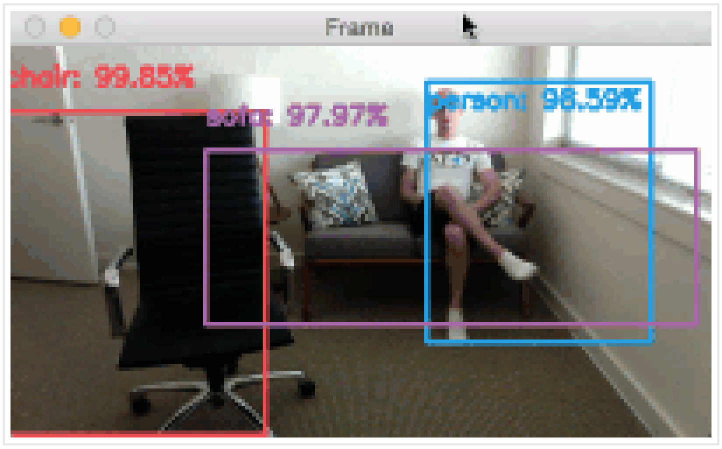
Real-time object detection with deep learning and OpenCV

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



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."*

Great question, thanks for asking Emmanuel.

Luckily, extending our previous tutorial on object detection with deep learning and OpenCV to ***real-time video streams*** is fairly straightforward — we simply need to combine some efficient, boilerplate code for real-time video access and then add in our object detection.

By the end of this tutorial you'll be able to apply deep learning-based object detection to real-time video streams using OpenCV and Python — to learn how, *just keep reading*.

Looking for the source code to this post?
Jump right to the downloads section.

Real-time object detection with deep learning and OpenCV

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 learnir                Python
1 # import the necessary packages
2 from imutils.video import VideoStream
3 from imutils.video import FPS
4 import numpy as np
5 import argparse
6 import imutils
7 import time
8 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).

Note: *Make sure to download and install opencv and and opencv-contribreleases for OpenCV 3.3. This will ensure that the deep neural network (`dnn`) module is installed. You **must** have OpenCV 3.3 (or newer) to run the code in this tutorial.*

Next, we'll parse our command line arguments:

```
Real-time object detection with deep learnir                Python
10 # construct the argument parse and parse the arguments
11 ap = argparse.ArgumentParser()
12 ap.add_argument("-p", "--prototxt", required=True,
13     help="path to Caffe 'deploy' prototxt file")
14 ap.add_argument("-m", "--model", required=True,
15     help="path to Caffe pre-trained model")
16 ap.add_argument("-c", "--confidence", type=float, default=0.2,
17     help="minimum probability to filter weak detections")
18 args = vars(ap.parse_args())
```

Compared to last week, we don't need the image argument since we're working with streams and videos — other than that the following arguments remain the same:

- `--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 learnir Python

```
20 # initialize the list of class labels MobileNet SSD was trained
21 # detect, then generate a set of bounding box colors for each cl
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 learnir 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 warmu
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 `VideoStream` (**Line 35**), then we wait for the camera to warm up (**Line 36**), and finally we start the frames per second counter (**Line 37**). The `VideoStream` and `FPS` classes are part of my `imutils` package.

Now, let's loop over each and every frame (for speed purposes, you could skip frames):

Real-time object detection with deep learnir Python

```
39 # loop over the frames from the video stream
40 while True:
41     # grab the frame from the threaded video stream and resize i
42     # to have a maximum width of 400 pixels
```

```
43      frame = vs.read()
44      frame = imutils.resize(frame, width=400)
45
46      # grab the frame dimensions and convert it to a blob
47      (h, w) = frame.shape[:2]
48      blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
49          0.007843, (300, 300), 127.5)
50
51      # pass the blob through the network and obtain the detection
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 learni                  Python
55      # loop over the detections
56      for i in np.arange(0, detections.shape[2]):
57          # extract the confidence (i.e., probability) associated
58          # the prediction
59          confidence = detections[0, 0, i, 2]
60
61          # filter out weak detections by ensuring the `confidence
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
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,
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),
75                  COLORS[idx], 2)
76              y = startY - 15 if startY - 15 > 15 else startY + 15
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 single image. We also apply a check to the confidence (i.e., probability) associated with each detection. If the confidence is high enough (i.e. above the threshold), then we'll display the prediction in the terminal as well as draw the prediction on the image with text and a colored bounding box. Let's break it down line-by-line:

Looping through our `detections` , first we extract the `confidence` value (**Line 60**).

If the `confidence` is above our minimum threshold (**Line 64**), we extract the class label index (**Line 68**) and compute the bounding box coordinates around the detected object (**Line 69**).

Then, we extract the *(x, y)*-coordinates of the box (**Line 70**) which we will will use shortly for drawing a rectangle and displaying text.

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 learni Python

```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 take care of:

Real-time object detection with deep learni Python

