NUS-ISS
Real Time Audio-Visual Sensing and Sense Making

Module 8 - Sense making from multi-modal audio-visual data, part 2

Dr. Tan Jen Hong
Lecturer & Consultant
Institute of System Science
National University of Singapore
issjht@nus.edu.sg

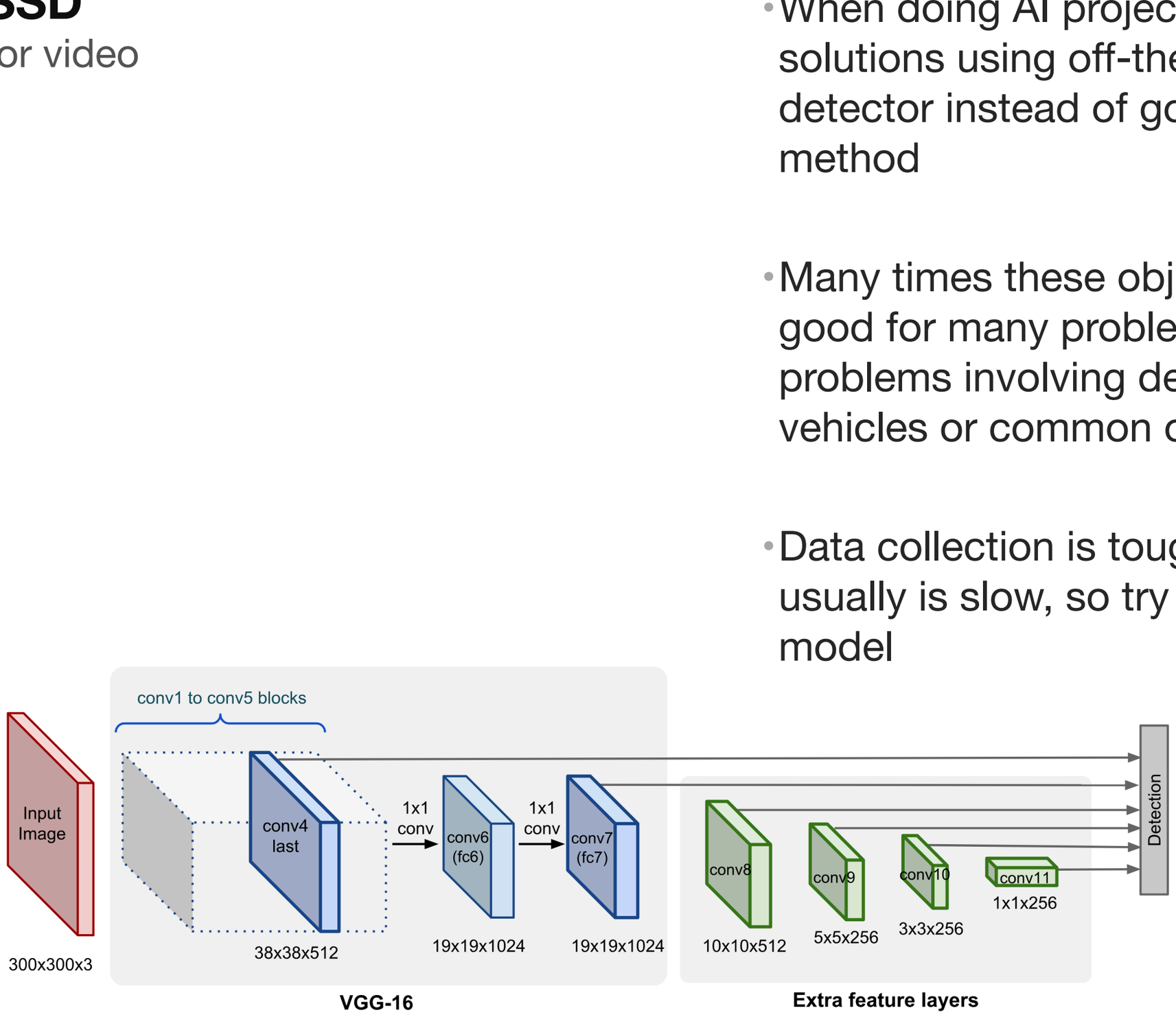
Using off-the-shelf object detector for video analysis

SSD
for video

- When doing AI projects, try to think of solutions using off-the-shelf object detector instead of going for fancy method

- Many times these object detectors are good for many problems, epsecially problems involving detection of human, vehicles or common objects in video

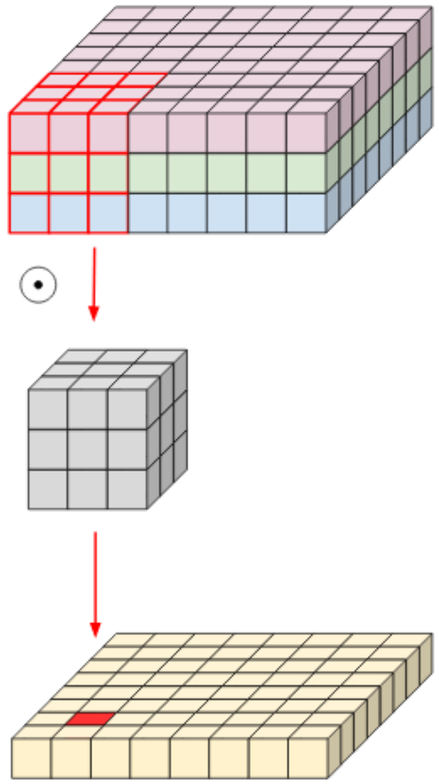
- Data collection is tough, fancy method usually is slow, so try to use trained model



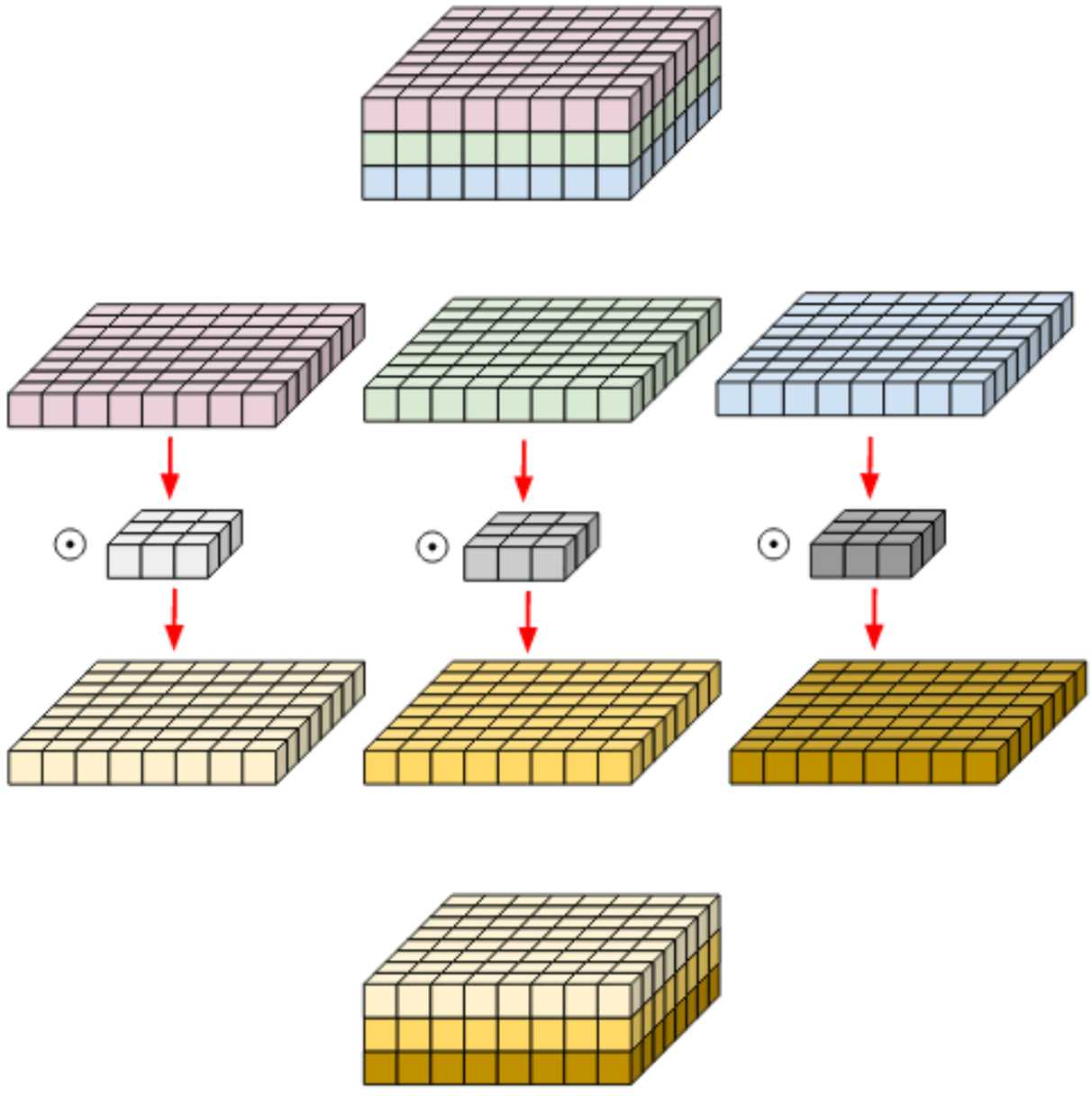
Source: https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html

