Au 515 Drones et Asservissement Visuel
TP1 : Deep Learning and Object Detection

Jamy Chahal[1], Jonatan Alvarez[1], and A. Öztürk Suri[1]

***Index terms*— Index Terms—Vision, OpenCV, Object Detection, Deep Learning**

I. BACKGROUND

The current TP is to apply real-time object detection using deep learning and OpenCV to work with an image, video streams, and video files.

II. INSTRUCTIONS

The instructions for this TP session are enumerated. Please be careful about reading and following them. At the end of the TP:

1) Answer all the questions distributed along with the text.
2) Correct and make the code work.
3) Show your working code to the professor.
4) Submit all your answers as a pdf file (not codes) to the **Moodle TP3_submission** space.

III. OBJECT DETECTION FROM A VIDEO

To build a deep learning-based real-time object detector with OpenCV you will need to:

- access your webcam/video stream in an efficient manner
- apply object detection to each frame.

To see how this is done, open up a new file, name it as object_detection_from_video.py and insert *Code 1* into it as a first step.

Code 1

```
1 # import the necessary packages
2 import numpy as np
3 import argparse
4 import cv2
```

Then, you begin by importing packages on lines 2-4. You will need OpenCV. To get your system set up, simply install OpenCV using the relevant instructions for your system. Then, please add *Code 2* to your code as a second step and pay special attention to the parameters presented in it. **Question 1: Why do you need to construct the argument parser and what are the meaning of the following arguments?**

- prototxt
- weights
- thr

Code 2

```
5 # construct the argument parse
6 parser = argparse.ArgumentParser(
    description='Script to run MobileNet-SSD
    object detection network ')
7 parser.add_argument("--video", help="path to
    video file. If empty, camera's stream
    will be used")
8 parser.add_argument("--prototxt",
    default="MobileNetSSD_deploy.prototxt",
    help='Path to text network file: '
    'MobileNetSSD_deploy.prototxt for Caffe
    model or ')
9 parser.add_argument("--weights",
    default="MobileNetSSD_deploy.caffemodel",
    help='Path to weights: '
    'MobileNetSSD_deploy.caffemodel for
    Caffe model or ')
10 parser.add_argument("--thr", default=0.2,
    type=float, help="confidence threshold to
    filter out weak detections")
11 args = parser.parse_args()
```

As a third step, you will initialize class labels and corresponding random COLORS by adding *Code 3* into your code.

Code 3

```
12 # Labels of Network.
13 classNames = { 0: 'background',
    1: 'aeroplane', 2: 'bicycle', 3: 'bird',
    4: 'boat', 5: 'bottle', 6: 'bus', 7: 'car',
    8: 'cat', 9: 'chair', 10: 'cow',
    11: 'diningtable', 12: 'dog', 13: 'horse',
    14: 'motorbike', 15: 'person',
    16: 'pottedplant', 17: 'sheep', 18: 'sofa',
    19: 'train', 20: 'tvmonitor' }
14  COLORS = np.random.uniform(0, 255,
        size=(len(CLASSES), 3))
```

As a fourth step, you will add *Code 4* into your code. Then, **Question 2: please explain what happens in lines 15 and 16?**

Code 4

```
15    cap = cv2.VideoCapture('motor_bike.mp4')
16    net = cv2.dnn.readNetFromCaffe(args.prototxt,
      args.weights)
```

Now, let's loop over each frame by adding *Code 5* into your code. Before going further, **Question 3: what is blob?** In the code, first, you will read a frame (Line 19) from the

[1] LS2A IPSA Paris, Bis, 63 Boulevard de Brandebourg, 94200 Ivry-sur-Seine, France name.surname@ipsa.fr

stream, followed by resizing it (Line 21). **Question 4: what is happening in line 22?**

MobileNet requires fixed dimensions for input image(s) so you have to ensure that it is resized to 300x300 pixels. Then, you will set a scale factor to the image because the network has objects with different sizes. You will perform a mean subtraction (127.5, 127.5, 127.5) to normalize the input.

