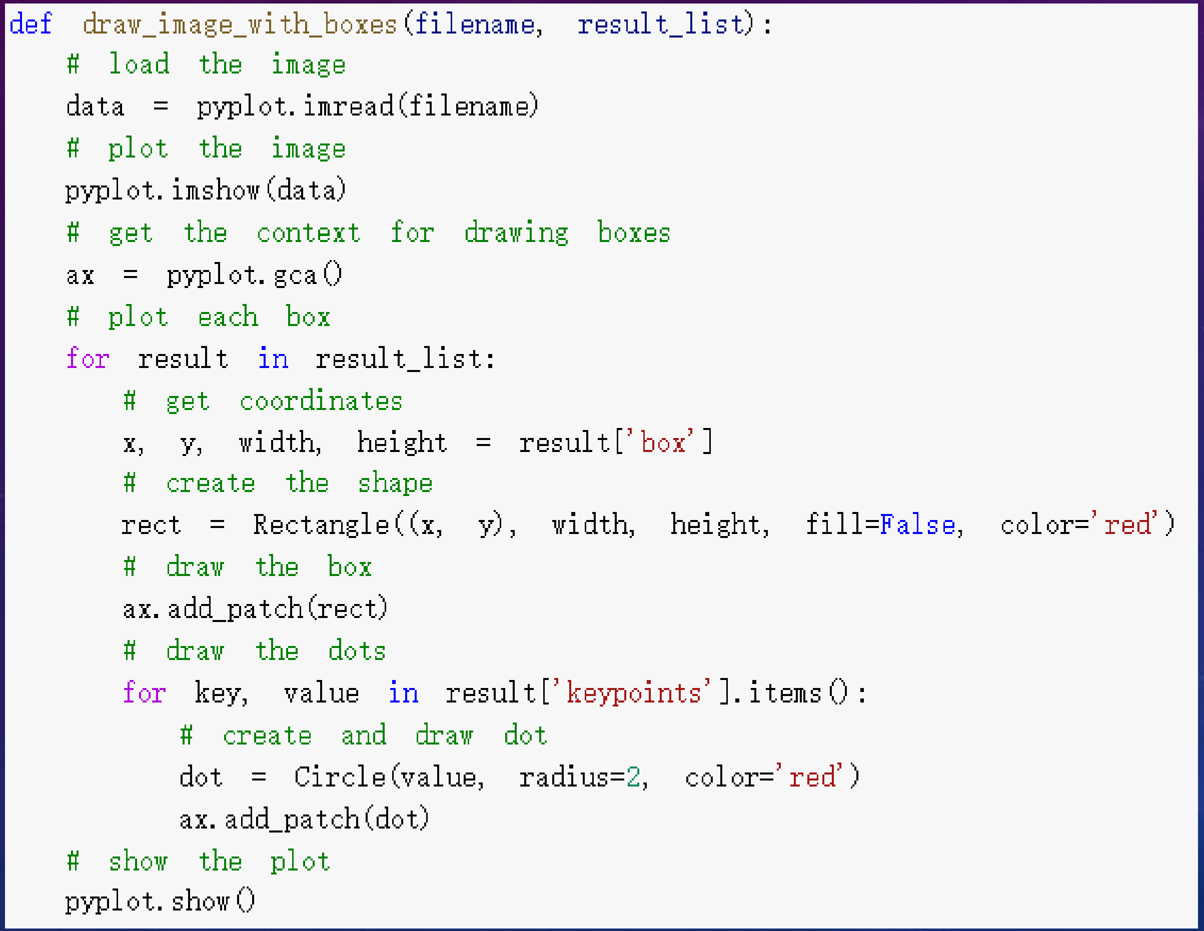
- Detect faces in the image



```
faces = detector.detect_faces(pixels)
```

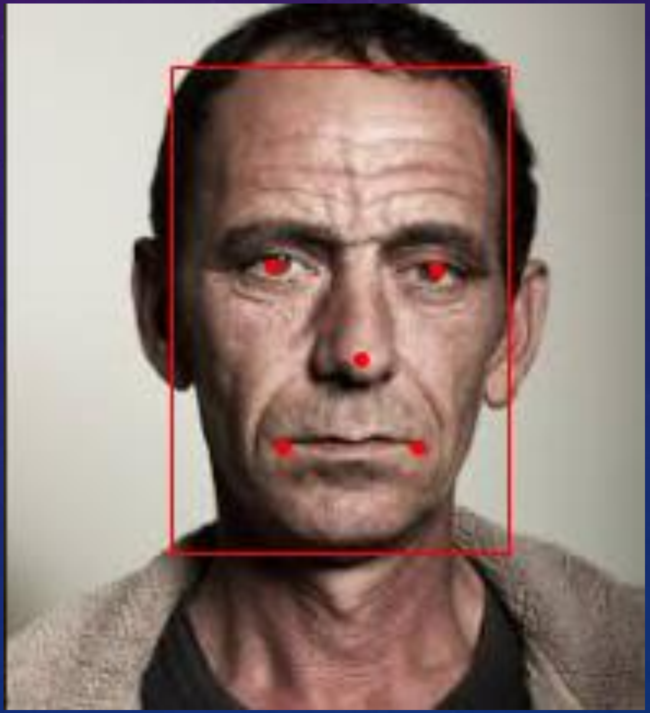
Example 6-1 Face Detection_MTCNN

- Draw an image with detected objects



```
def  draw_image_with_boxes(filename,  result_list):
    #  load  the  image
    data  =  pyplot.imread(filename)
    #  plot  the  image
    pyplot.imshow(data)
    #  get  the  context  for  drawing  boxes
    ax  =  pyplot.gca()
    #  plot  each  box
    for  result  in  result_list:
        #  get  coordinates
        x,  y,  width,  height  =  result[’box’]
        #  create  the  shape
        rect  =  Rectangle((x,  y),  width,  height,  fill=False,  color=’red’)
        #  draw  the  box
        ax.add_patch(rect)
        #  draw  the  dots
        for  key,  value  in  result[’keypoints’].items():
            #  create  and  draw  dot
            dot  =  Circle(value,  radius=2,  color=’red’)
            ax.add_patch(dot)
    #  show  the  plot
    pyplot.show()
```

- Result



Exercise 6-1 – Face Detection

- Please download “face_detection.ipynb” and open “face_detection.ipynb” by Colab.

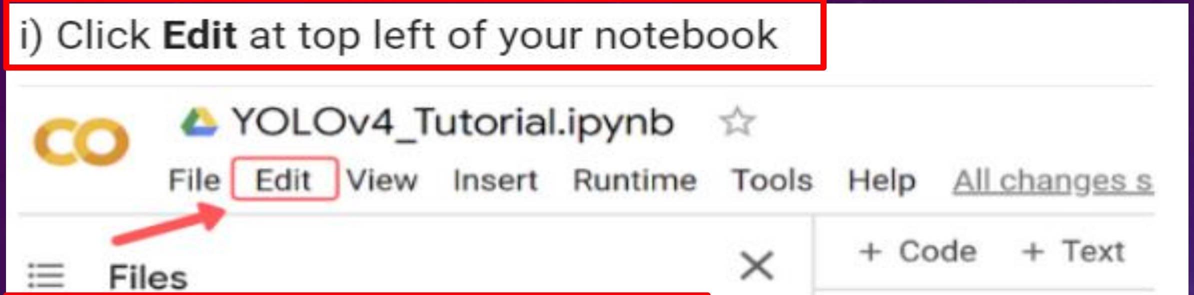
1. Please compare OpenCV and MTCNN and find the face pictures on the Internet by yourself, including a single person and multiple people.