```python
91 # stop the timer and display FPS information
92 fps.stop()
93 print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
94 print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
95
96 # do a bit of cleanup
97 cv2.destroyAllWindows()
98 vs.stop()
```

When we've exited the loop, we stop the `fps` counter (**Line 93**) and print information about the frames per second to our terminal (**Lines 94 and 95**).

We close the open window (**Line 98**) followed by stopping the video stream (**Line 99**).

If you've made it this far, you're probably ready to give it a try with your webcam — to see how it's done, let's move on to the next section.

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                    Shell
$ python real_time_object_detection.py \
    --prototxt MobileNetSSD_deploy.prototxt.txt \
    --model MobileNetSSD_deploy.caffemodel
[INFO] loading model...
[INFO] starting video stream...
[INFO] elapsed time: 55.07
[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:

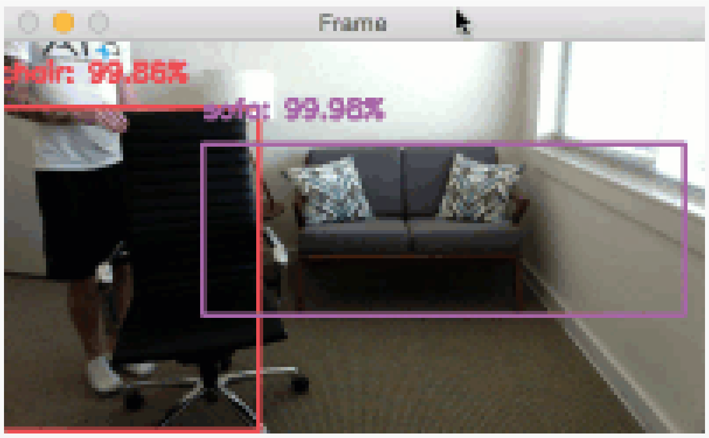


Figure 1: A short clip of real-time object detection with deep learning and OpenCV + Python.

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!***

The full video can be found below:

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 can't beat this book — click here to learn more.

Downloads:

DOWNLOAD THE GUIDE!

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

146 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?

Adrian Rosebrock September 18, 2017 at 1:56 pm # REPLY

Please see my reply to "Flávio".

Flávio Rodrigues September 18, 2017 at 11:01 am # REPLY

Hi, Adrian. Thanks a lot for another great tutorial. Have you got a real-time example working on a Pi 3? (Maybe skipping frames?) I'm just using a Pi (for OpenCV and DL) and I'd like to know to what extent is it'd be usable. What about the frame rate? Nice work as always. Cheers.

Adrian Rosebrock September 18, 2017 at 1:56 pm # REPLY

I haven't applied this method to the Raspberry Pi (yet) but I will very soon. It will be covered in a future blog post. I'll be sharing any optimizations I've made.

Nicolas September 18, 2017 at 11:01 am # REPLY

Hi Adrian.

Excelent, you are a great developer! But, I want to know how develop a face-tracking with opencv and python in the Backend, but capturing video en canvas with HTML5 real-time and after draw and object depending of the Backend´s Response, for example, a Moustache. Too, this tracking has support with head movement and Moustache adapts.

Thanks.

tommy September 18, 2017 at 11:22 am # REPLY

hi what's the best FPS on say a typical Macbook assuming u used threading and other optimisations?

tommy September 18, 2017 at 11:27 am # REPLY

in particular, does it mean if i used N threads to process the frames, the FPS will be N times better?

does the dnn module use any threading / multi-core underneath the hood?

Adrian Rosebrock September 18, 2017 at 1:54 pm # REPLY

No, the threading here only applies to polling of frames from the video sensor. The threading helps with I/O latency on the video itself. If you're looking to speedup the actual detection you should push the object detection to the GPU (which I think can only be done with the C++ bindings of OpenCV).

Adrian Rosebrock September 18, 2017 at 1:55 pm # REPLY

I used my MacBook Pro to collect the results to this blog post — approximately 5-7 FPS depending on your system specs.

vinu September 18, 2017 at 1:00 pm # REPLY

Thanks
its so much help

and i needs to detect only helmet in realtime

Ashwin Venkat September 18, 2017 at 9:28 pm # REPLY

Hi interesting thought, did it work

Eng.AAA September 18, 2017 at 1:31 pm # REPLY

Thanks for awesome Tutorials .
I have question about: can I track the location of the chair in video, I mean if the chair moving can I track its location.
Thanks

Adrian Rosebrock September 18, 2017 at 1:52 pm # REPLY

I would suggest using object detection (such as the method used in this post) to detect the chair/object. Then once you have the object detected pass the location into an object tracking algorithm, such as correlation tracking.

Eng.AAA September 18, 2017 at 5:59 pm # REPLY

I hope it will cover with an example in new Deep Learning Book

Thanks

Sydney September 18, 2017 at 1:36 pm # REPLY

Thanks for the tutorial man. The method is quite effective, running better on a CPU. I am still trying to figure out how i can use a video as my source instead of the webcam.

Adrian Rosebrock September 18, 2017 at 1:52 pm # REPLY

Thanks Sydney, I'm glad it worked for you 🙂

As far as working with video files, take a look at this post.

Ashraf September 18, 2017 at 2:07 pm # REPLY

great article! continue the great way!

Adrian Rosebrock September 18, 2017 at 2:21 pm # REPLY

Thanks Ashraf 🙂

Walid Ahmed September 18, 2017 at 2:18 pm # REPLY

Thanks. I waited for 18 Sep to read this blog!

Just one questions , isnt 0.2 so low as confidence?
would not this result in low precision?

Adrian Rosebrock September 18, 2017 at 2:21 pm # REPLY

With object detection we typically have lower probabilities in the localization. You can tune the threshold probability to your liking.

Jacques September 18, 2017 at 2:18 pm # REPLY

Hey Mate,

Many thanks for the great example code – just what I needed :)..

How would this perform on a Pi 3? I intend testing it asap, but I would guess the classification function would be really slow (I was getting up to 4 seconds on your previous tutorial using cv DNN)? Any views on how to compensate for the slower processor?

Do you believe that this code would be too much for the Pi3?

-J

Adrian Rosebrock September 18, 2017 at 2:21 pm # REPLY

Hi Jacques — please see my reply to “Flávio”. I haven’t yet tried the code on the Raspberry Pi but will be documenting my experience with it in a future blog post.

amitoz September 18, 2017 at 3:20 pm # REPLY

Hey Adrian,

Once we have detected an object, how difficult you think will it be to segment it in real time using deep learning? Share ur insight pls.

Adrian Rosebrock September 20, 2017 at 7:26 am # REPLY

Object detection and object segmentation are two totally different techniques. You would need to use a deep learning network that was trained to perform object segmentation. Take a look at DeepMask.

Kamel Rush September 18, 2017 at 3:46 pm # REPLY

Hi,

I tried to run the code, but got this:

File "C:\ProgramData\Miniconda33\lib\site-packages\imutils\convenience.py", line 69, in resize
(h, w) = image.shape[:2]