Depthwise convolution

Normal convolution

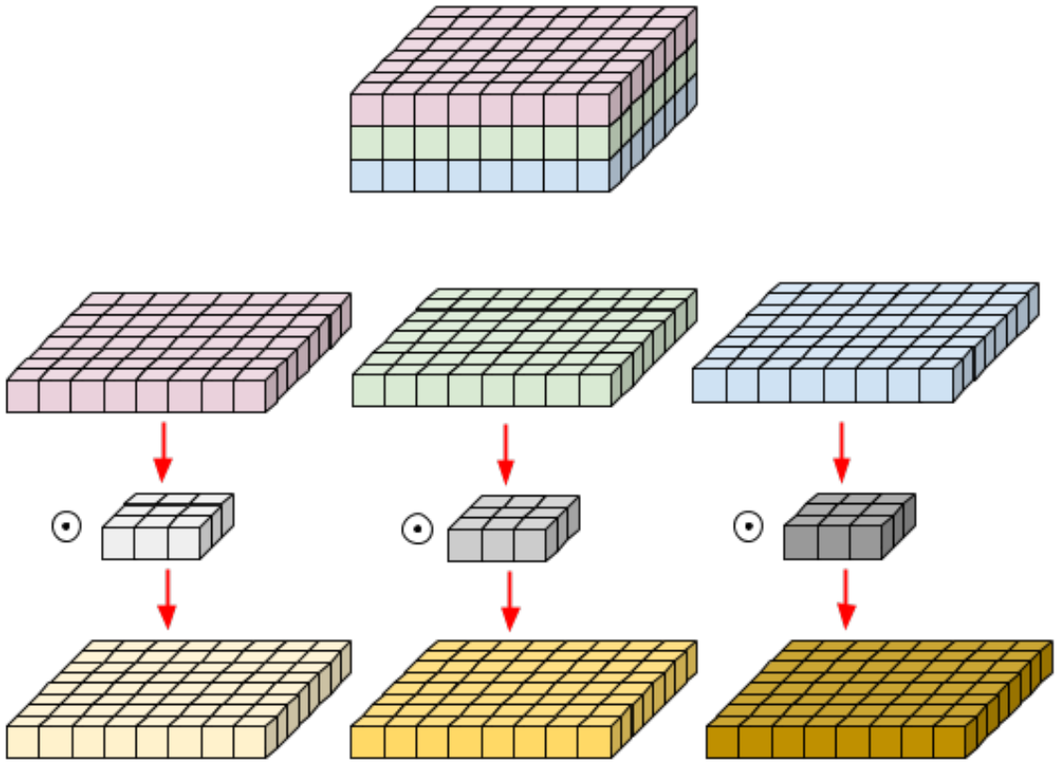


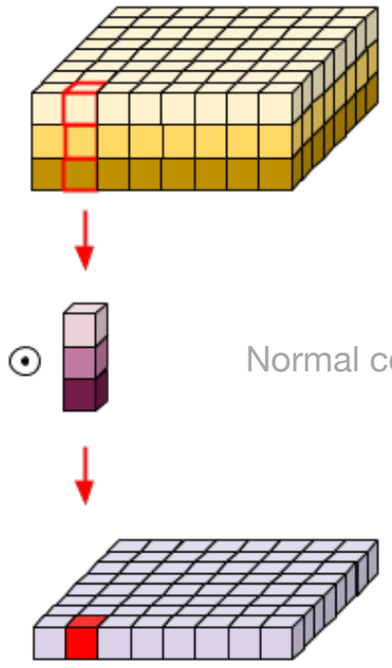
Depthwise convolution



Source: https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

Depthwise separable convolution



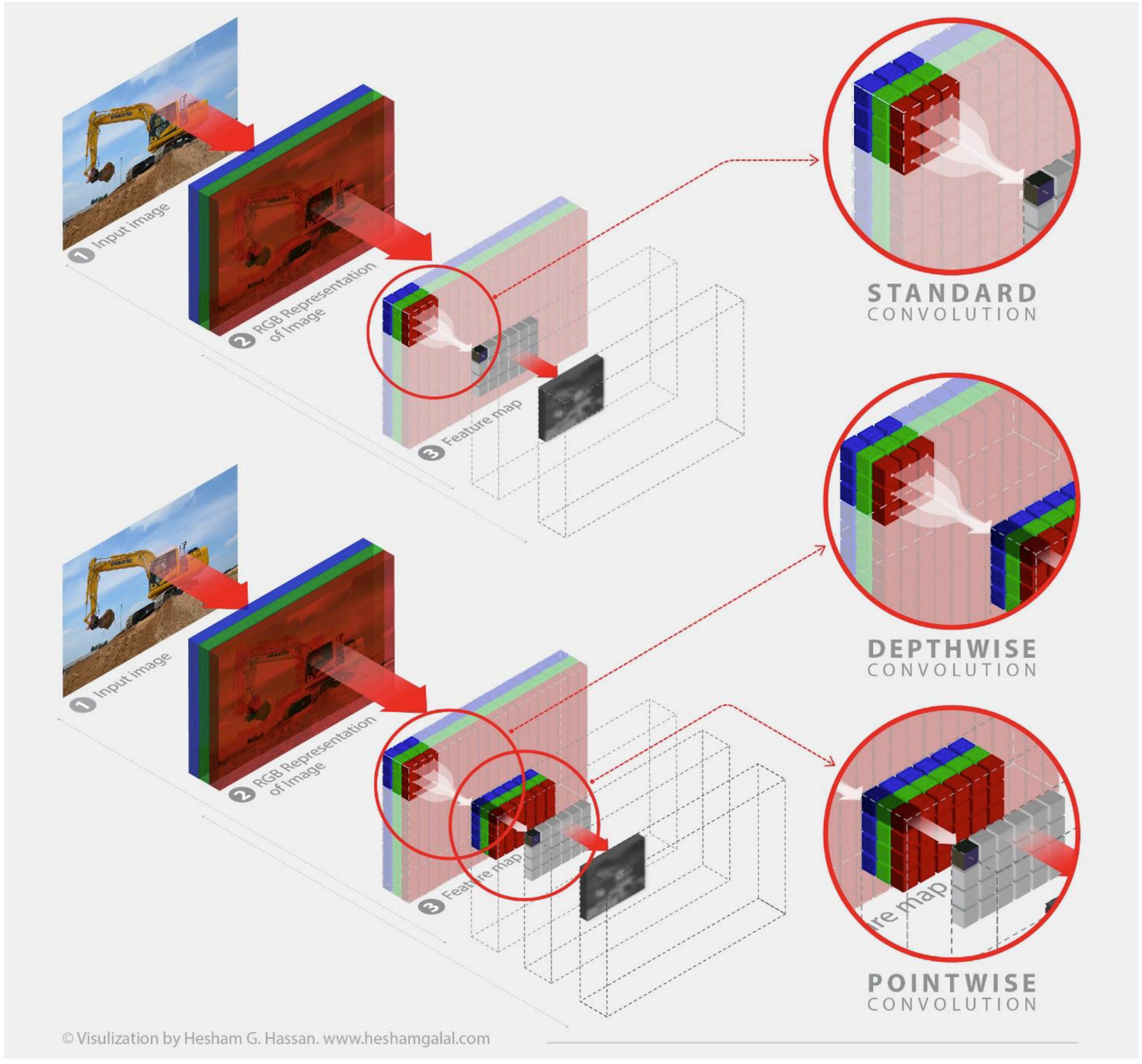


Normal convolution with kernel size (1,1)

