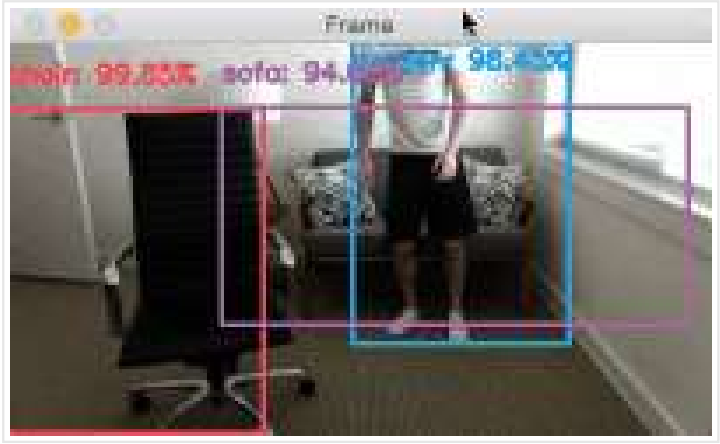
pyimagesearch
be awesome at building image search engines

Real-time object detection with deep learning and OpenCV

by **Adrian Rosebrock** on September 18, 2017 in **Deep Learning**, **OpenCV 3**, **Tutorials**

113



Today's blog post was inspired by PyImageSearch reader, Emmanuel. Emmanuel emailed me after last week's tutorial on object detection with deep learning + OpenCV and asked:

> *"Hi Adrian,*
>
> *I really enjoyed last week's blog post on object detection with deep learning and OpenCV, thanks for putting it together and for making deep learning with OpenCV so accessible.*
>
> *I want to apply the same technique to real-time video.*
>
> *What is the best way to do this?*
>
> *How can I achieve the most efficiency?*
>
> *If you could do a tutorial on real-time object detection with deep learning and OpenCV I would really appreciate it."*

Free 21-day crash course on computer vision & image search engines

Great question, thanks for asking Emmanuel.

Luckily, extending our previous tutorial on object detection with deep learning and OpenCV to ***real-time***

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Today's blog post is broken into two parts.

In the first part we'll learn how to extend last week's tutorial to apply real-time object detection using deep learning and OpenCV to work with video streams and video files. This will be accomplished using the highly efficient `VideoStream` class discussed in this tutorial.

From there, we'll apply our deep learning + object detection code to actual video streams and measure the FPS processing rate.

Object detection in video with deep learning and OpenCV

To build our deep learning-based real-time object detector with OpenCV we'll need to (1) access our webcam/video stream in an efficient manner and (2) apply object detection to each frame.

To see how this is done, open up a new file, name it `real_time_object_detection.py` and insert the following code:

Real-time object detection with deep learning and OpenCV — Python

```
# import the necessary packages
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2
```

We begin by importing packages on **Lines 2-8.** For this tutorial, you will need imutils and OpenCV 3.3.

To get your system set up, simply install OpenCV using the relevant instructions for your system (while ensuring you're following any Python virtualenv commands).

Free 21-day crash course on computer vision & image search engines

newer) to run the code in this tutorial.

Next, we'll parse our command line arguments:

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

- `--prototxt` : The path to the Caffe prototxt file.
- `--model` : The path to the pre-trained model.
- `--confidence` : The minimum probability threshold to filter weak detections. The default is 20%.

We then initialize a class list and a color set:

Real-time object detection with deep learning and OpenCV — Python

```
20 # initialize the list of class labels MobileNet SSD was trained to
21 # detect, then generate a set of bounding box colors for each class
22 CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
23     "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
24     "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
25     "sofa", "train", "tvmonitor"]
26 COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
```

On **Lines 22-26** we initialize `CLASS` labels and corresponding random `COLORS`. For more information on these classes (and how the network was trained), please refer to last week's blog post.