YOLO

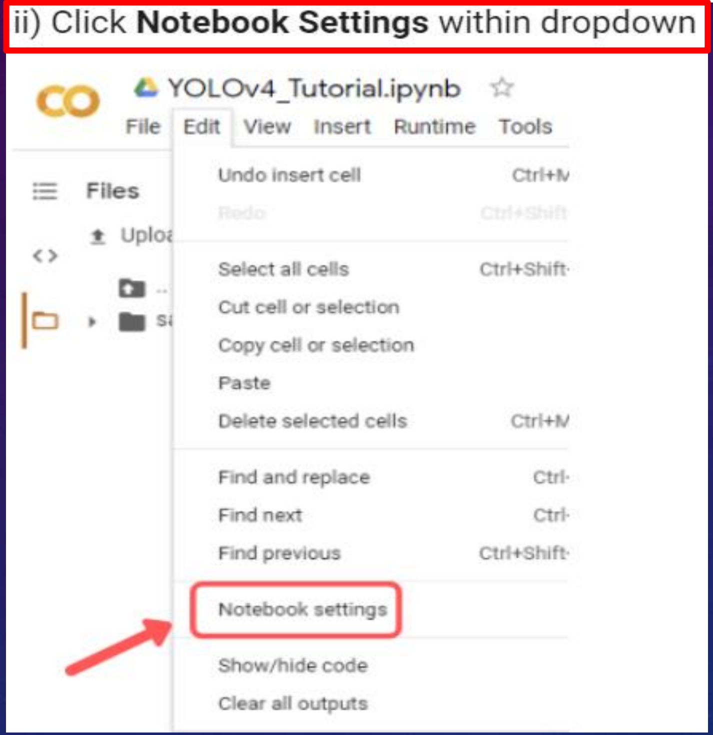
Example 6-2 YOLO

- To evaluate the YOLO v3 tiny, we run an experiment on 2014 COCO_Val dataset_5000images, and the performance can be reported by Table 1. can be used to make the PR (Precision and Recall) curve shown in Figure 1.
- The YOLO v3 tiny is an example, and you need to do the same for v3 and v4.

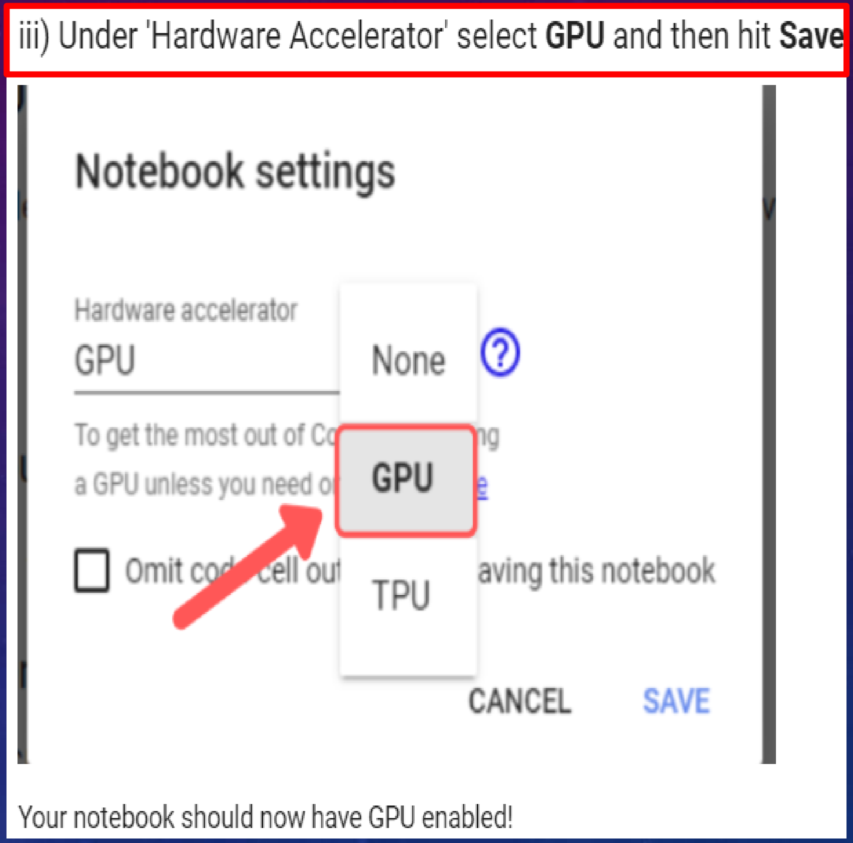
Example 6-2 YOLO – Enabling GPU



i) Click **Edit** at top left of your notebook



ii) Click **Notebook Settings** within dropdown



iii) Under 'Hardware Accelerator' select **GPU** and then hit **Save**

Notebook settings

Hardware accelerator

GPU

To get the most out of Co... ng a GPU unless you need o... e

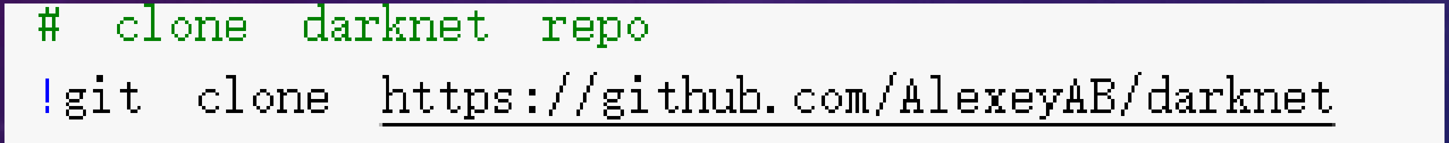
□ Omit cod... cell out... aving this notebook

Your notebook should now have GPU enabled!

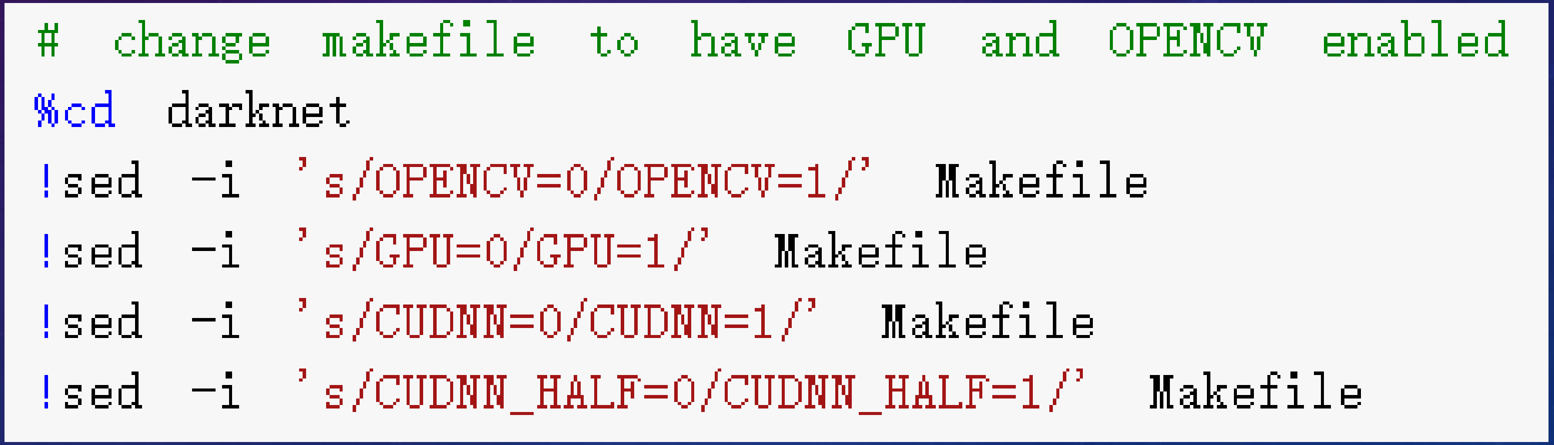
- make sure that pytorch, cuda and the GPU are ready to go

Example 6-2 YOLO – Cloning and Building Darknet

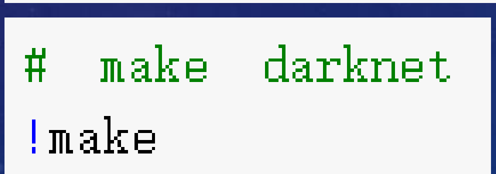
- The following cells will clone the Darknet from AlexeyAB’s famous repository, adjust the Makefile to enable OPENCV and GPU for Darknet and then build Darknet.



```
# clone darknet repo
!git clone https://github.com/AlexeyAB/darknet
```



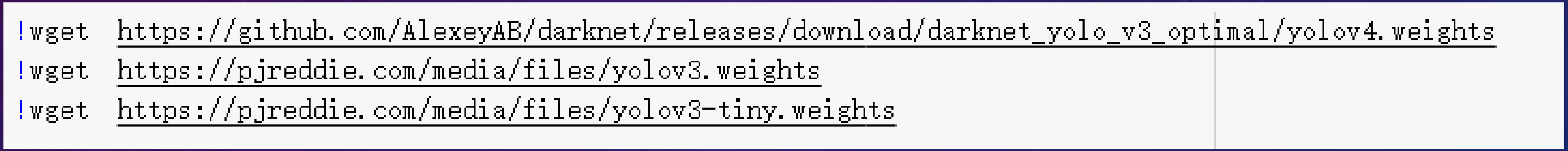
```
# change makefile to have GPU and OPENCV enabled
%cd darknet
!sed -i ’s/OPENCV=0/OPENCV=1/’ Makefile
!sed -i ’s/GPU=0/GPU=1/’ Makefile
!sed -i ’s/CUDNN=0/CUDNN=1/’ Makefile
!sed -i ’s/CUDNN_HALF=0/CUDNN_HALF=1/’ Makefile
```



```
# make darknet
!make
```

Example 6-2 YOLO – Download Pretrained Weights

- YOLOv3, tiny-v3, and v4 have been trained already on the COCO dataset with 80 object classes that they can predict. We can grab these pretrained weights so that we can run YOLO on these pretrained models for object detection.