AttributeError: 'NoneType' object has no attribute 'shape'

Adrian Rosebrock September 20, 2017 at 7:25 am # REPLY

It sounds like OpenCV cannot access your webcam. I detail common causes for NoneType errors in this blog post.

Leopard Li September 20, 2017 at 9:06 pm # REPLY

Hi, have you resolved this problem? I got this problem too.
But when I changed src=1 to src=0 in line "vs = VideoStream(src=1).start()", it just worked!
Hope this could be helpful to you if it still bothers you.

Adrian Rosebrock September 21, 2017 at 7:14 am # REPLY

Thank you for mentioning this. I used `src=1` because I have two webcams hooked up to my system. Most people should be using `src=0` as they likely only have one webcam. I will update the blog post.

Enjoy September 28, 2017 at 9:02 am # REPLY

you can try change Line35 :vs = VideoStream(src=0).start() to vs = VideoStream(usePiCamera=args[“picamera”] > 0).start()

and add ap.add_argument(“-pi”, “–picamera”, type=int, default=-1, help=”whether or not the Raspberry Pi camera should be used”) after Line 14

it‘s work for me

Abhi October 1, 2017 at 12:34 am # REPLY

I get the AttributeError: ‘NoneType' object has no attribute ‘shape error as well and I tried the solution recommended by Enjoy (since I am getting this error with src=0) but the code does not run on my pi3. Every time run this code the OS crashes and pi reboots. Not sure what I am doing wrong here. Any help is appreciated.

Adrian Rosebrock October 2, 2017 at 9:47 am # REPLY

Are you using a USB camera or the Raspberry Pi camera module? Please make sure OpenCV can access your webcam. I detail these types of NoneType errors and why they occur in this post.

I'll also be doing a deep learning object detection + Raspberry Pi post later this month.

Abhi October 4, 2017 at 8:35 pm #

I am using the raspberry pi camera. And I can access the camera fine, since I tested it with your pi home surveillance code.

Abhi October 4, 2017 at 9:16 pm #

Also I forgot to mention that I tried the following from your unifying pi camera and cv2 video capture post with the appropriate argument.

# initialize the video stream and allow the cammera sensor to warmup
vs = VideoStream(usePiCamera=args["picamera"] > 0).start()

Adrian Rosebrock October 6, 2017 at 5:12 pm #

Hi Abhi — thank you for the additional comments. Unfortunately, without direct access to your Raspberry Pi I'm not sure what the exact issue is. I'll be covering object detection using deep learning on the Raspberry Pi the week of October 16th. I would suggest checking the code once I post the tutorial and see if you have the same error.

DreamChaser October 8, 2017 at 11:36 pm # REPLY

Thanks for the post! I was having the same 'NoneType' error. I changed the camera source but that didn't fix it. I added your argument update, along with adding –pi=1 to the command line and it worked. Thanks to the author (and everyone else who have posted) – it's great to have help when you start.

Adrian Rosebrock October 9, 2017 at 12:18 pm # REPLY

Thanks for the comment. I'll be covering how to perform real-time object detection with the Raspberry Pi here on the PyImageSearch blog in the next couple of weeks.

zakizadeh September 18, 2017 at 3:55 pm # REPLY

hi .
i want to get position of Specified object in image . all examples are about multi object detection . but i want to get position of only one object

. for example i want to get position of a book in image , not all object in image . only one of them . how can i do that ?

Adrian Rosebrock September 20, 2017 at 7:23 am # REPLY

First, you would want to ensure that your model has been trained to detect books. Then you can simply ignore all classes except the book class by checking only the index and probability associated with the book class. Alternatively you could fine-tune your network to apply detect books.

Kevin Lee September 19, 2017 at 1:19 am # REPLY

Thanks for great tutorial.

Is it running on the cpu? If so, is there a parameter we can change to gpu mode?

kevin

Adrian Rosebrock September 20, 2017 at 7:19 am # REPLY

This runs on the CPU. I don't believe it's possible to access the GPU via the Python bindings. I would suggest checking the OpenCV documentation for the C++ bindings.

Arvind Gautam September 19, 2017 at 1:27 am # REPLY

Hi Adrian .

Its really a great tutorial .You are the Rock star of Computer Vision .

I have also implemented a Faster-RCNN with VGG16 and ZF network on my own Sports videos to detect logos in the video.I am getting good accuracy with both the networks,but I am able to processed only 7-8 frames/sec with VGG16 and 14-15 frames/sec with ZF network .To process the video in real time,I am skipping every 5th frame. I have compared the results in both the cases (without skipping frames and skipping every 5th frame) having almost same accuracy .Can you guide me that I am doing right thing or not ? What can be the optimal value of skipping the frame to process in real time without hurting the accuracy.

Adrian Rosebrock September 20, 2017 at 7:17 am # REPLY

There is no real "optimal" number of frames to skip — you are doing the skip frames correctly. You normally tune the number of skip frames to give you the desired frame processing rate without sacrificing accuracy. This is normally done experimentally.

Luke Cheng September 19, 2017 at 2:32 am # REPLY

Hi I'm just curious how you trained your caffe model because I feel like the training process you used could be really good. thanks!

Adrian Rosebrock September 20, 2017 at 7:16 am # REPLY

Please see my reply to "Thang".

David Killen September 19, 2017 at 8:34 am # REPLY

This is very interesting, thank you. Unless I missed it, you aren't using prior and posterior probabilities across frames at all. I appreciate that if an object doesn't move then there is no more information to be extracted but if it were to move slightly but change because of rotation or some other movement then there is some independence and the information can be combined. We can see this when you turn the swivel-chair; the software loses track of it when it's face on (t=28 to t=30). Is this something that can be done or is it too difficult?

PS Can you explain why the human-identification is centred correctly at the start of the full video but badly off at the end please? It's almost as if the swivel chair on the left of the picture is pushing the human-box off to the right, but I can't see why it would do that.

Adrian Rosebrock September 20, 2017 at 7:13 am # REPLY