Source: https://medium.com/@zurister/depth-wise-convolution-and-depth-wise-separable-convolution-37346565d4ec

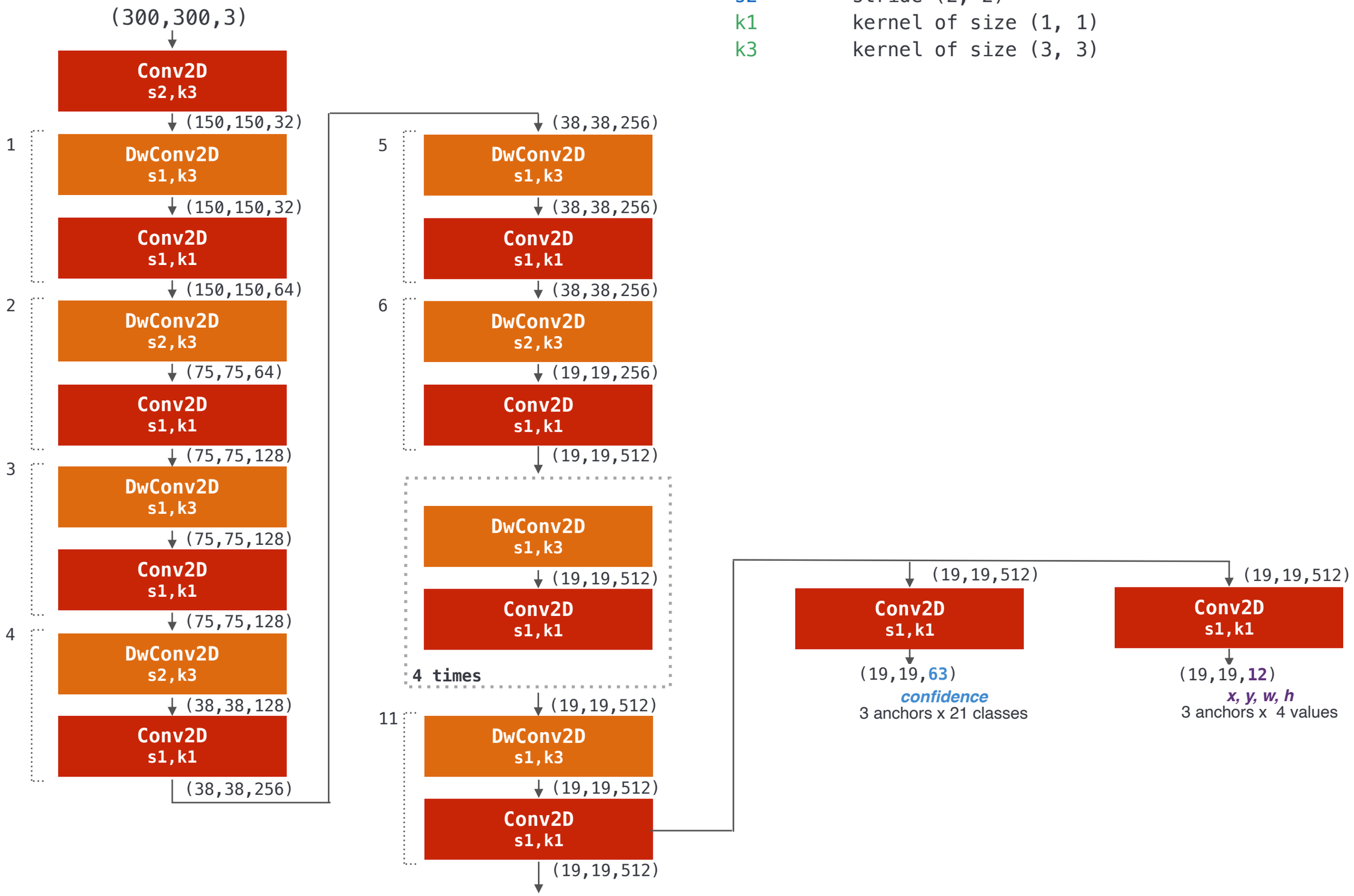
Depthwise separable convolution

Another look



Mobilenet SSD

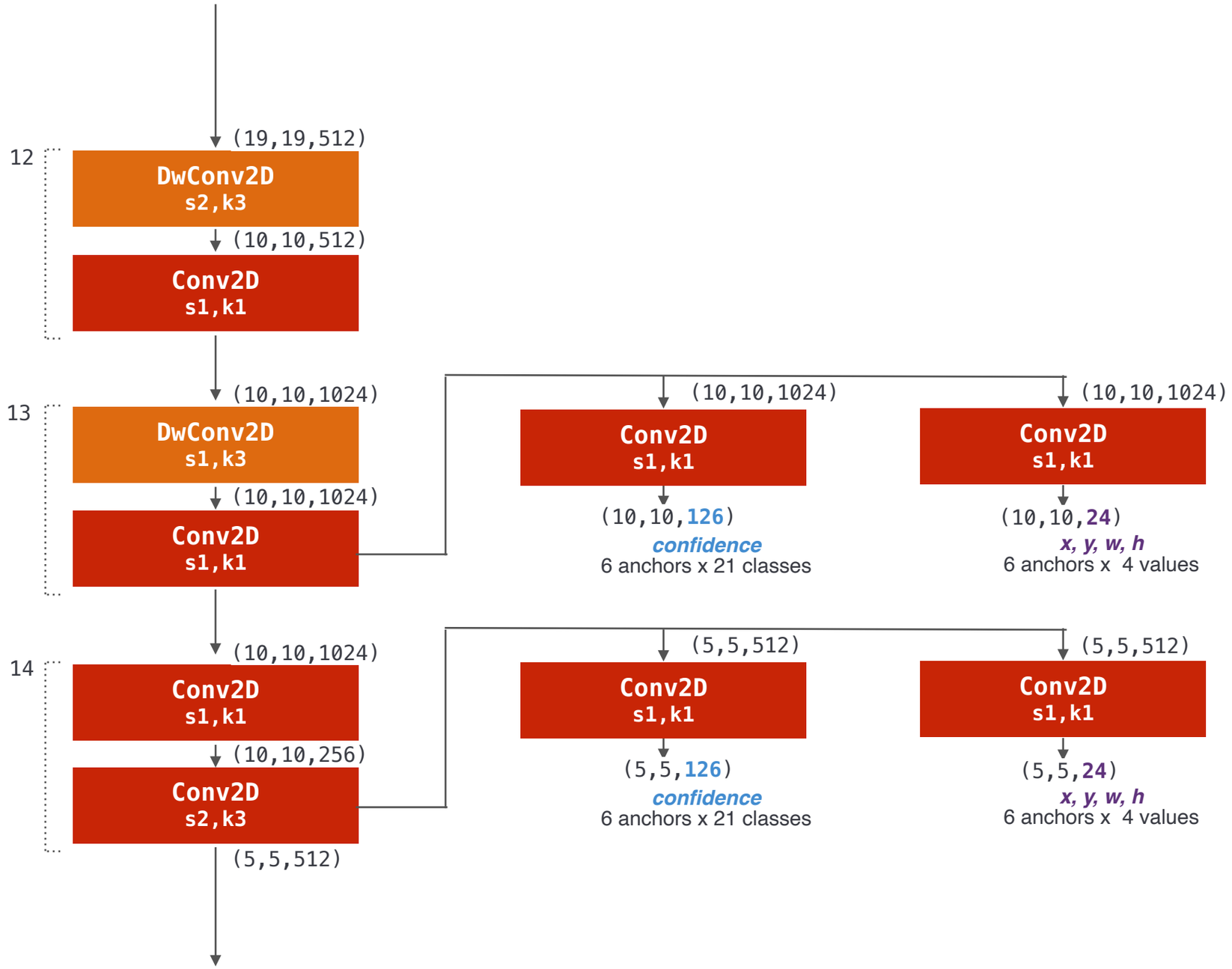
Net structure, part 1



Conv2D	2D convolution with padding and ReLU
DwConv2D	2D depthwise convolution with padding and ReLU
s1	stride (1, 1)
s2	stride (2, 2)
k1	kernel of size (1, 1)
k3	kernel of size (3, 3)