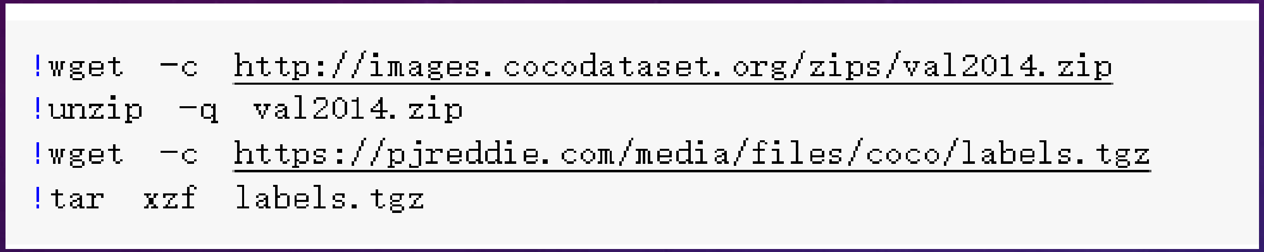
```
!wget  https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights
!wget  https://pjreddie.com/media/files/yolov3.weights
!wget  https://pjreddie.com/media/files/yolov3-tiny.weights
```

- Other weights are also available, please check out the link below.

https://github.com/AlexeyAB/darknet#pre-trained-models

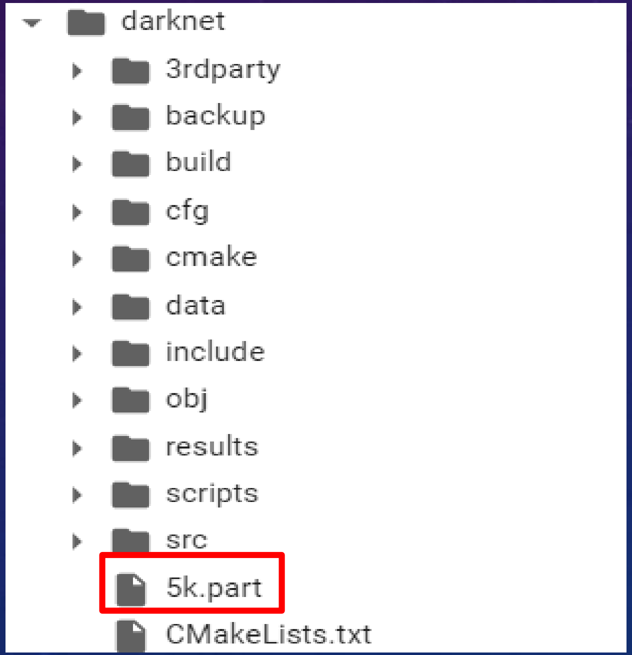
Example 6-2 YOLO – Dataset preparation

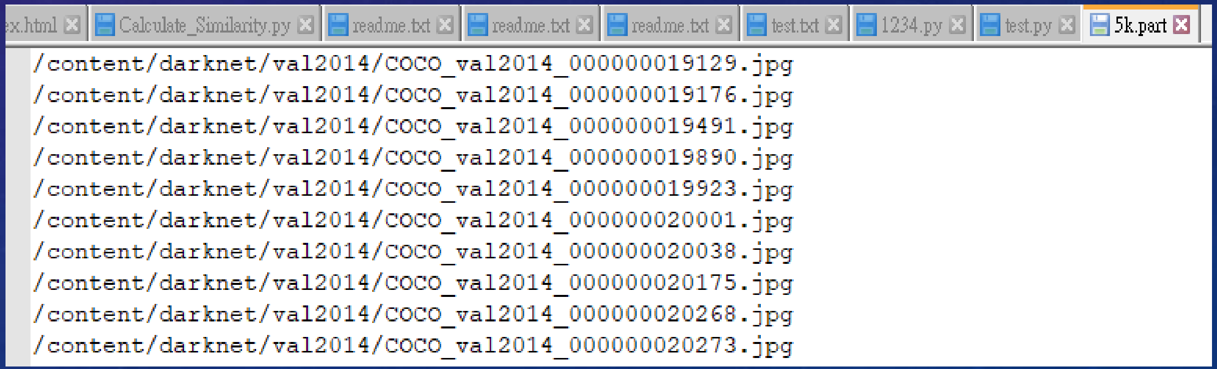
- Download the COCO val2014 dataset and labels.



```
!wget  -c  http://images.cocodataset.org/zips/val2014.zip
!unzip  -q  val2014.zip
!wget  -c  https://pjreddie.com/media/files/coco/labels.tgz
!tar  xzf  labels.tgz
```

- Upload 5k.part on darknet folder

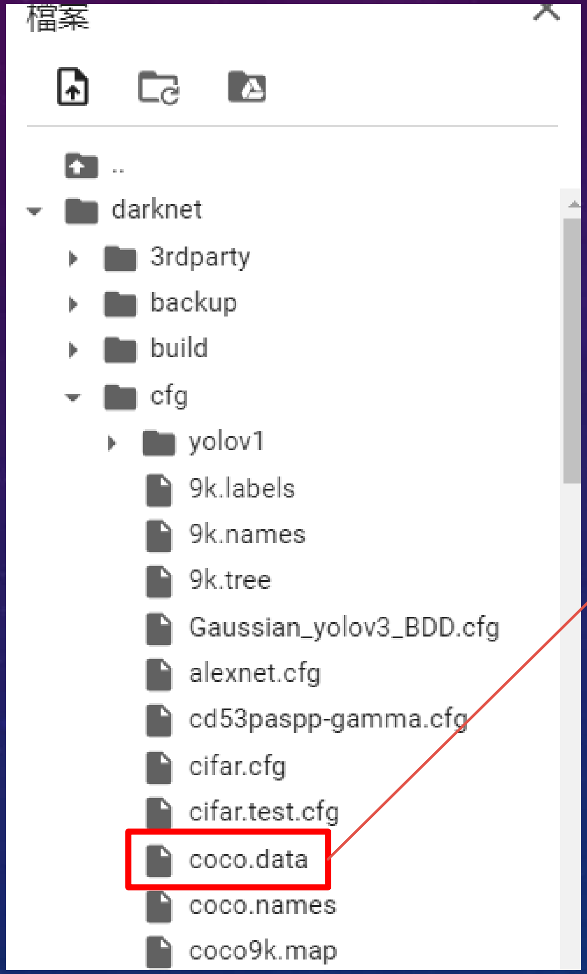




```
/content/darknet/val2014/COCO_val2014_000000019129.jpg
/content/darknet/val2014/COCO_val2014_000000019176.jpg
/content/darknet/val2014/COCO_val2014_000000019491.jpg
/content/darknet/val2014/COCO_val2014_000000019890.jpg
/content/darknet/val2014/COCO_val2014_000000019923.jpg
/content/darknet/val2014/COCO_val2014_000000020001.jpg
/content/darknet/val2014/COCO_val2014_000000020038.jpg
/content/darknet/val2014/COCO_val2014_000000020175.jpg
/content/darknet/val2014/COCO_val2014_000000020268.jpg
/content/darknet/val2014/COCO_val2014_000000020273.jpg
```