I'm only performing object detection in this post, not object tracking. You could apply object tracking to detected objects and achieve a smoother tracking. Take a look at correlation tracking methods.

As for the "goodness" of a detection this is based on the anchor points of the detection. I can't explain the entire Single Shot Detector

(SSD) framework in a comment, but I would suggest reading the original paper to understand how the framework is used. Please see the first blog post in the series for more information.

Jacques September 19, 2017 at 2:53 pm # REPLY

I ran it on my Pi3 last night. works nicely! Each frame takes a little over a second to classify. The rate is quite acceptable. Cool work and looking forward to any optimisations that you think will work..

How much do you think rewriting the app in C++ will increase the performance on the Pi? I know CV is C/C++, but I am keen to profile the diff in a purely compiled language.

Adrian Rosebrock September 20, 2017 at 7:09 am # REPLY

In general you can expect some performance increases when using C/C++. Exactly how much depends on the algorithms that are being executed. Since we are already using compiled OpenCV functions the primary overhead is the function call from Python. I would expect a C++ program to execute faster but I don’t think it will make a substantial difference.

Hubert de Lassus September 19, 2017 at 8:45 pm # REPLY

Great example code! Thank you. How would you modify the code to read an mp4 file instead of the camera?

Adrian Rosebrock September 20, 2017 at 7:00 am # REPLY

You would swap out the VideoStream class for a FileVideoStream.

Thang September 20, 2017 at 2:49 am # REPLY

Many thanks, but can you show me how to program trained file as in your project you used MobileNetSSD_deploy.caffemodel file.

Adrian Rosebrock September 20, 2017 at 6:58 am # REPLY

As I mention in the previous post the MobileNet SSD was pre-trained. If you're interested in training your own deep learning models, in particular object detectors, you'll want to go through the ImageNet Bundle of Deep Learning for Computer Vision with Python.

memeka September 20, 2017 at 5:55 am # REPLY

Hi Adrian,

Thanks for the great series of articles.
I've tried this on an Odroid XU4 (which is more powerful than the RPi – better CPU, better GPU, USB3) – with OpenCV compiled with NEON optimizations and OpenCL enabled (Odroid XU4 has OpenCL working, and GPU in theory should reach 60GFlops).

Do you know if OpenCL is actually used by net.forward()? It would be interesting to benchmark GPU vs GPU if OpenCL is indeed used.

I was able to run detection at 3fps (3.01 fps to be exact :D) with no optimizations and 640×480 resolution (no resize in the code), but I am confident I can reach >5fps with some optimizations, because:
* I have my board underclocked to 1.7Ghz (stock is 2 Ghz, but I can try overclocking up to 2.2 Ghz)
* I think I/O was the bottleneck, since even underclocked, CPU cores were used ~60%; adding some threading and buffering to the input should speed things up
* to remove some delay from GTK, I used gstreamer output to tcpsink, and viewed the stream with VLC. This would also work great in the "security camera" scenario, where you want to see a stream remotely over the web.
(PS: with gstreamer – from command line – I can actually use the hw h264 encoder in the odroid; but the exact same pipeline – well, except the appsink detour – is not working in opencv python; this would be useful to save the CPU for detection and still have h264 streaming, IF I can make it work…)

I can't wait to see your optimizations for the RPi, I'll keep you posted with the framerate I can get on my Odroid 🙂

Adrian Rosebrock September 20, 2017 at 6:56 am # REPLY

I'm not sure if OpenCL is enabled by default, that's a good question. And thanks for sharing your current setup! I'll be doing a similar blog post as this one for the Raspberry Pi in the future — I'll be sure to read through your comments and share optimizations and discuss which ones did or did not work (along with an explanation as to why). Thanks again!

memeka September 20, 2017 at 9:02 am # REPLY

So I was wrong – webcam was not the bottleneck. Even with threading, I still get 3fps max. I timed and indeed net.forward() takes 300ms. So the only way I may speed this up is getting the CPU to 2 or 2.2Ghz, and trying to write it in C++…

Tom September 21, 2017 at 4:21 pm # REPLY

For comparison:
Running the script from this blog post gave 0.4 fps on Raspberry Pi 3.
Demo that comes with Movidius Compute Stick running SqueezeNet gives 3 fps, though having inference in separate thread from video frames display assures a nice smooth experience. Just mind that it does not support Raspbian Stretch yet, so use archived card img's (that's due to built in Python 3.4 vs 3.5).

Adrian Rosebrock September 22, 2017 at 8:57 a… # REPLY

Thanks for sharing Tom!

memeka September 21, 2017 at 2:08 am # REPLY

With C++, as expected, performance is very similar.
I've tried using Halide dnn, but CPU target didn't really get an output (I lost patience after >10s), and OpenCL target resulted in a crash due to some missing function in some library…

So 3 fps is as best as I could get in the end.

With CPU at 2GHz, it scales it down to 1.8Ghz due to heat.
But still, cores don't get used 100% – any idea why? As you can see from here: https://imgur.com/a/D9tdp max usage stays just below 300% from the max 400%, and no core gets used more than 80% – do you know if this is an OpenCV thing?

Ldw September 21, 2017 at 10:40 pm #

REPLY

Hi Adrian, I tried running the code and got this : AttributeError: 'module' object has no attribute 'dnn'
Any ideas what's the issue? Thanks in advance!

Ldw September 21, 2017 at 11:29 pm #

REPLY

Just to add on I've downloaded OpenCV 3.3's zip file here. Did i download at the wrong place, or did i download it the wrong way? What i did was just download the zip file from that website and added into my Home from the archive manager. Sorry for bothering!

Adrian Rosebrock September 22, 2017 at 8:55 am #

REPLY

Once you download OpenCV 3.3 you still need to compile and install it. Simply downloading the .zip files is not enough. Please see this page for OpenCV install instructions on your system.

Roberto Maurizzi September 22, 2017 at 4:02 am #

REPLY

Hi Adrian, thanks for your many interesting and useful posts!

I missed this post and I did try to adapt your previous post to do what you did here by myself, searching docs and more examples, to read frames from a stream coming from an ONVIF surveillance camera that streams rtsp h264 video.