Code 5

```
17    while(cap.isOpened()):
18        # Capture frame-by-frame
19        ret, frame = cap.read()
20        # resize frame for prediction
21        frame_resized = cv2.resize(frame,
          (300,300))
22        blob = cv2.dnn.blobFromImage(
          frame_resized, 0.007843,
          (300, 300), (127.5, 127.5, 127.5), False)
23        #Set to network the input blob
24        net.setInput(blob)
25        #Prediction of network
26        detections = net.forward()
27        #Size of frame resize (300x300)
28        cols = frame_resized.shape[1]
29        rows = frame_resized.shape[0]
```

At this point, you have detected objects in the input frame. It is now time to look at confidence values and determine if you should draw a box + label surrounding the object. You will start by looping over detections by keeping in mind that multiple objects can be detected in a single image by using *Code 6*.

You will also apply a check to the confidence associated with each detection. If the confidence is high enough (i.e. above the threshold), then you will display the prediction in the terminal as well as draw the prediction on the image with text and a colored bounding box.

The remaining steps in the frame are to capture loop involve displaying the frame, to check for a quit key, and to update your frames per second counter by using *Code 7, 8, and 9*.

If you break out of the loop, you can press key 'q'. If you have made it this far, you're probably ready to give it a try with your webcam to see how it is done. Let's move on to the next section.

Code 6

```
30    for i in range(detections.shape[2]):
31        #Confidence of prediction
32        confidence = detections[0, 0, i, 2]
33        # Filter prediction
34        if confidence > args.thr:
35        # Class label
36        class_id = int(detections[0, 0, i, 1])
37        # Object location
38        xLeftBottom = int(detections[0, 0,
          i, 3] * cols)
39        yLeftBottom = int(detections[0, 0,
          i, 4] * rows)
40        xRightTop   = int(detections[0, 0,
          i, 5] * cols)
41        yRightTop   = int(detections[0, 0,
          i, 6] * rows)
42        # Factor for scale to original size of
          # frame
43        heightFactor = frame.shape[0]/300.0
44        widthFactor = frame.shape[1]/300.0
45        # Scale object detection to frame
46        xLeftBottom = int(widthFactor *
          xLeftBottom)
47        yLeftBottom = int(heightFactor *
          yLeftBottom)
48        xRightTop   = int(widthFactor *
          xRightTop)
49        yRightTop   = int(heightFactor *
          yRightTop)
50        # Draw location of object
51        cv2.rectangle(frame, (xLeftBottom,
          yLeftBottom), (xRightTop, yRightTop),
          (0, 255, 0))
```

Code 7

```
52    # Draw label and confidence of prediction
53    # in frame resized
54    if class_id in classNames:
55        label = classNames[class_id] + ": " +
          str(confidence)
56        labelSize, baseLine = cv2.getTextSize(
          label, cv2.FONT_HERSHEY_SIMPLEX, 0.5, 1)
57        yLeftBottom = max(yLeftBottom,
          labelSize[1])
58        cv2.rectangle(frame, (xLeftBottom,
          yLeftBottom - labelSize[1]),
          (xLeftBottom + labelSize[0], yLeftBottom
          + baseLine), (255, 255, 255), cv2.FILLED)
59        cv2.putText(frame, label, (xLeftBottom,
          yLeftBottom), cv2.FONT_HERSHEY_SIMPLEX,
          0.5, (0, 0, 0))
```

Code 8

```
60  cv2.namedWindow("frame", cv2.WINDOW_NORMAL)
61  if ret == True:
62      # Display the resulting frame
63      cv2.imshow('Frame',frame)
64      # Press Q on keyboard to  exit
65      if cv2.waitKey(25) & 0xFF == ord('q'):
66          break
67  # Break the loop
68  else:
69      break
```

Code 9

```
70  # When everything is done, release the video
    # capture object
71  cap.release()
72  # Closes all the frames
73  cv2.destroyAllWindows()
```

IV. REAL-TIME DEEP LEARNING OBJECT DETECTION

To see the real-time deep-learning-based object detector inaction, open up a new file, name it as object_detection_from_webcam.py and insert *Replace Code 4* into it with all your remaining code.

Provided that OpenCV can access your webcam you should see the output video frame with any detected objects.

To see how this is done, open up a new file, name it as object_detection_from_video.py and insert *Code 1* into it as a first step.

Replace Code 4

```
    # Open video file or capture device.
    if args.video:
        cap = cv2.VideoCapture(args.video)
    else:
        cap = cv2.VideoCapture(0)
```

Question 5: what is your proposed solution for detecting objects from an image? You can use provided image in the download file to show your results.

V. REFERENCES

For more references and information about the content of the TP, the reader is encouraged to read the slides and information provided by the professor as well as the documentation, manuals, and tutorials available at different websites of Python, OpenCV, and Deep learning applications.