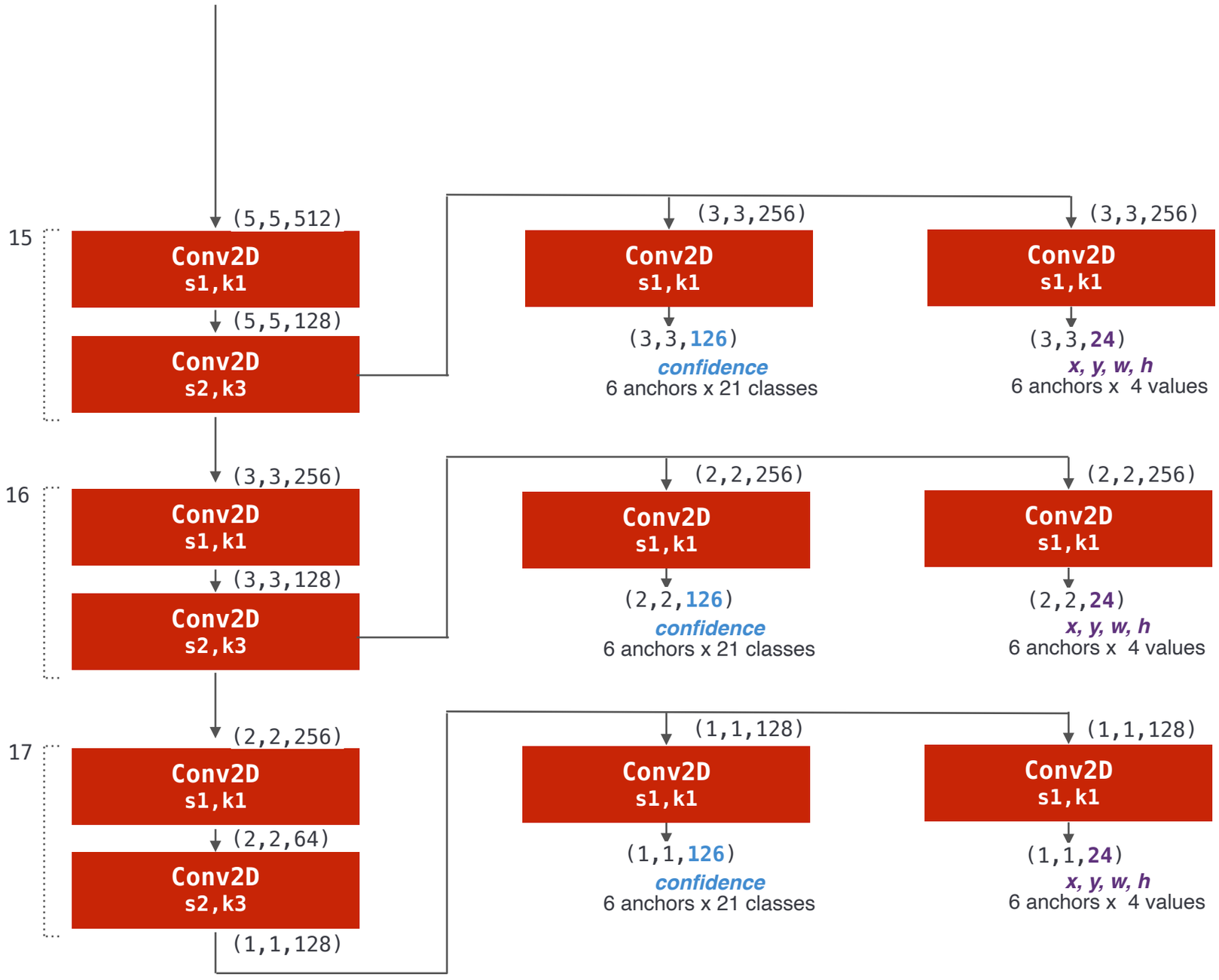
Mobilenet SSD

Net structure, part 2



Mobilenet SSD

Net structure, part 3



SSD
for video

- Approach: grab each frame, feed frame to SSD model, get and draw output, save video
- Use opencv to achieve, opencv version must be greater than 3.4.0
- Get objects in ironman trailer!

To start

setup a few things

```
> import cv2
> videopath       = 'ironman.mp4'
> outpath         = 'mssd_ironman.mp4'
> prototxt        = 'MobileNetSSD_deploy.prototxt'
> caffemodel      = 'MobileNetSSD_deploy.caffemodel'
> scoreThres      = 0.5
> 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'}
```

Some basic setup

Before the loop

- Load SSD and get the video frame rate, width and height before we loop through each frame

```
> net             = cv2.dnn.readNetFromCaffe(prototxt,
                                             caffemodel)

> vs              = cv2.VideoCapture(videopath)
> fps             = vs.get(cv2.CAP_PROP_FPS)                get frame rate (frame per second)
> W               = int(vs.get(cv2.CAP_PROP_FRAME_WIDTH))   get frame width
> H               = int(vs.get(cv2.CAP_PROP_FRAME_HEIGHT))  get frame height
> writer          = None
```

Analyze video
in a while loop

- To analyze video, we loop through each frame under a while loop
- We break the loop when there is no more frame to read, in this case the `grabbed` will be `False`

```
> while True:
      (grabbed,
       frame)     = vs.read()

      if not grabbed:
          break

      output      = frame.copy()
      blob        = cv2.dnn.blobFromImage(image=cv2.resize(frame,(300,300)),
                                          scalefactor=0.007843,
                                          size=(300, 300),
                                          mean=(127.5, 127.5, 127.5),
                                          swapRB=False,
                                          crop=False)
      ...
```

This resize is somehow required to give more accurate bounding boxes

1/127.5 = 0.007843

the SSD accepts 300x 300

subtract input from mean

No need to swap red and blue channel for this particular model

Analyze video
in a while loop

- The size of the blob is (1,3,300,300)

```
> while True:
      ...

      rows          = blob.shape[2]
      cols          = blob.shape[3]
      net.setInput(blob)
      pred          = net.forward()
```

Get the rows

Get the columns

Get the output. The output is a (1,1,n,7). n is the number of objects detected

```
      numOfObjects= pred.shape[2]

      for i in range(numOfObjects):
          confidence       = pred[0, 0, i, 2]
```

Get the number of objects detected

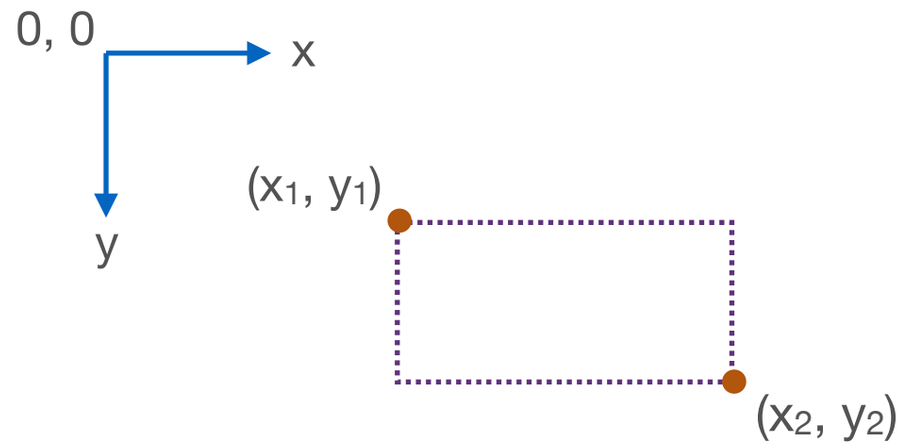
Get confidence score

```
          if confidence > scoreThres:
              classId      = int(pred[0, 0, i, 1])
              ...
```