I'm having trouble with rtsp part: on Windows I get warnings from the cv2.VideoCapture() call ([rtsp @ 000001c517212180] Nonmatching transport in server reply – same from your imutils.VideoStream), on linux I get nothing but the capture isn't detected as open.

Any advice about this? I already tried to check my ffmpeg installation, copied it to the same folder from which my python's loading opencv dlls and if I try ffplay it can stream from the camera (after a few warnings: "'circular_buffer_size' option was set but it is not supported on this build (pthread support is required)")

I was able to use ffserver to convert the rtsp stream from rtsp/h264 to a mjpeg stream, but it consumes more CPU than running the dnn… any advice?

Roberto

Roberto Maurizzi September 22, 2017 at 4:36 am # REPLY

Update: I suspect the reason is explained here: http://answers.opencv.org/question/120699/can-opencv-310-be-set-to-capture-an-rtsp-stream-over-udp/

Adrian Rosebrock September 22, 2017 at 8:52 am # REPLY

It's been a long time since I've tried RTSP. I've made a note to cover this in a future blog post. Thanks for the comments!

Roberto Maurizzi September 22, 2017 at 10:21 am # REPLY

I continued my research on a solution or workaround and found OpenCVmerged this patch about 3 days ago: https://github.com/opencv/opencv/pull/9292/files
I'll have to find some nightly builds and test it again 🙂

Adrian Rosebrock September 22, 2017 at 11:29 am#

Thanks for sharing, Roberto!

Lin September 22, 2017 at 5:06 am # REPLY

Hi Adrian,

Yesterday I leave a reply about the error like:

AttributeError: 'NoneType' object has no attribute 'shape'

But today I read the board carefully, found that someone has encountered the same problem.
And I already resolve the problem .

Thanks.

Adrian Rosebrock September 22, 2017 at 8:51 am # REPLY

Thanks for the update, Lin!

Adrian Rosebrock September 22, 2017 at 8:58 am # REPLY

Change `src=0` and then read this post on NoneType errors.

Jorge September 23, 2017 at 10:48 pm # REPLY

Hi Adrian. Thanks for your great job. Im thinking about the possibility of applying the recognition only for people in real time on the video stream of four security cameras in mosaic. It would be like having someone watching four cameras at a time and triggering alerts if people are detected in x consecutive frames. Maybe send an email with the pix. What do you think about this and how can be implemented?

Adrian Rosebrock September 24, 2017 at 8:43 am # REPLY

You would need to have four video cameras around the area you want to monitor. Depending on how they are setup you could stream the frames over the network, although this would include a lot of I/O latency. You might want to use a Raspberry Pi on each camera to do local on-board processing, although I haven't had a chance to investigate how fast this code would run on the Raspberry Pi. You also might want to consider doing basic motion detection as a first step.

Jorge September 24, 2017 at 9:38 am # REPLY

I was referring to using the mosaic of the four cameras as a single image and running the CNN detector of this post on that image only for the person category. Do you think it would be possible? And what observation or suggestion would you make?

Adrian Rosebrock September 24, 2017 at 10:03 am # REPLY

Ah, got it. I understand now.

Yes, that is certainly possible and would likely work. You might get a few false detections from time to time, such as if there are parts of a person moving in each of the four corners and a classification is applied across the borders of the detections. But that is easily remedied since you'll be constructing the mosaic yourself and you can filter out detections that are on the borders.

So yes, this approach should work.

Jorge September 24, 2017 at 10:18 am #

Thanks for the feedback Adrian!!!

Enjoy September 24, 2017 at 12:56 am # REPLY

WHY ?

Traceback (most recent call last):
…
(h, w) = image.shape[:2]
AttributeError: 'NoneType' object has no attribute 'shape'

Adrian Rosebrock September 24, 2017 at 8:41 am # REPLY

Please see my reply to "Lin" above. Change `src=0` in the `VideoStream` class. I've also updated the blog post to reflect this change.

Aleksandr Rybnikov September 24, 2017 at 4:06 am # REPLY

Hi Adrian!
Thanks for the another great post and tutorial!
As you've maybe noticed, bounding boxes are inaccurate – they're very wide comparing to the real size of object. It happens due to the following thing: you're using blobFromImage finction, but it takes a central crop from the frame. And this central crop goes to the ssd model. But later you multiply unit box coordinates by full frame size. To fix it you can simply pass cv.resize(frame, (300,300)) as first parameter of blobFromImage() and all will be ok

Adrian Rosebrock September 24, 2017 at 8:41 am # REPLY

Thank you for pointing this out, Aleksandr! I've updated the code in the blog post. I'll also make sure to do a tutorial dedicated to the parameters of `cv2.blobFromImage` (and how it works) so other readers will not run into this issue as well. Thanks again!

Enjoy September 24, 2017 at 10:44 am # REPLY

OpenCV: out device of bound (0-0): 1
OpenCV: camera failed to properly initialize!

Adrian Rosebrock September 24, 2017 at 12:26 pm # REPLY

Please double-check that you can access your webcam/USB camera via OpenCV. Based on your error messages you might have compiled OpenCV without video support.

RicardoGomes September 24, 2017 at 9:46 pm # REPLY

Nice tutorial, I managed to make it run in rpi but it detects objects with error, my television appeared as a sofa and the fan like chair. What could it be?

RicardoGomes September 24, 2017 at 9:48 pm # REPLY

In rpi it was very slow, would it need some kind of optimization?

Adrian Rosebrock September 26, 2017 at 8:32 am # REPLY

I'll be covering deep learning-based object detection using the Raspberry Pi in the next two blog posts. Stay tuned!

Henry September 25, 2017 at 2:27 am # REPLY

Hi Adrian,

Nice tutorial, thank you so much.

Besides, can the same code accept rtsp/rtmp video stream?
If the answer is "No", do you know any python module that can support rtsp/rtmp as video stream input? Many thanks.

Adrian Rosebrock September 26, 2017 at 8:29 am # REPLY

This exact code couldn't be used, but you could explore using the `cv2.VideoCapture` function for this.

Sydney September 26, 2017 at 11:16 am # REPLY

Hie Adrian. Any pointers on how i can implement this as a web based application?