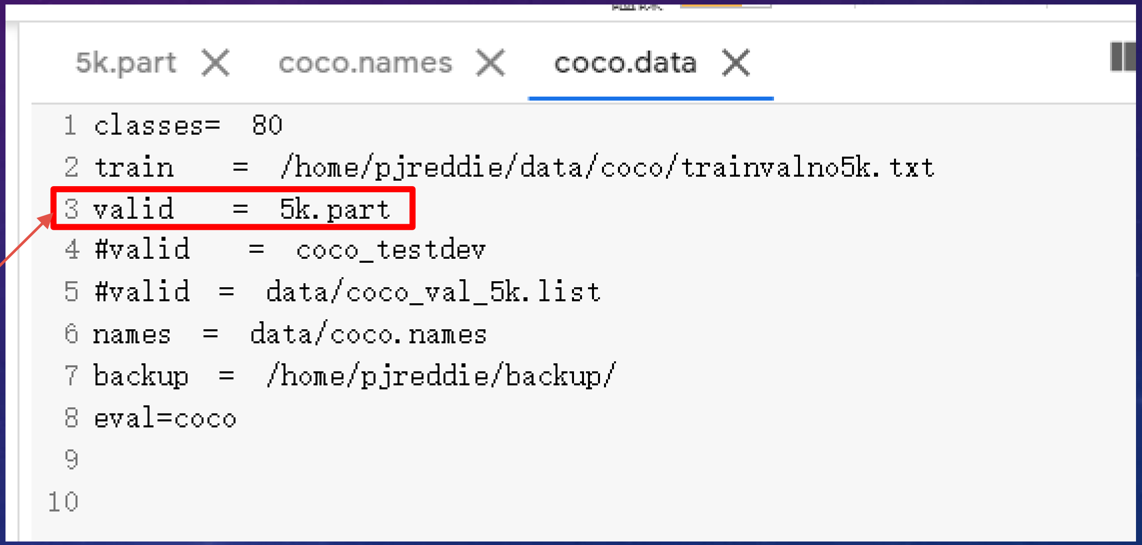
Example 6-2 YOLO – Dataset preparation

- Change the /content/darknet/cfg/coco.data information



檔案

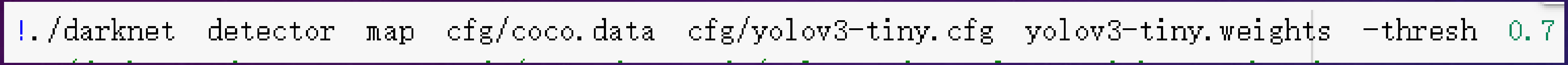
..
darknet
3rdparty
backup
build
cfg
yolov1
9k.labels
9k.names
9k.tree
Gaussian_yolov3_BDD.cfg
alexnet.cfg
cd53paspp-gamma.cfg
cifar.cfg
cifar.test.cfg
coco.data
coco.names
coco9k.map



```
classes= 80
train  =  /home/pjreddie/data/coco/trainvalno5k.txt
valid  =  5k.part
#valid   =  coco_testdev
#valid  =  data/coco_val_5k.list
names  =  data/coco.names
backup  =  /home/pjreddie/backup/
eval=coco


```

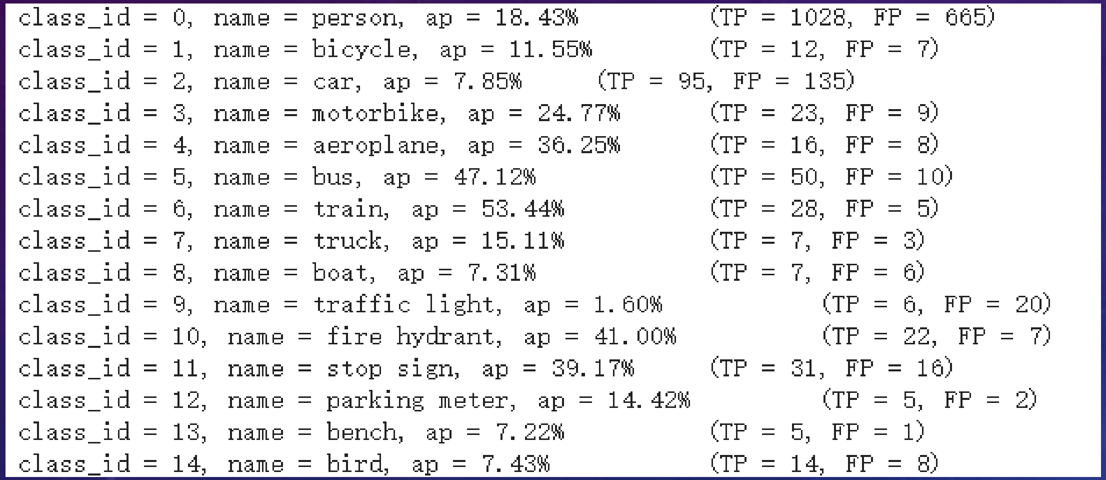
Example 6-2 YOLO – Evaluation

- Evaluate the results for YOLOV3-tiny pretrain models.



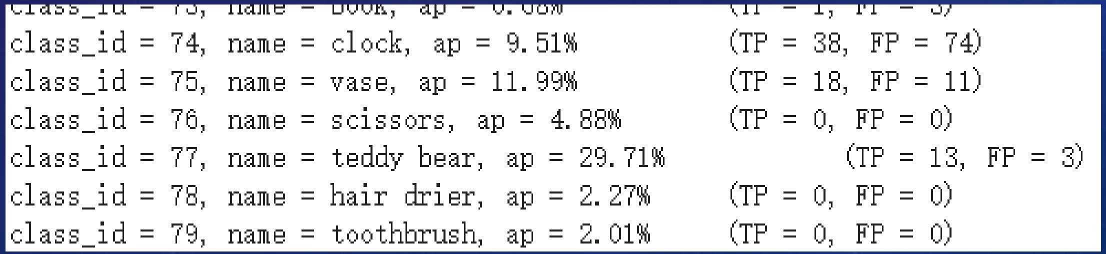
```
!./darknet  detector  map  cfg/coco.data  cfg/yolov3-tiny.cfg  yolov3-tiny.weights  -thresh  0.7
```

- For each class result



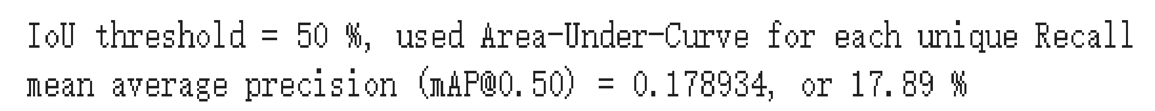
```
class_id = 0,  name = person,  ap = 18.43%          (TP = 1028,  FP = 665)
class_id = 1,  name = bicycle,  ap = 11.55%         (TP = 12,  FP = 7)
class_id = 2,  name = car,  ap = 7.85%      (TP = 95,  FP = 135)
class_id = 3,  name = motorbike,  ap = 24.77%       (TP = 23,  FP = 9)
class_id = 4,  name = aeroplane,  ap = 36.25%       (TP = 16,  FP = 8)
class_id = 5,  name = bus,  ap = 47.12%             (TP = 50,  FP = 10)
class_id = 6,  name = train,  ap = 53.44%           (TP = 28,  FP = 5)
class_id = 7,  name = truck,  ap = 15.11%           (TP = 7,  FP = 3)
class_id = 8,  name = boat,  ap = 7.31%             (TP = 7,  FP = 6)
class_id = 9,  name = traffic light,  ap = 1.60%            (TP = 6,  FP = 20)
class_id = 10,  name = fire hydrant,  ap = 41.00%           (TP = 22,  FP = 7)
class_id = 11,  name = stop sign,  ap = 39.17%      (TP = 31,  FP = 16)
class_id = 12,  name = parking meter,  ap = 14.42%          (TP = 5,  FP = 2)
class_id = 13,  name = bench,  ap = 7.22%           (TP = 5,  FP = 1)
class_id = 14,  name = bird,  ap = 7.43%            (TP = 14,  FP = 8)
```