If the confidence score exceed certain threshold, keep it and get the class id

Analyze video

in a while loop




```
> while True:
      ...
      for i in range(numOfObjects):
          ...
          if confidence > scoreThres:
              ...
              x1          = int(pred[0, 0, i, 3] * cols)
              y1          = int(pred[0, 0, i, 4] * rows)
              x2          = int(pred[0, 0, i, 5] * cols)
              y2          = int(pred[0, 0, i, 6] * rows)
```

the value of x1, y1, x2, y2 is in the range of 0 to 1. Scale these values in respect to the input size, which is 300 x 300

```
              hFactor     = H/300.0
              wFactor     = W/300.0
```

Get the scaling factor for each dimension

```
              x1          = int(wFactor*x1)
              y1          = int(hFactor*y1)
              x2          = int(wFactor*x2)
              y2          = int(hFactor*y2)
              x           = x1
              y           = y1
              w           = x2-x1
              h           = y2-y1
              ...
```

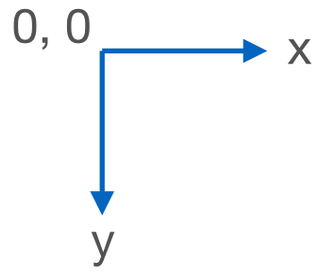
Get the actual x, y in the original video

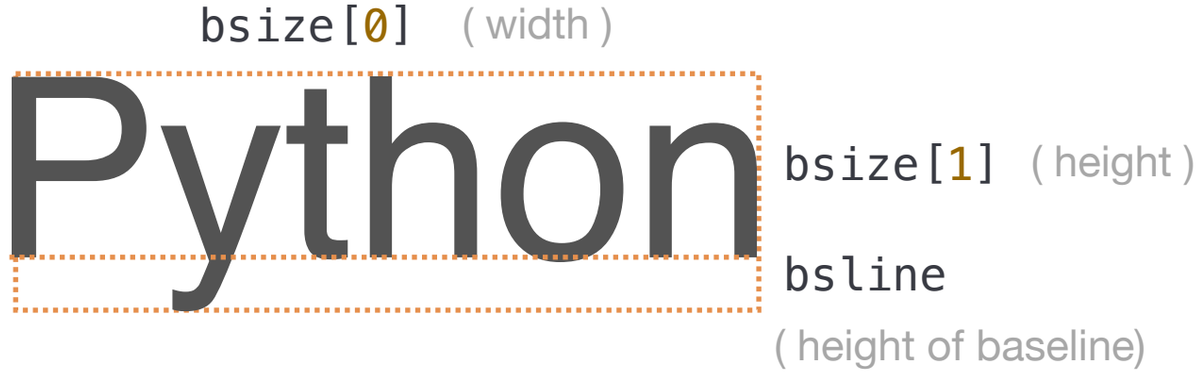
Get the width of the bounding box

Get the height of the bounding box

Analyze video

get the size of the text






```
> while True:
      ...
      for i in range(numOfObjects):
          ...
          if confidence > scoreThres:
              ...
              txtlbl      = "{} : {:.2f}".format(classNames[classId],
                                                 confidence)
              txtsize     = cv2.getTextSize(txtlbl,
                                            cv2.FONT_HERSHEY_SIMPLEX,
                                            0.5,
                                            1)
              bsize       = txtsize[0]
              bsline      = txtsize[1]
              ...
```

The text to be displayed

Get the size of the text

font scale

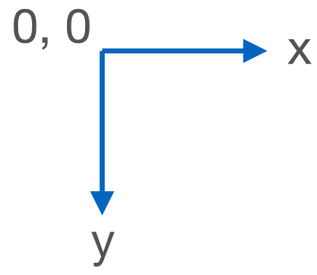
font thickness

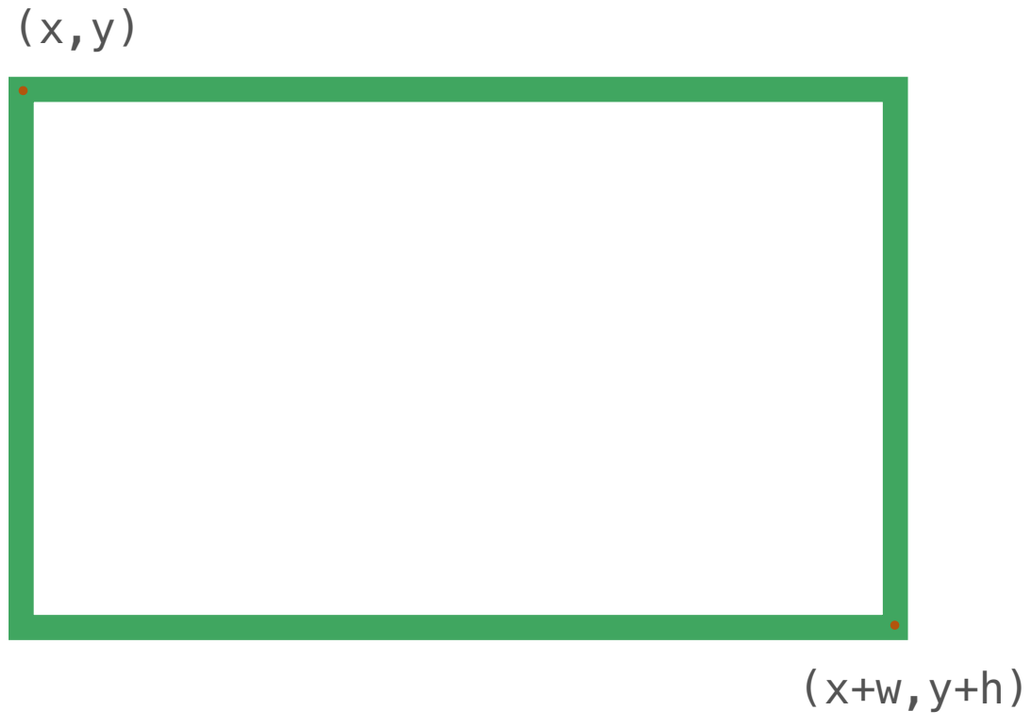
extract the width and height

extract the height of baseline

Analyze video

draw the bounding box

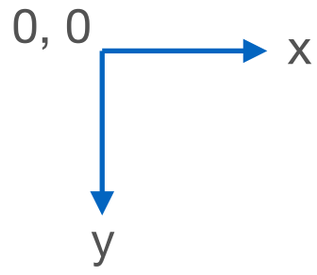


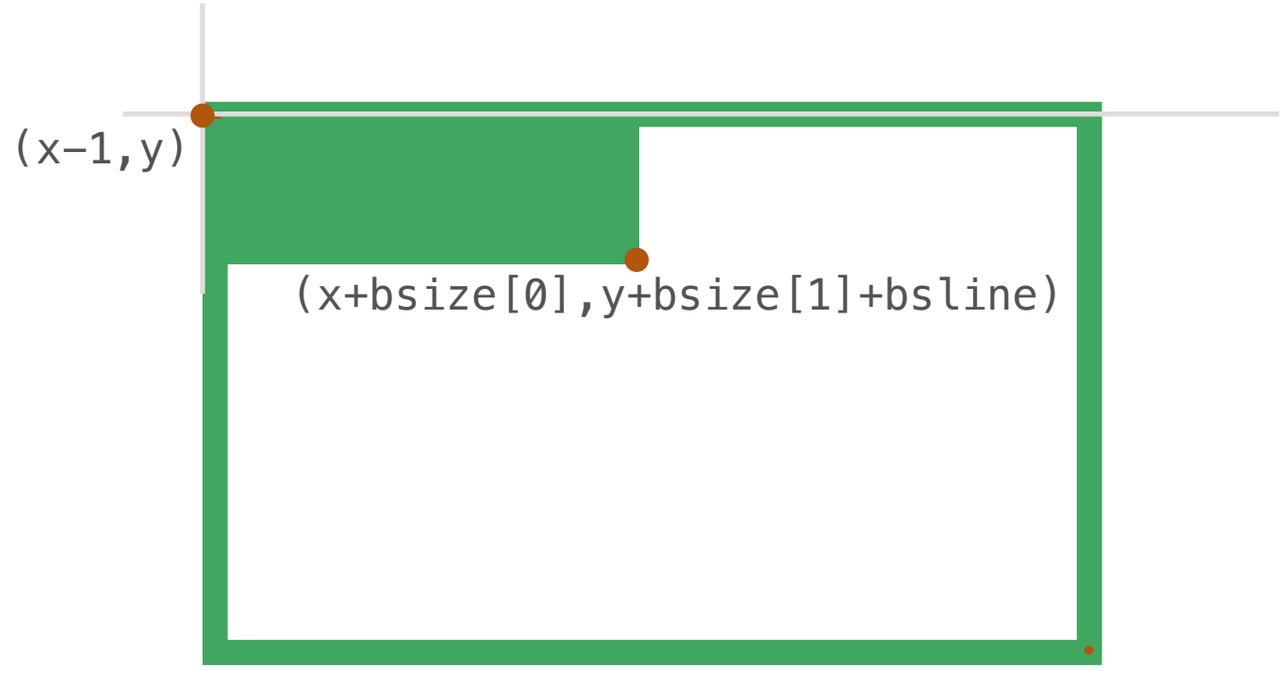



```
> while True:
      ...
      for i in range(numOfObjects):
          ...
      if confidence > scoreThres:
          ...
          cv2.rectangle(output,
                        (x,y),
                        (x+w,y+h),
                        (0, 255, 0),    colour
                        2)              thickness
          ...
```