Adrian Rosebrock September 28, 2017 at 9:28 am # REPLY

Are you trying to build this as a REST API? Or trying to build a system that can access a user's webcam through the browser and then apply object detection to the frame read from the webcam?

Sydney September 28, 2017 at 12:14 pm # REPLY

I want to be able to upload a video using a web interface, then perform object detection on the uploaded video showing results on the webpage.

Adrian Rosebrock September 28, 2017 at 12:29 REPLY

Can you elaborate on what you mean by "showing the results"? Do you plan on processing the video in the background and then once it's done show the output video to the user? If you can explain a little more of what exactly you're trying to accomplish and what your end goal is myself and others can try to provide suggestions.

sydney September 30, 2017 at 10:45 am #

Ok. I need to run the application on google cloud platform and provide an interface for users to upload their own videos.

Adrian Rosebrock October 2, 2017 at 9:57 am #

Have users upload the videos and then bulk process the videos in the background and save the annotations. You can either then (1) draw the bounding boxes on the resulting images or (2) generate a new video via cv2.VideoWriter with the bounding boxes drawn on top of them.

memeka September 27, 2017 at 12:55 am #

REPLY

Hi Adrian,

As mentioned above, I am getting 3fps on detection (~330ms in net.forward()), and I'm saving the output via a gstreamer pipeline (convert to h264, then either store in file, or realtime streaming with hls).

In order to improve the output fps, I decided to read a batch of 5 frames, do detection on the first, then apply the boxes and text to all 5 before sending them to the gst pipeline.

Using cv2.VideoCapture, I end up with around the half the previous framerate (so an extra 300-350ms spent in 4xVideoCapture.read()), which I am not very happy with.

So I decided to modify imutils.WebcamVideoStream to do 5 reads, and I have (f1, f2, f3, f4, f5) = MyWebcamVideoStream.read() – using this approach I only lose ~50ms and I can get close to 15fps output. However, the problem here is that the resulting video has frames out of order. I tried having the 5 read() protected by a Lock, but without much success.

Any suggestion on how I can keep the correct frame order with threaded WebcamVideoStream?

Thanks.

Adrian Rosebrock September 27, 2017 at 6:40 am #

REPLY

The `WebcamVideoStream` class is threaded so I would suggest using a thread-safe data structure to store the frames. Something like Python's `Queue` class would be a really good start.

memeka September 27, 2017 at 8:28 am #

REPLY

Thanks Adrian,
I figured out what the problem was: reading 5 frames was actually taking longer than net.forward(), so WebcamVideoStream was returning the same 5 frames as before; by reducing the batch to 4 frames, and also synchronising the threads, I managed to get 2.5 fps detected + extra 3 frames for each detection for a total of 10fps webcam input/ pipeline output.

Adrian Rosebrock September 27, 2017 at 8:49 a

REPLY

Congrats on resolving the issue! The speed your getting is very impressive, I'll have to play around with the Odroid in the future 🙂

memeka September 28, 2017 at 4:58 am #

Thanks Adrian

Since there are many here who, like me, would like to use this for a security camera, I would like to share my end script, maybe somebody else would find it useful: http://paste.debian.net/988135/
It reads the input from a .json file, such as: http://paste.debian.net/988136/

* gst_input defines the source (doesn't actually have to be gst, "0" will work for /dev/video0 webcam)
* gst_output defines the output
* batch_size defines the number of frames read at once. On my system, 4 was optimal (reading 4 frames took similar amount of time to detection on 1 frame)
* base_confidence defines the minimum confidence for an object to be considered
* detect_classes contains "class_name":"confidence" that you want to track (e.g. 'person'). Note that

confidence here can be lower than "base_confidence"
* detect_timeout defines a time (in s) since a class is considered "detected" again. E.g. if detect_time = 10s, and same class was detected 2s ago, it won't be considered "detected" again
* detect_action contains a script to be executed on detection. Script needs to have as input "class", "confidence", and "filename"

The output video (e.g. the HLS stream in the json example above) contains all classes detected w/ surrounding boxes and labels. Of course, detection is done only on the 1st frame out of batch_size, but all frames have the boxes and labels.
On detecting a class specified in "detect_classes", the script saves the image in a 'detected' folder (in the format timestamp_classname.jpg), then executes the action specified.
In my case, I can always watch the stream online and see what the camera detects, but I can choose to have an action taken (e.g. send email/notification with a picture) when certain objects are detected.
With ~330ms net.forward() and a batch of 4, I can achieve reliably 10fps output.

If somebody has suggestions on how I can improve this, please leave a comment 🙂

Adrian Rosebrock September 28, 2017 at 8:58 am#

Awesome, thanks for sharing memeka!

Ying September 27, 2017 at 12:34 pm # REPLY

hi Adrian,

thank you so much for your tutorial! I am a big fan!

I was wondering can I use pre recorded video clips instead of live camera to feed the video stream? Could you suggest how I can achieve this please?

Adrian Rosebrock September 28, 2017 at 9:12 am # REPLY

Yes. Please see my reply to “Hubert de Lassus”.

tiago September 27, 2017 at 4:36 pm # REPLY

How can I provide the –prototxt and –model direct argument in source code?

args = vars(ap.parse_args())

Adrian Rosebrock September 28, 2017 at 9:06 am # REPLY

Please read up on command line arguments. You need to execute the script via the command line — that is where you supply the arguments. The code DOES NOT have to be edited.

Foobar October 1, 2017 at 8:30 pm # REPLY

When the network was trained did the training data have bounding boxes in it? Or was it trained without and OpenCV can just get the bounding boxes by itself?

Adrian Rosebrock October 2, 2017 at 9:37 am # REPLY

When you train an object detector you need the class labels + the bounding boxes. OpenCV cannot generate the bounding boxes itself.

Foobar October 2, 2017 at 8:22 pm # REPLY

Are the bounding boxes drawn on the training data or is there some other method of doing it?