⋮



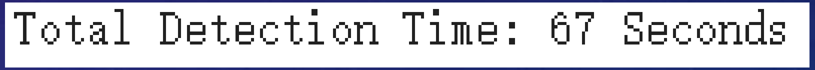
```
class_id = 73,  name = book,  ap = 0.68%            (TP = 1,  FP = 3)
class_id = 74,  name = clock,  ap = 9.51%           (TP = 38,  FP = 74)
class_id = 75,  name = vase,  ap = 11.99%           (TP = 18,  FP = 11)
class_id = 76,  name = scissors,  ap = 4.88%        (TP = 0,  FP = 0)
class_id = 77,  name = teddy bear,  ap = 29.71%             (TP = 13,  FP = 3)
class_id = 78,  name = hair drier,  ap = 2.27%      (TP = 0,  FP = 0)
class_id = 79,  name = toothbrush,  ap = 2.01%      (TP = 0,  FP = 0)
```

- mAP



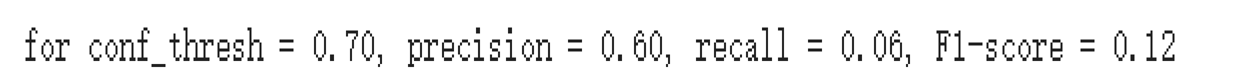
```
IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.178934, or 17.89 %
```

- Speed



```
Total Detection Time: 67 Seconds
```

- Precision and Recall



```
for conf_thresh = 0.70, precision = 0.60, recall = 0.06, F1-score = 0.12
```

Example 6-2 YOLO – PR-Curve

	Precision	Recall
Threshold 0.7	0.6	0.06
Threshold 0.5	0.51	0.11
Threshold 0.3	0.41	0.15
Threshold 0.1	0.23	0.24

Table 1

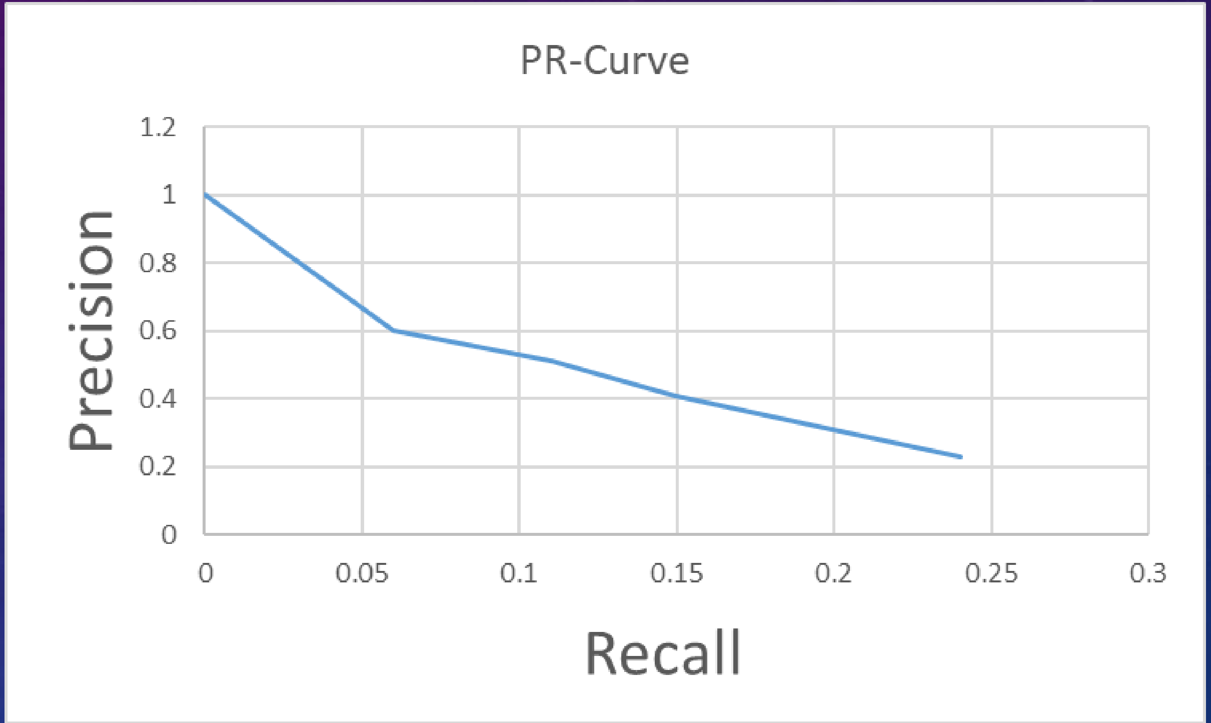


Figure 1

Exercise 6-2 – YOLO

- Please download “YOLO.ipynb” and open “YOLO.ipynb” by Colab.

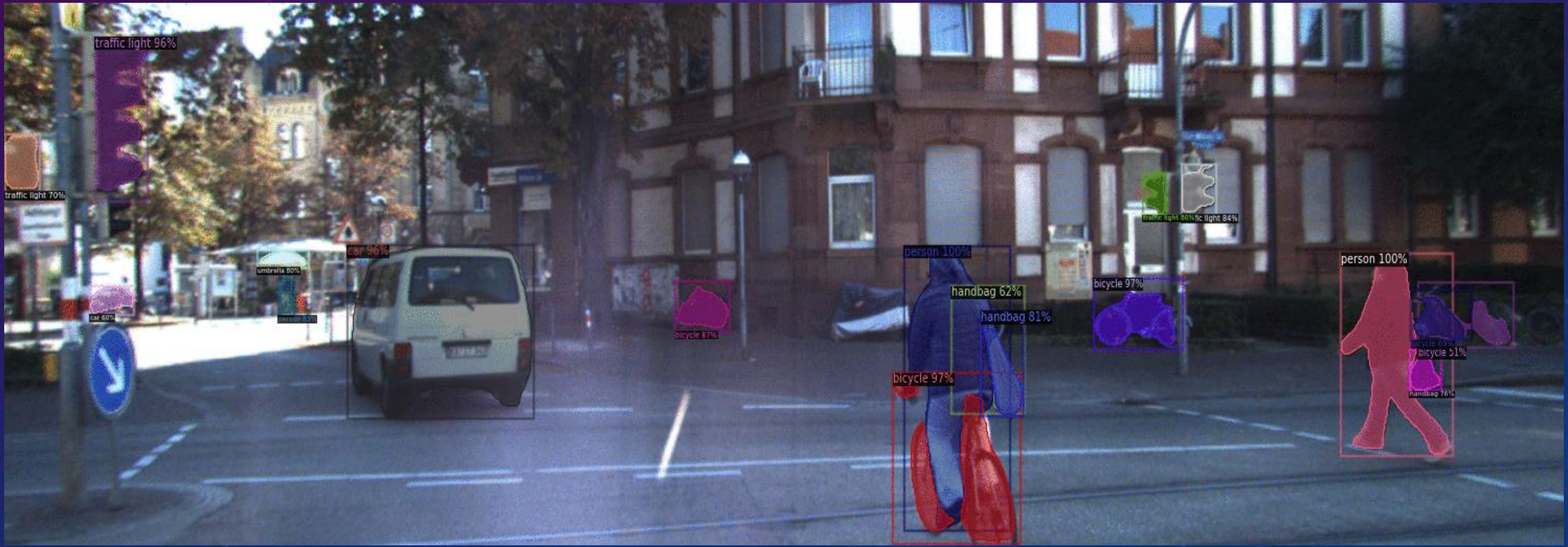
1. Choose YOLOv3-tiny, YOLOv3, YOLOv4 pretrain model to test on COCO_val2014 and compare their performance and speed on IOU [0.3].