Analyze video

in a while loop



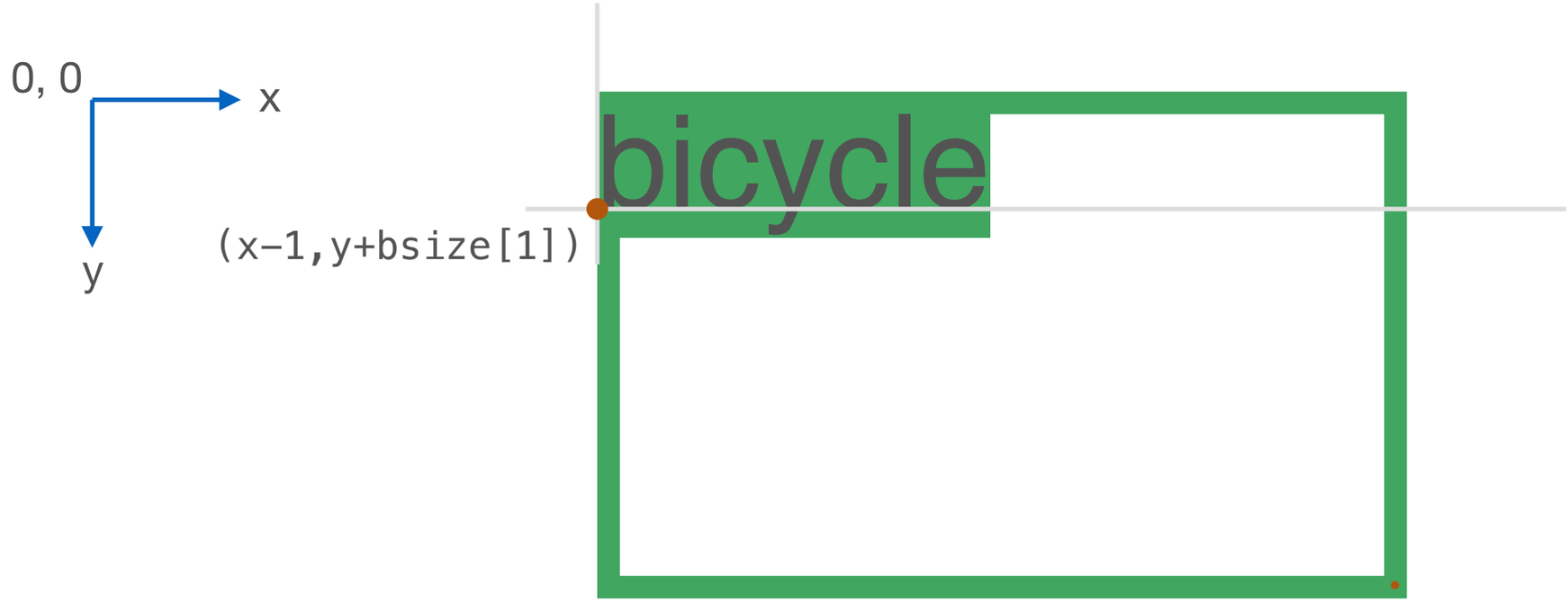



```
> while True:
      ...
      for i in range(numOfObjects):
          ...
          if confidence > scoreThres:
              ...
              cv2.rectangle(output,
                            (x-1,y),
                            (x+bsize[0],y+bsize[1]+bsline),
                            (0, 255, 0),
                            -1)
              ...
```

fill the rectangle with colour

Analyze video

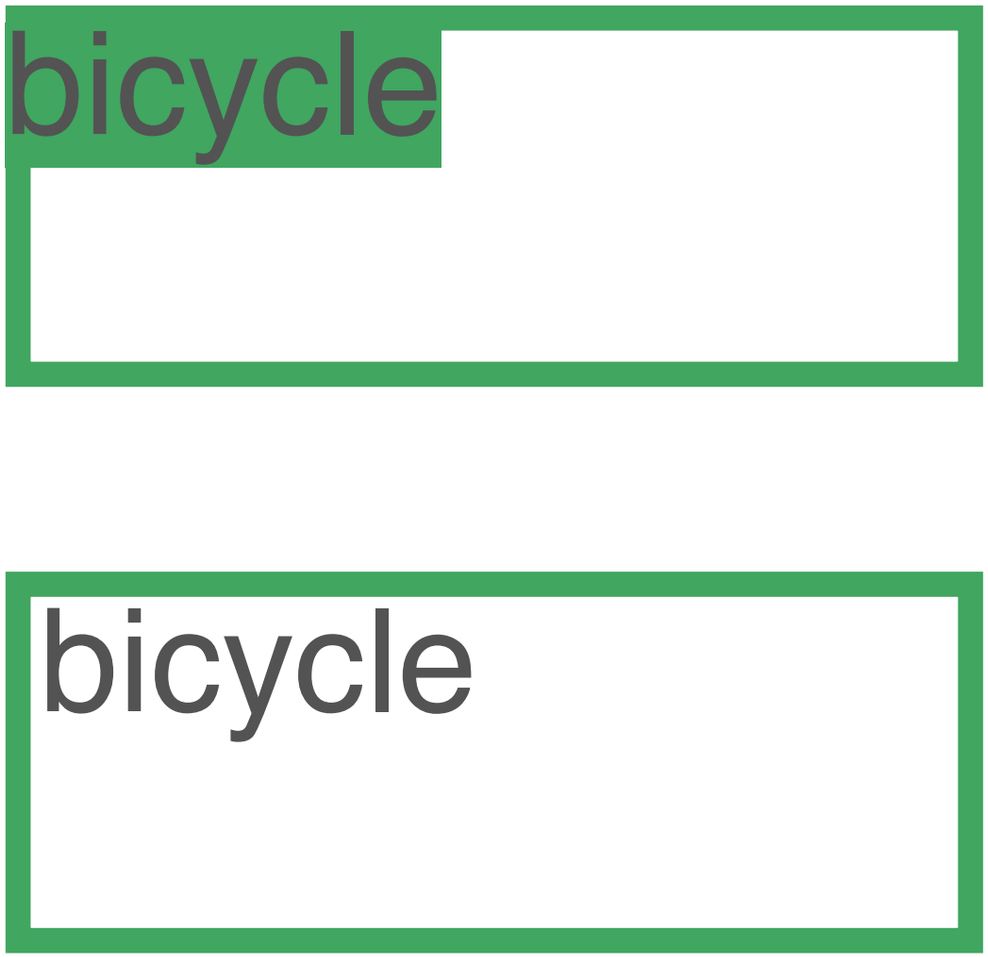
in a while loop




```
> while True:
      ...
      for i in range(numOfObjects):
          ...
      if confidence > scoreThres:
          ...
          cv2.putText(output,
                      txtlbl,
(x, y) position at bottom-left    (x-1,y+bsize[1]),
                font type         cv2.FONT_HERSHEY_SIMPLEX,
                font scale        0.5,
                colour            (0, 0, 0),
                font thickness    1,
                line type         cv2.LINE_AA)
```

A few possible ways to display bounding box

Do take note of their cons ..