Adrian Rosebrock October 3, 2017 at 11:05 am # REPLY

The bounding boxes are not actually drawn on the raw image. The bounding box (x, y)-coordinates are saved in a separate file, such as a .txt, .json, or .xml file.

Foobar October 3, 2017 at 4:26 pm #

Thank you Adrian for your help.

Jussi Wright October 5, 2017 at 4:38 am #

REPLY

Hi,

I got the detector to work on the video with the help of your another blog (https://www.pyimagesearch.com/2017/02/06/faster-video-file-fps-with-cv2-videocapture-and-opencv/).

But I have a couble of supplementary questions.
1. How can I easily get a saved video where recognizations are displayed (Can I use cv2.imwrite)?
2. How can I remove the unnecessary labels I do not need (cat, bottle etc). Removing only the label name produces an error code.
3. How do I adjust the code so that only the detections with an accuracy of more than 70-80% are displayed.
4. Do you know ready models for identifying road signs, for example?

Jussi October 5, 2017 at 6:18 am #

REPLY

Ok, I found a point for adjusting the accuracy of the detection:
ap.add_argument("-c", "–confidence", type=float, default=0.2 <—

Also I found your blog (https://www.pyimagesearch.com/2016/02/22/writing-to-video-with-opencv/), but I could not find the right settings for me… I get error:
…argument -c/–codec: conflicting option string: -c

Adrian Rosebrock October 6, 2017 at 5:06 pm #

REPLY

You need to update your command line arguments. If you have conflicting options, change the key for the command line arguments. I would suggest reading up on command line argumentsbefore continuing.

To address your other questions:

1. Answered from your comment.
2. You cannot remove just the label name. Check the index of the label (i.e., `idx`) and ignore all that you are uninterested in.
3. Provide `--confidence 0.7` as a command line arguments.

4. It really depends on the road signs. Most road signs are different in various countries.

chetan j October 6, 2017 at 3:55 am # REPLY

hi,
great work, nice tutorial

just one question, i tried to run this code in my system, it works nice but have delay 5 to 8 sec to detect objects.

how to overcome this problem.

Adrian Rosebrock October 6, 2017 at 4:54 pm # REPLY

What are the specs of your system? 5-8 seconds is a huge delay. It sounds like your install of OpenCV may not be optimized.

chetan j October 9, 2017 at 3:15 am # REPLY

hi,
im using reaspbeyy pi 3- code runs fine but have delay of 5 to 8 sec.

how to resolve this problem

Adrian Rosebrock October 9, 2017 at 12:14 pm # REPLY

I will be discussing optimizations and how to improve the frames per second object detection rate on the Raspberry Pi in future posts. I would suggest starting herewith a discussion on how to optimize your OpenCV + Raspberry Pi install.

inayatullah October 16, 2017 at 3:44 am # REPLY

I have reimplemented the same, but with using sddcaffe for python.When i detector is applied on every second frame then on my system I can get 12 to 14 frames per second. My code is available here https://github.com/inayatkh/realTimeObjectDetection

Adrian Rosebrock October 16, 2017 at 12:19 pm # REPLY

Thanks for sharing, Inayatullah!

Chetan J October 7, 2017 at 9:51 am # REPLY

I'm using Raspberry Pi 3,
Code runs fine but slower operation

vinu October 9, 2017 at 7:29 pm # REPLY

hi adrin
how can i assign a unique id number to each and every human object

Adrian Rosebrock October 13, 2017 at 9:16 am # REPLY

What you're referring to is called "object tracking". Once you detect a particular object you can track it. I would suggest researching correlation trackers. Centroid-based tracking would be the easiest to implement.

Justin October 9, 2017 at 10:19 pm # REPLY

Hi Adrian.
Thank you for this post.
I followed you and made it with Rpi3
But too slow…
How can I fix it?

and when I started real_time_~.py
I got this message.
[INFO] loading model…
[INFO]starting video stream…

** (Frame:3570): WARNING **: Error retrieving accessibility bus address: org.freedesktop.DBus.Error.ServiceUnknown: The name org.a11y.Bus was not provided by any .service files

what should I do??

Adrian Rosebrock October 13, 2017 at 9:12 am #

REPLY

Please see this post on optimizing the Raspberry Pi for OpenCV. The commenter "jsmith" also has a solution.

For what it's worth, this is NOT an error message. It's just a warning from the GUI library and it can be ignored.

Dim October 10, 2017 at 1:25 am #

REPLY

First of all – thank you forbthia tutorial – very informative. Maybe i missed this but do you have any tutorials on real time custom object detection? I want to add additinal object that is not included in the trained model…

Adrian Rosebrock October 13, 2017 at 9:11 am #

REPLY

Hi Dim — I cover object detection in detail inside the PyImageSearch Gurus course. I would suggest starting there.

Mindfreak October 11, 2017 at 9:59 am #

REPLY

Great work sir.
but while I am trying to run code it gives me error:

AttributeError: module 'cv2' has no attribute 'dnn'

how to solve this error?
I am using OpenCV 3.2.0 version.
Thanks in advance..

Adrian Rosebrock October 13, 2017 at 8:56 am #

REPLY

The dnn module is only available in **OpenCV 3.3** and newer. Please upgrade to OpenCV 3.3 and you'll be able to run the code.

Mahsa October 12, 2017 at 8:32 am #

REPLY

Thank you for this awesome tutorial, this works quite nice on my laptop computer whereas it has too much delay on odroid (which I might try out the optimized opencv you've posted)

but Is there a way to retrain the exact model but with fewer classes?? since I only need two of those classes.

Adrian Rosebrock October 13, 2017 at 8:45 am # REPLY

You would need to apply fine-tuning to reduce the number of classes or you can just ignore the indexes of classes you don't care about. However, keep in mind that the total number of classes isn't going to significantly slow down the network. Yes, less classes means less computation — but there is a ton of computation and depth earlier in the network.

Shenghui Yang October 13, 2017 at 3:53 pm # REPLY

Hi Adrian

Thanks for the wonderful tutorial. I have a small question. I got an error when running codes:

AttributeError: 'module' object has no attribute 'dnn'

I have installed the opencv3.3.0, and it works. How can I deal with it?

Thank you.

Adrian Rosebrock October 14, 2017 at 10:38 am # REPLY

Hmm, I know you mentioned having OpenCV 3.3 installed but it sounds like you may not have it properly installed. What is the output o:

```
$ python
>>> import cv2
>>> cv2.__version__
```

Andrey October 16, 2017 at 12:37 pm # REPLY

This is very motivational post to try this technique. Thank you Adrian.

How difficult it would be to switch to TensorFlow instead?

Adrian Rosebrock October 16, 2017 at 12:54 pm # REPLY

TensorFlow instead of Caffe? That depends on the model. You would need a TensorFlow-based model trained for object detection. As far as I understand, the OpenCV loading capabilities of pre-trained TensorFlow models is still in heavy development and not as good as the Caffe ones (yet). For what it's worth, I'll be demonstrating how to train your own custom deep learning object detectors and then deploy them inside Deep Learning for Computer Vision with Python.

Andrey Cheremskuy October 17, 2017 at 9:04 am # REPLY

Thank you.

Adel October 18, 2017 at 6:32 pm # REPLY

thanks very much for the tutorial … how train the SSD for custome data like hand detection ?

Adrian Rosebrock October 19, 2017 at 4:45 pm # REPLY

I am covering how to train your own custom deep learning object detectors (such as Faster R-CNN and SSD) inside my book, Deep Learning for Computer Vision with Python.

Sunil Badak October 19, 2017 at 11:42 am # REPLY

hi Adrian,
we are doing a final year B.E project in which we need to give the movement to the Robot depending upon the object that Robot has detected , in such way that that Robot will approach the detected object. Any Idea how to achieve this?. Thanks

Adrian Rosebrock October 19, 2017 at 4:42 pm # REPLY

Congrats on doing your final B.E. project, that's very exciting. Exactly how you achieve this project is really dependent on your robot and associated libraries. Are you using a Raspberry Pi? If so, take a look at the GPIO libraries to help you get started.

John McDonald October 20, 2017 at 9:17 pm # REPLY

Adrian, this is amazing. But what if we want to detect something else besides a chair etc. How could we make our own detector?

Adrian Rosebrock October 22, 2017 at 8:36 am # REPLY

Hi John — I'm actually covering how to train your own deep learning-based object detectors inside Deep Learning for Computer Vision with Python. Be sure to take a look!

Darren October 22, 2017 at 7:13 am # REPLY

will this work on mobile phones? because im currently working with object detection also but im using mobile phones for it

Adrian Rosebrock October 22, 2017 at 8:22 am # REPLY

This code is for Python so you would need to translate it to the OpenCV bindings for the programming language associated with your phone, typically Java, Swift, or Objective-C.

Ying October 23, 2017 at 11:47 am # REPLY

Hi Adrian,

Can I use other caffe model to run this python code? e.g. yolov2, etc?

Adrian Rosebrock October 23, 2017 at 12:20 pm # REPLY

OpenCV 3.3's "dnn" module is still pretty new and not all Caffe models/layers are supported; however, a good many are. You'll need to test with each model you are interested in.

Justin October 23, 2017 at 12:15 pm #

REPLY

Hi Adrian! I'm back!
Thank you for the answer again.
Now, I'm trying to use this program for my school project.
I want to make a sushi detection machine.
So I need to have the pre-trained data(sushi images caffemodel).
How can I get it? How can I train and get my own data?
please let me know. Thank you
Have a good day.

Adrian Rosebrock October 23, 2017 at 12:18 pm #

REPLY

Hi Justin — I would like to refer you over to my book, Deep Learning for Computer Vision with Python where I discuss how to train your own custom deep learning classifiers in detail.

Win October 24, 2017 at 11:30 am #

REPLY

Hi i just want to ask what are the possible algorithms that you've used in doing it THANKS

Adrian Rosebrock October 24, 2017 at 2:49 pm #

Hi Win — this blog post was part of a two part series and I detailed MobileNet Single Shot Detectors (the algorithm used) in the prior week's blog post. You can access that blog post here: Object detection with deep learning and OpenCV.