2. Using the first question result draw the PR-Curve on IOU [0.1, 0.3, 0.5, 0.7].

Detectron2 for Video

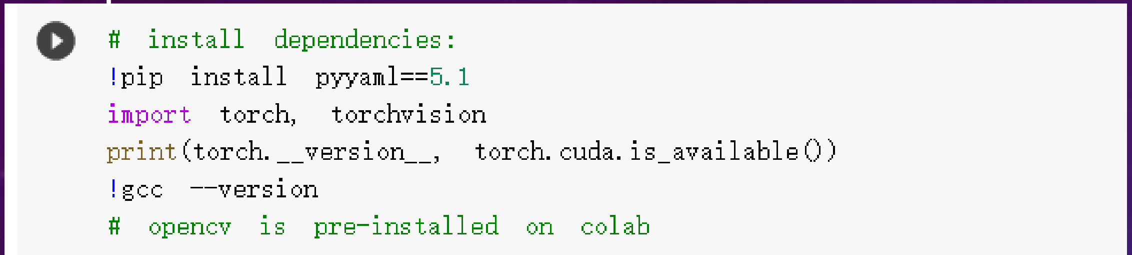
Example 6-3

- Use Facebook's Detectron2 object detection framework to detect the given video and build the masked image sequence.

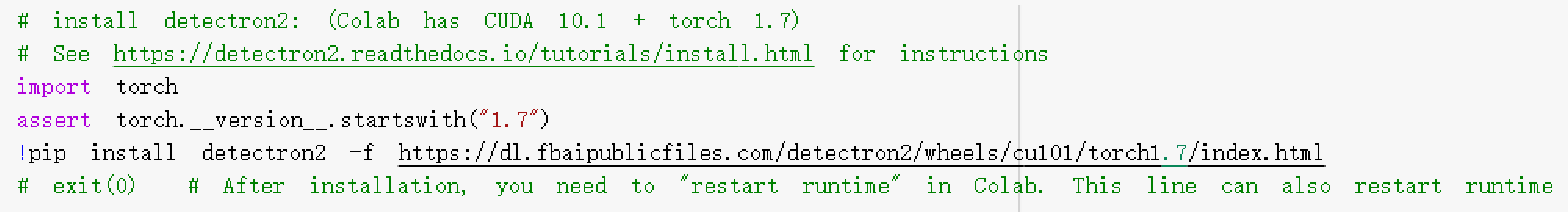


Example 6-3 Detectron2 evaluation

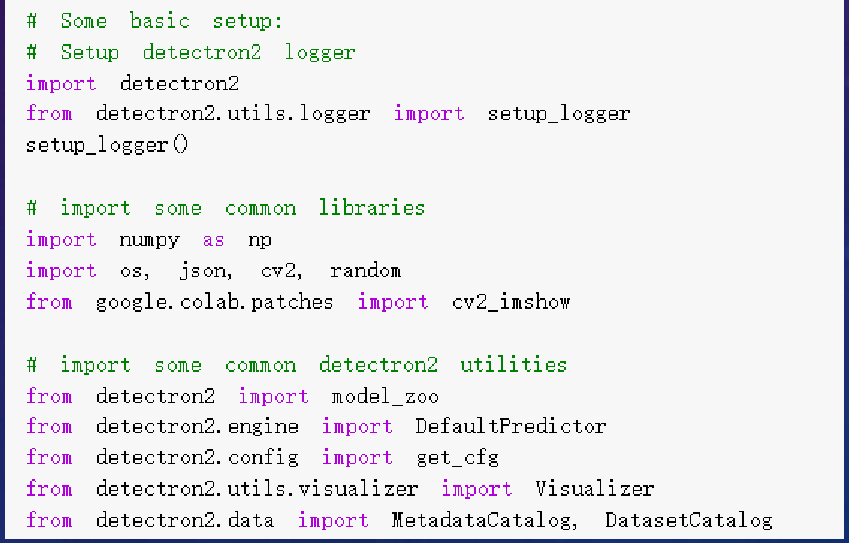
Setup the environment



```
# install dependencies:
!pip install pyyaml==5.1
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc --version
# opencv is pre-installed on colab
```



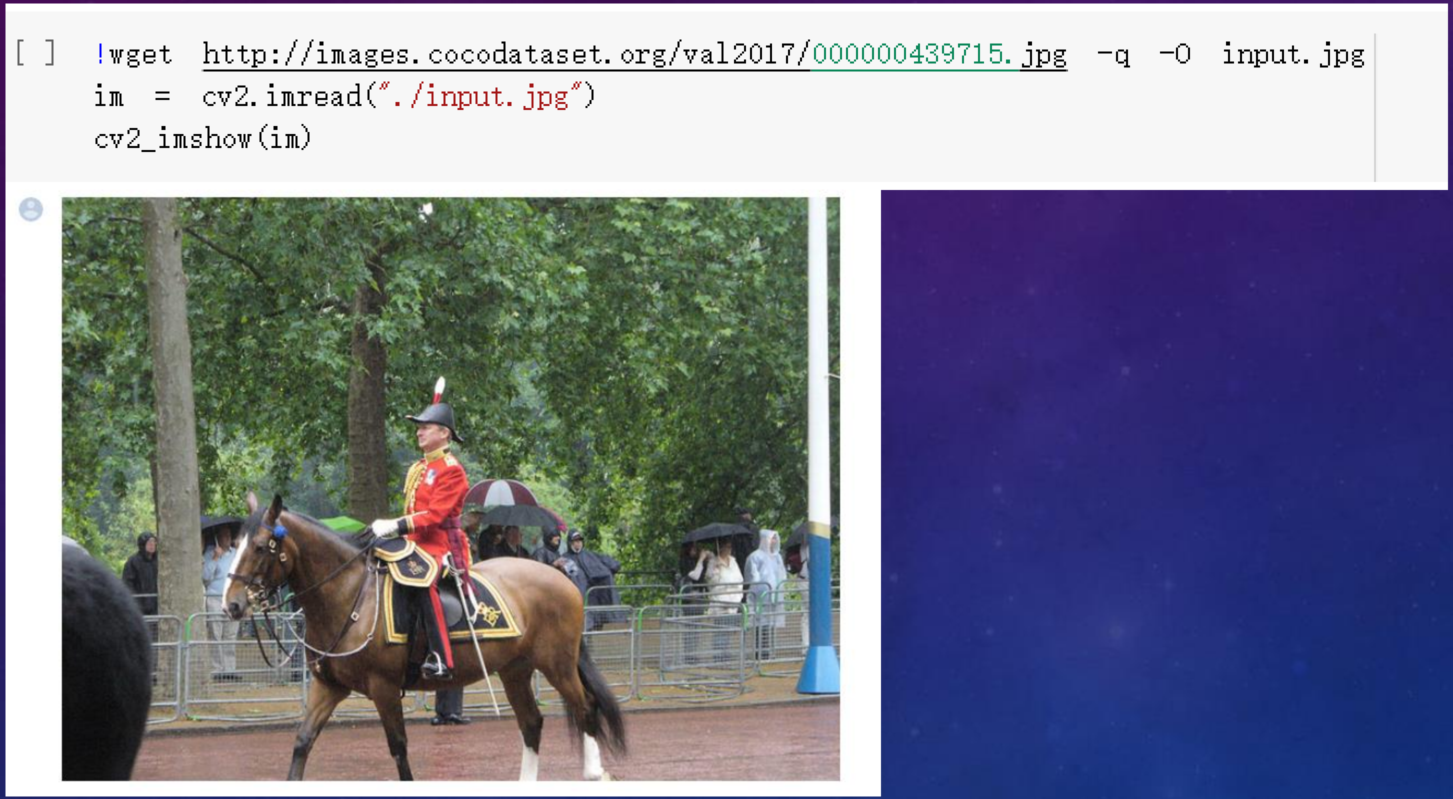
```
# install detectron2: (Colab has CUDA 10.1 + torch 1.7)
# See https://detectron2.readthedocs.io/tutorials/install.html for instructions
import torch
assert torch.__version__.startswith("1.7")
!pip install detectron2 -f https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.7/index.html
# exit(0)  # After installation, you need to "restart runtime" in Colab. This line can also restart runtime
```



```
# Some basic setup:
# Setup detectron2 logger
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()

# import some common libraries
import numpy as np
import os, json, cv2, random
from google.colab.patches import cv2_imshow

# import some common detectron2 utilities
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog, DatasetCatalog
```