Analyze video

in a while loop

```
> while True:
      ...

      if writer is None:                                  Setup the writer if not done
         fourcc = cv2.VideoWriter_fourcc(*"X264")         Use H.264 codec to save video
         writer = cv2.VideoWriter(outpath,
                                  fourcc,
                                  fps,
                                  (W, H),
                                  True)

      writer.write(output)
      cv2.imshow("SSD detection",output)                  Display the object detection in real-time

      if cv2.waitKey(1) >= 0:                             Terminate the analyzing process if ESC
          break                                           key is pressed

> writer.release()
> vs.release()
```

Common question:
Among all the available object detection methods (Faster RCNN, YOLO, SSD and etc), which one should I choose for my work?

The right solution?

for object detection

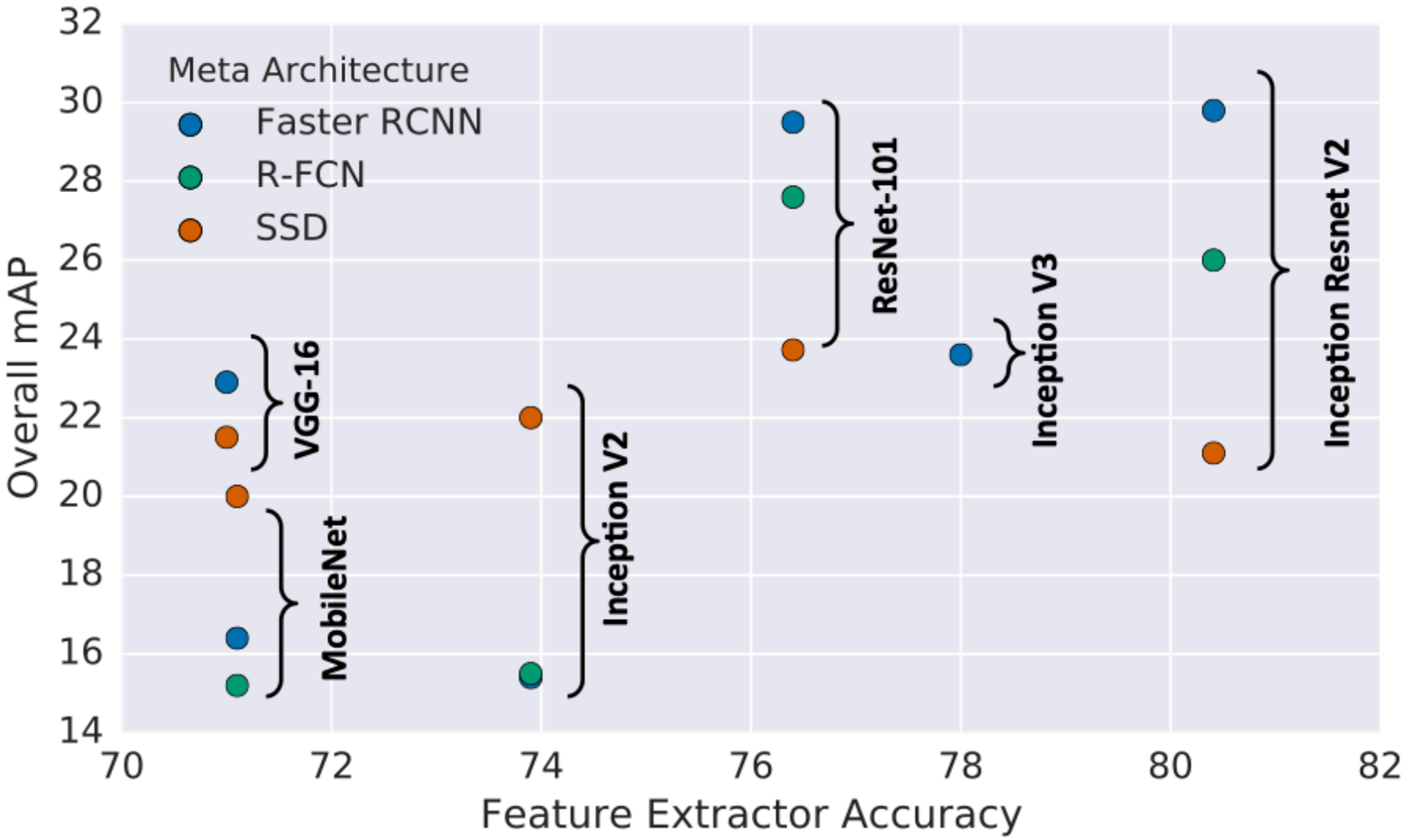
- The answer: it depends.
- Three factors to consider: hardware capability, the required accuracy, the required inference speed
- If accuracy is the most important factor (especially on small objects), use Faster RCNN
- If inference speed is the critical factor, pick either YOLO or SSD, or one-stage detector in general

Source: https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html

Comparison

by the overall accuracy

- mAP: mean average precision, the higher the better
- R-FCN: Region-based fully convolutional network