Now, let's load our model and set up our video stream:

Real-time object detection with deep learning and OpenCV — Python

```
28 # load our serialized model from disk
29 print("[INFO] loading model...")
30 net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
31
32 # initialize the video stream, allow the cammera sensor to warmup,
33 # and initialize the FPS counter
34 print("[INFO] starting video stream...")
35 vs = VideoStream(src=0).start()
36 time.sleep(2.0)
37 fps = FPS().start()
```

We load our serialized model, providing the references to our prototxt and model files on **Line 30** — notice how easy this is in OpenCV 3.3.

Next let's initialize our video stream (this can be from a video file or a camera). First we start the

Free 21-day crash course on computer vision & image search engines

frames per second counter (**Line 37**). The `VideoStream` and `FPS` classes are part of my `imutils` package.

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!


```
52      # predictions
53      net.setInput(blob)
54      detections = net.forward()
```

First, we read a `frame` (**Line 43**) from the stream, followed by resizing it (**Line 44**).

Since we will need the width and height later, we grab these now on **Line 47.** This is followed by converting the `frame` to a `blob` with the `dnn` module (**Lines 48 and 49**).

Now for the heavy lifting: we set the `blob` as the input to our neural network (**Line 53**) and feed the input through the `net` (**Line 54**) which gives us our `detections` .

At this point, we have detected objects in the input frame. It is now time to look at confidence values and determine if we should draw a box + label surrounding the object– you'll recognize this code block from last week:

Real-time object detection with deep learning and OpenCV — Python

```
55      # loop over the detections
56      for i in np.arange(0, detections.shape[2]):
57          # extract the confidence (i.e., probability) associated with
58          # the prediction
59          confidence = detections[0, 0, i, 2]
60
61          # filter out weak detections by ensuring the `confidence` is
62          # greater than the minimum confidence
63          if confidence > args["confidence"]:
64              # extract the index of the class label from the
65              # `detections`, then compute the (x, y)-coordinates of
66              # the bounding box for the object
67              idx = int(detections[0, 0, i, 1])
68              box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
69              (startX, startY, endX, endY) = box.astype("int")
70
71              # draw the prediction on the frame
72              label = "{}: {:.2f}%".format(CLASSES[idx],
73                  confidence * 100)
74              cv2.rectangle(frame, (startX, startY), (endX, endY),
```

Free 21-day crash course on computer vision & image search engines

```
77              cv2.putText(frame, label, (startX, y),
78                  cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)
```

We start by looping over our `detections`, keeping in mind that multiple objects can be detected in a

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

We build a text `label` containing the `CLASS` name and the `confidence` (**Lines 73 and 74**).

Let's also draw a colored rectangle around the object using our class color and previously extracted *(x, y)*-coordinates (**Lines 75 and 76**).

In general, we want the label to be displayed above the rectangle, but if there isn't room, we'll display it just below the top of the rectangle (**Line 77**).

Finally, we overlay the colored text onto the `frame` using the *y*-value that we just calculated (**Lines 78 and 79**).

The remaining steps in the frame capture loop involve (1) displaying the frame, (2) checking for a quit key, and (3) updating our frames per second counter:

Real-time object detection with deep learning and OpenCV — Python

```
80      # show the output frame
81      cv2.imshow("Frame", frame)
82      key = cv2.waitKey(1) & 0xFF
83
84      # if the `q` key was pressed, break from the loop
85      if key == ord("q"):
86          break
87
88      # update the FPS counter
89      fps.update()
```

The above code block is pretty self-explanatory — first we display the frame (**Line 82**). Then we capture a key press (**Line 83**) while checking if the 'q' key (for "quit") is pressed, at which point we break out of the frame capture loop (**Lines 86 and 87**).

Finally we update our fps counter (**Line 90**).

If we break out of the loop ('q' key press or end of the video stream), we have some housekeeping to

Free 21-day crash course on computer vision & image search engines

```
Real-time object detection with deep learning and OpenCV                Python
91 # stop the timer and display FPS information
92 fps.stop()
```

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Real-time deep learning object detection results

To see our real-time deep-learning based object detector in action, make sure you use the ***"Downloads"*** section of this guide to download the example code + pre-trained Convolutional Neural Network.