Example 6-3 Detectron2 evaluation - Dataset preparation

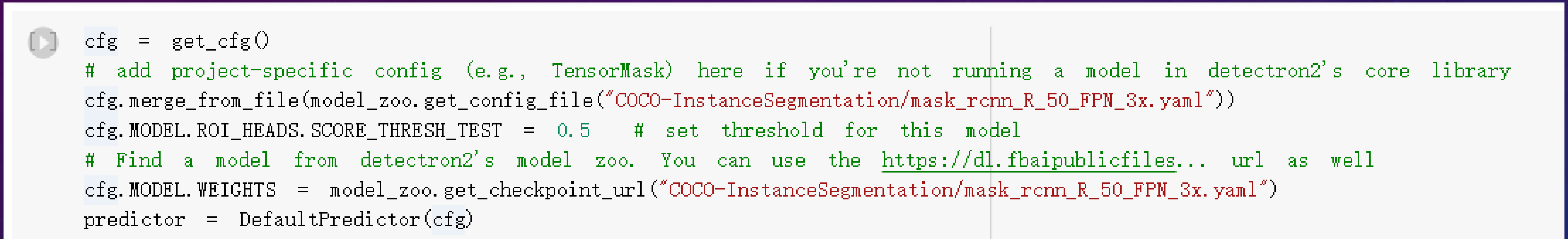
Show the sample image from COCO dataset.



```
!wget http://images.cocodataset.org/val2017/000000439715.jpg -q -O input.jpg
im = cv2.imread("./input.jpg")
cv2_imshow(im)
```

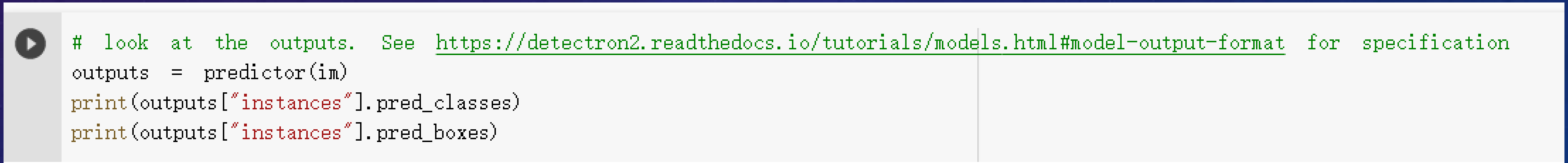
Example 6-3 Detectron2 evaluation - Dataset preparation

- Create the detectron2 config and the detectron2 “DefaultPredictor”.



```
cfg = get_cfg()
# add project-specific config (e.g., TensorMask) here if you're not running a model in detectron2's core library
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml"))
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5  # set threshold for this model
# Find a model from detectron2's model zoo. You can use the https://dl.fbaipublicfiles... url as well
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
predictor = DefaultPredictor(cfg)
```

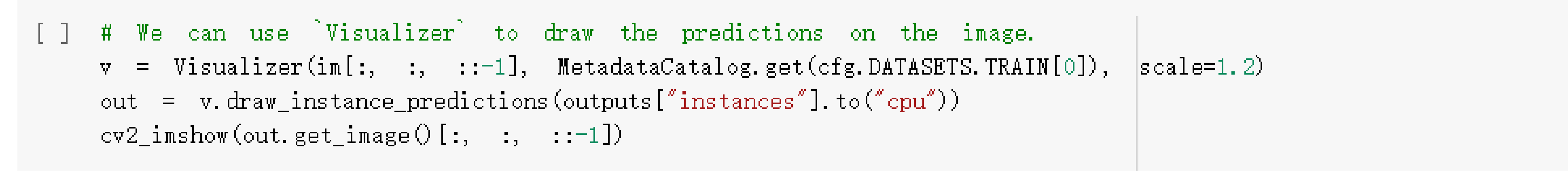
- Run the inference on the given sample image.



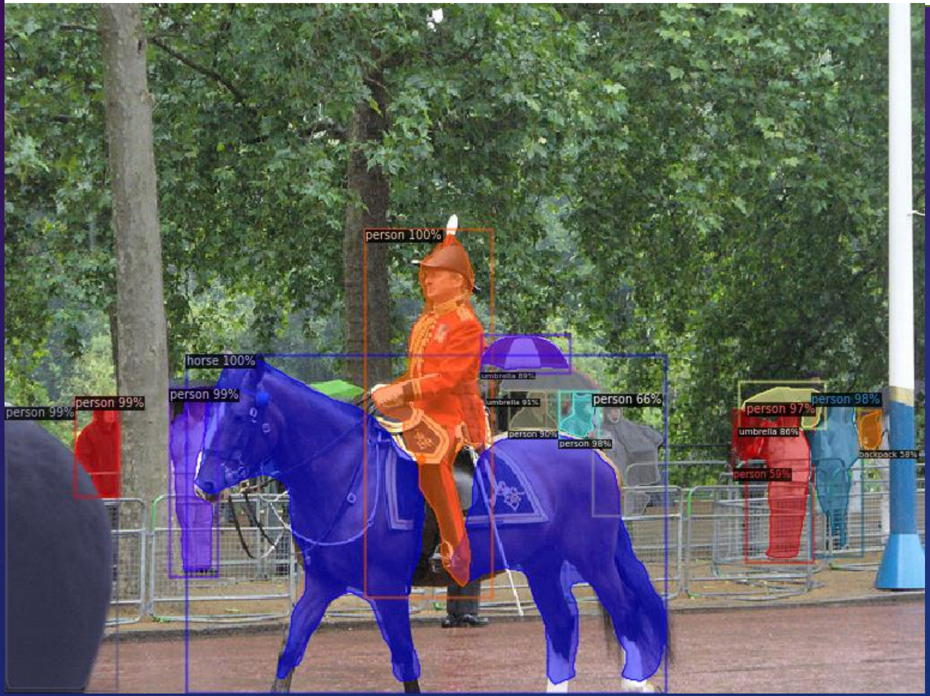
```
# look at the outputs. See https://detectron2.readthedocs.io/tutorials/models.html#model-output-format for specification
outputs = predictor(im)
print(outputs["instances"].pred_classes)
print(outputs["instances"].pred_boxes)
```

Example 6-3 Detectron2 evaluation - Dataset preparation

- Use “Visualizer” to draw the predictions on the image.

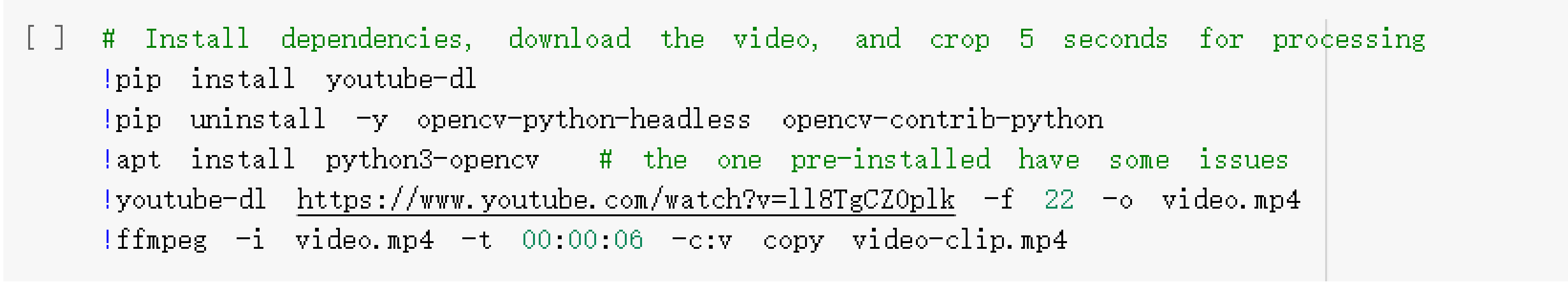


```
# We can use `Visualizer` to draw the predictions on the image.
v = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=1.2)
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
```