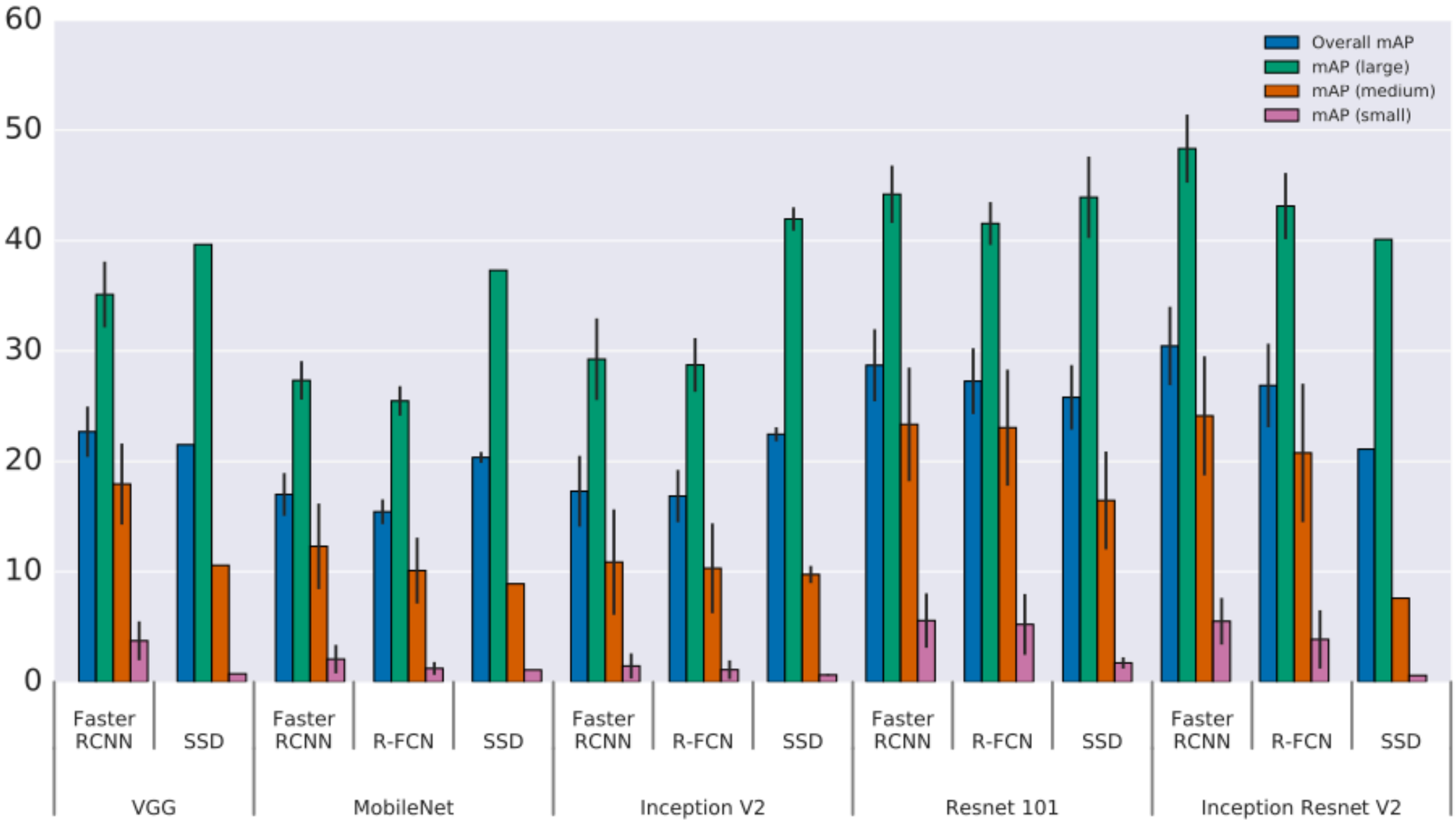


Source: https://arxiv.org/pdf/1611.10012.pdf

Comparison

by the accuracy on different object size

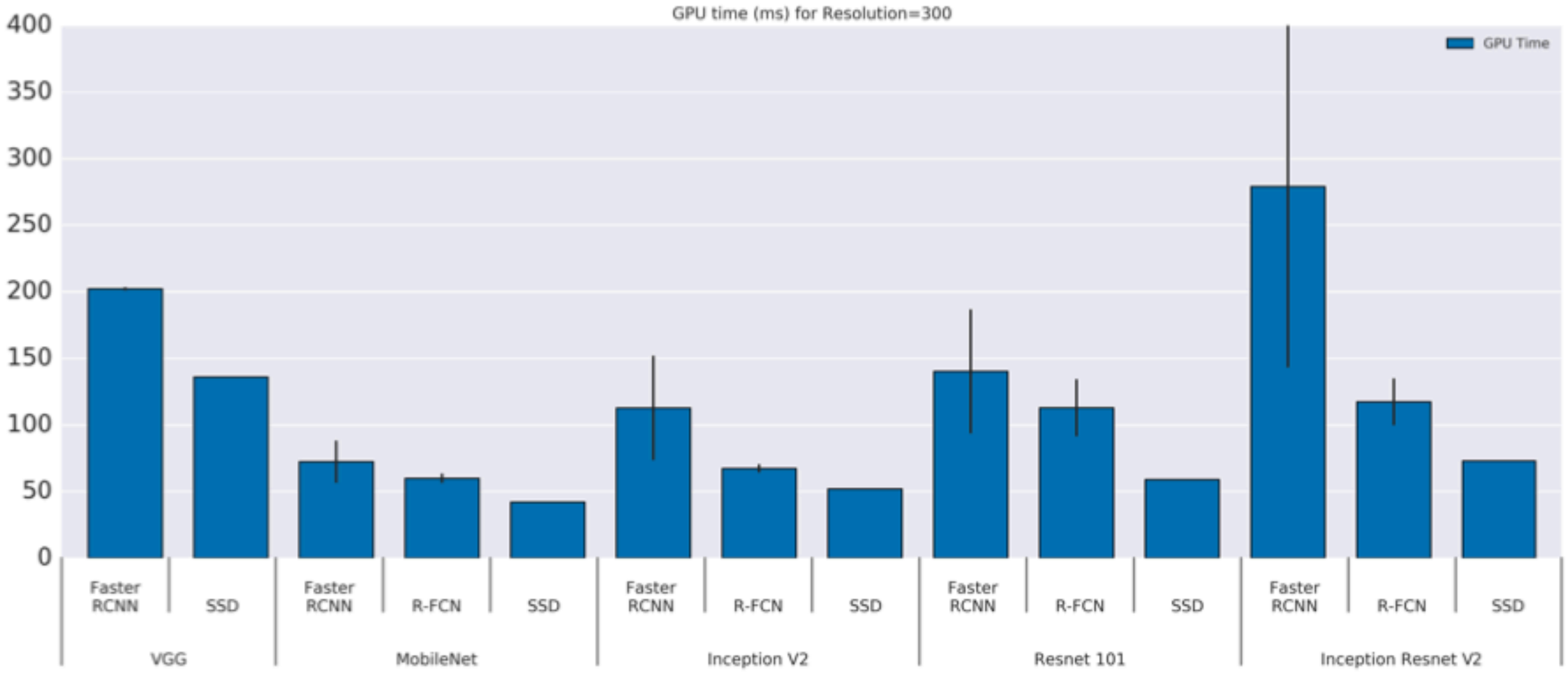
- pink bars for small objects in image



Source: https://arxiv.org/pdf/1611.10012.pdf

Comparison
by inference time

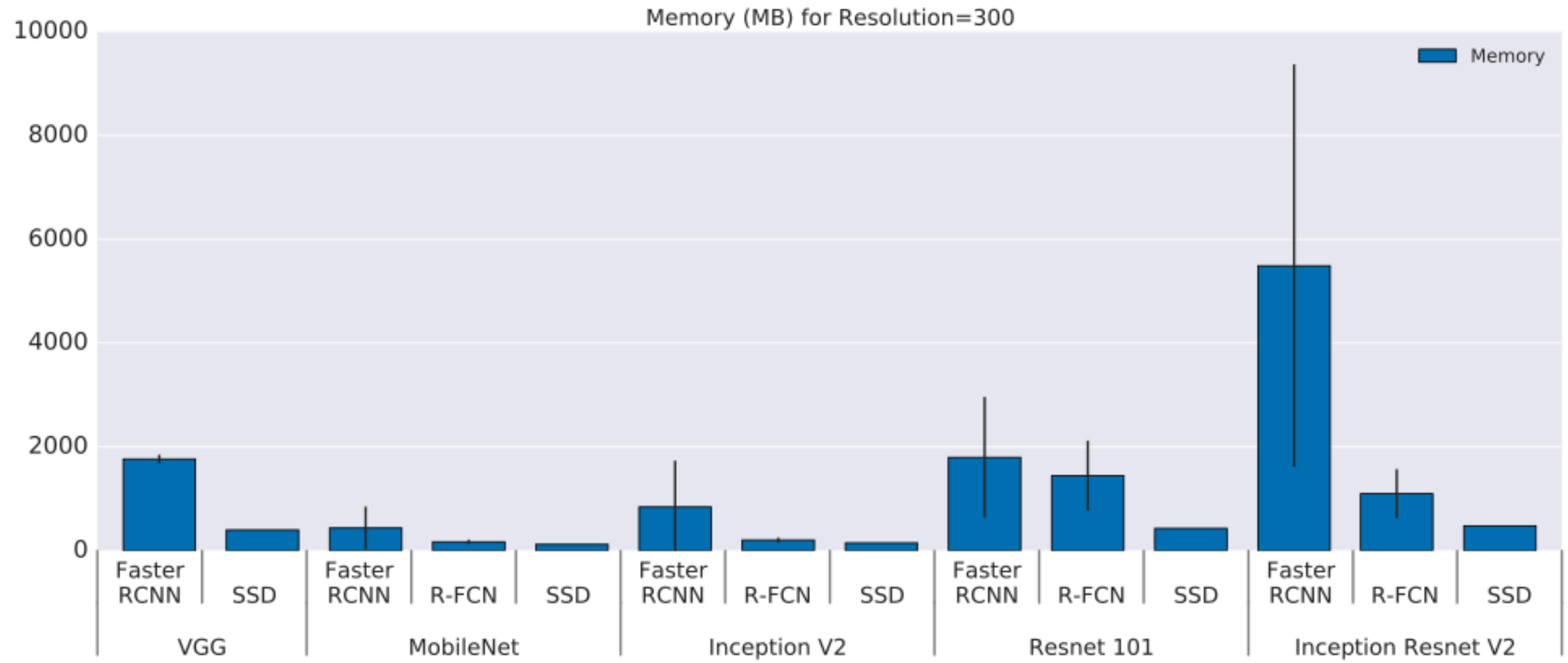
- The inference time on GPU in milisecond for an image with a size of 300 x 300



Source: https://arxiv.org/pdf/1611.10012.pdf

Comparison
by total memory usage

- SSD in generaly uses less 1GB of memory during inference



Source: https://arxiv.org/pdf/1611.10012.pdf