From there, open up a terminal and execute the following command:

```
Real-time object detection with deep learning and OpenCV                Shell
1 $ python real_time_object_detection.py \
2     --prototxt MobileNetSSD_deploy.prototxt.txt \
3     --model MobileNetSSD_deploy.caffemodel
4 [INFO] loading model...
5 [INFO] starting video stream...
6 [INFO] elapsed time: 55.07
7 [INFO] approx. FPS: 6.54
```

Provided that OpenCV can access your webcam you should see the output video frame with any detected objects. I have included sample results of applying deep learning object detection to an example video below:

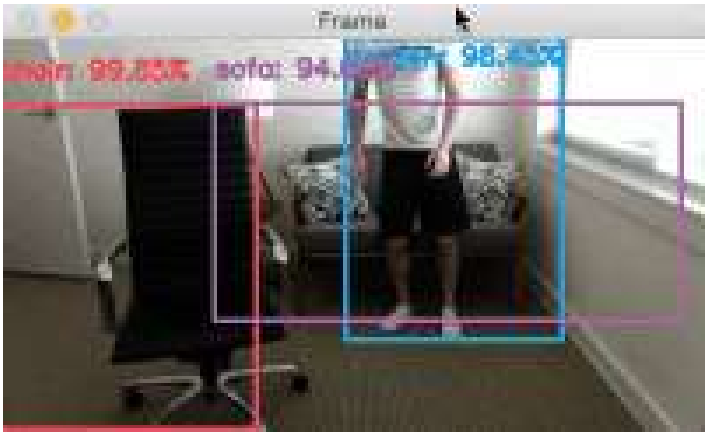


Figure 1: A short clip of real-time object detection with deep

Free 21-day crash course on computer vision & image search engines

Notice how our deep learning object detector can detect not only myself (a *person*), but also the *sofa* I am sitting on and the *chair* next to me — ***all in real-time!***

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Summary

In today's blog post we learned how to perform real-time object detection using deep learning + OpenCV + video streams.

We accomplished this by combing two separate tutorials:

1. Object detection with deep learning and OpenCV
2. Efficient, threaded video streams with OpenCV

The end result is a deep learning-based object detector that can process approximately 6-8 FPS (depending on the speed of your system, of course).

Further speed improvements can be obtained by:

1. Applying skip frames.
2. Swapping different variations of MobileNet (that are faster, but less accurate).
3. Potentially using the quantized variation of SqueezeNet (I haven't tested this, but imagine it would be faster due to smaller network footprint).

In future blog posts we'll be discussing deep learning object detection methods in more detail.

In the meantime, be sure to take a look at my book, *Deep Learning for Computer Vision with Python*, where I'll be reviewing object detection frameworks such as Faster R-CNNs and Single Shot Detectors!

If you're interested in studying deep learning for computer vision and image classification tasks, you just

Free 21-day crash course on computer vision & image search engines

Downloads:

Free 21-day crash course on computer vision & image search engines

Interested in computer vision and image search engines, but don't know where to start? Let me help. I've created a free, 21-day crash course that is hand-tailored to give you the best possible introduction to computer vision. Sound good? Enter your email below to start your journey to becoming a computer vision master.

Email Address

LET'S DO IT!

Resource Guide (it's totally free).

Enter your email address below to get my **free 11-page Image Search Engine Resource Guide PDF**. Uncover **exclusive techniques** that I don't publish on this blog and start building image search engines of your own!

Your email address

DOWNLOAD THE GUIDE!

cnn, **coco**, **convolutional neural network**, **deep learning**, **machine learning**, **opencv 3**, **pascal voc**, **real-time**, **video**, **video stream**

‹ Object detection with deep learning and OpenCV

Pre-configured Amazon AWS deep learning AMI with Python ›

92 Responses to *Real-time object detection with deep learning and OpenCV*

Daniel Funseth September 18, 2017 at 10:55 am #

REPLY

wow! this is really impressive, will it be able to run on a RPI 3?

Free 21-day crash course on computer vision & image search engines