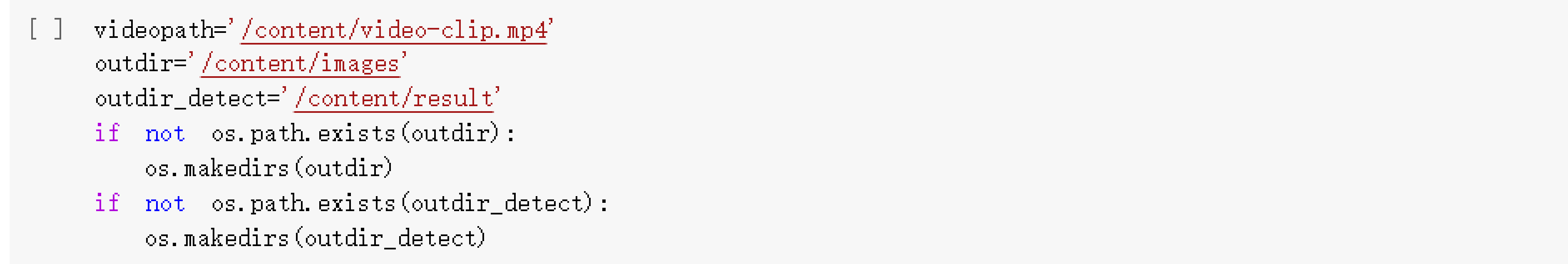
Example 6-3 Detectron2 evaluation - Dataset preparation

- Download the video and clip 5 seconds for inference.



```
# Install dependencies, download the video, and crop 5 seconds for processing
!pip install youtube-dl
!pip uninstall -y opencv-python-headless opencv-contrib-python
!apt install python3-opencv  # the one pre-installed have some issues
!youtube-dl https://www.youtube.com/watch?v=ll8TgCZ0plk -f 22 -o video.mp4
!ffmpeg -i video.mp4 -t 00:00:06 -c:v copy video-clip.mp4
```

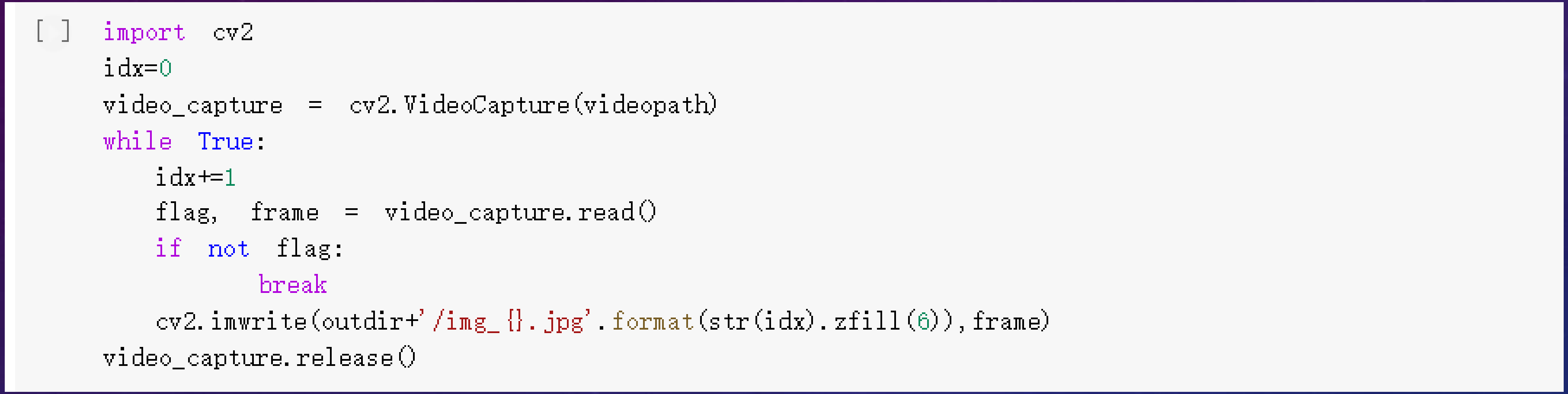
- Upload the clipped video and create the folder for saving the images.



```
videopath='/content/video-clip.mp4'
outdir='/content/images'
outdir_detect='/content/result'
if not os.path.exists(outdir):
    os.makedirs(outdir)
if not os.path.exists(outdir_detect):
    os.makedirs(outdir_detect)
```

Example 6-3 Detectron2 evaluation - Dataset preparation

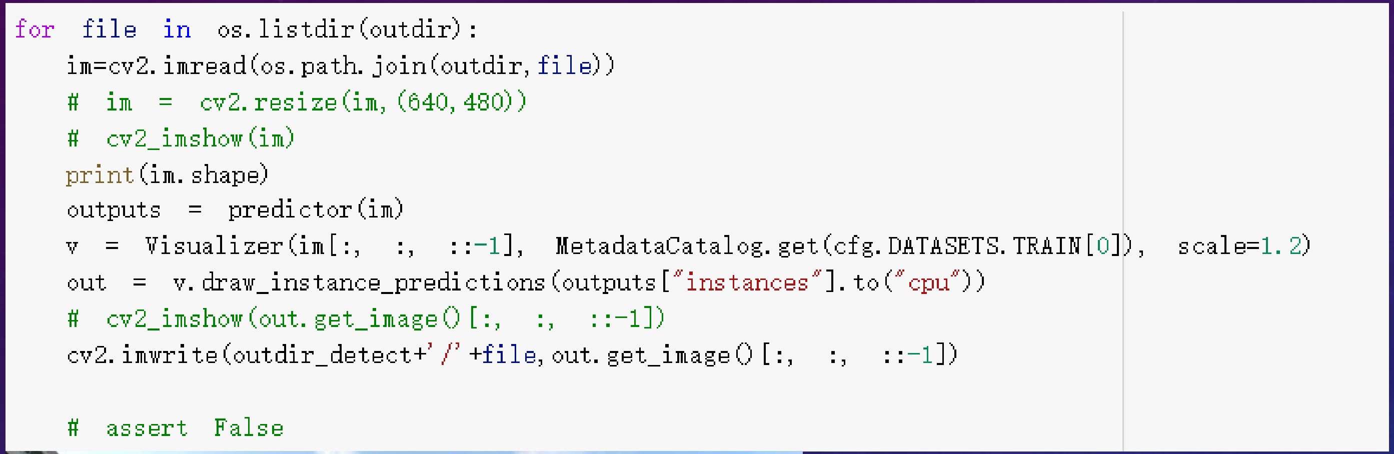
- Use OpenCV function to get the frames and save these images.



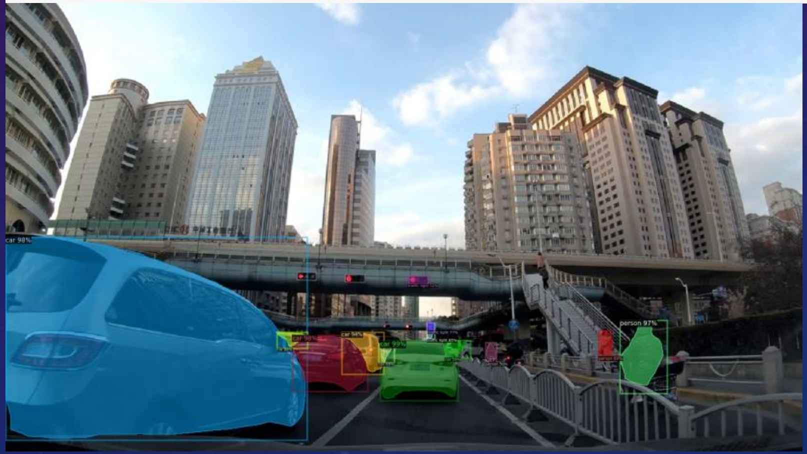
```
import cv2
idx=0
video_capture = cv2.VideoCapture(videopath)
while True:
    idx+=1
    flag, frame = video_capture.read()
    if not flag:
            break
    cv2.imwrite(outdir+’/img_{}.jpg’.format(str(idx).zfill(6)),frame)
video_capture.release()
```

Example 6-3 Detectron2 evaluation - Dataset preparation

- Use predictor of Detectron2 to generate the masked images



```
for file in os.listdir(outdir):
    im=cv2.imread(os.path.join(outdir,file))
    # im = cv2.resize(im,(640,480))
    # cv2_imshow(im)
    print(im.shape)
    outputs = predictor(im)
    v = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=1.2)
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    # cv2_imshow(out.get_image()[:, :, ::-1])
    cv2.imwrite(outdir_detect+'/'+file,out.get_image()[:, :, ::-1])

    # assert False
```



Exercise 6-3 – Detectron2

- Please download “Detectrom2.ipynb” and open “Detectrom2.ipynb” on the Colab.
    - Upload a short-time video to the Colab.
    - Use Facebook’s Detectron2 object detection framework to detect your video and make the masked images.
    - Generate the GIF file from the sequence of images.