

exercise10

November 18, 2022

```
[1]: # This cell is used for creating a button that hides/unhides code cells to
      ↵quickly look only the results.
# Works only with Jupyter Notebooks.

from IPython.display import HTML

HTML('''<script>
code_show=true;
function code_toggle() {
if (code_show){
$('div.input').hide();
} else {
$('div.input').show();
}
code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
<form action="javascript:code_toggle()"><input type="submit" value="Click here
      ↵to toggle on/off the raw code."></form>'''')
```

```
[1]: <IPython.core.display.HTML object>
```

```
[2]: # Description:
#   Exercise10 notebook.
#
# Copyright (C) 2019 Antti Parviaainen
# Based on the SSD PyTorch implementation by Max deGroot and Ellis Brown
#
# This software is distributed under the GNU General Public
# Licence (version 2 or later);

import os
import numpy as np
from matplotlib import pyplot as plt
import cv2
```

```

import torch
from torch.autograd import Variable
from ssd import build_ssd
from data import VOCDetection, VOCAnnotationTransform
from data import VOC_CLASSES as labels

import warnings
warnings.filterwarnings("ignore")

# Select data directory
if os.path.isdir('/coursedata'):
    course_data_dir = '/coursedata'
    docker = False
elif os.path.isdir('../..../coursedata'):
    docker = True
    course_data_dir = '../..../coursedata'
else:
    # Specify course_data_dir on your machine
    docker = True
    course_data_dir = '/home/jovyan/work/coursedata/'

print('The data directory is %s' % course_data_dir)
data_dir = os.path.join(course_data_dir, 'exercise-10-data')
print('Data stored in %s' % data_dir)

```

The data directory is /coursedata
Data stored in /coursedata/exercise-10-data

1 CS-E4850 Computer Vision Exercise Round 10

The problems should be solved before the exercise session and solutions returned via MyCourses. Upload to MyCourses both: this Jupyter Notebook (.ipynb) file containing your solutions and the exported pdf version of this Notebook file. If there are both programming and pen & paper tasks kindly combine the two pdf files (your scanned/LaTeX solutions and the exported Notebook) into a single pdf and submit that with the Notebook (.ipynb) file. Note that you should be sure that everything that you need to implement works with the pictures specified in this exercise round.

Docker users: 1. One file, data file containing the weights, was again above the file size limit imposed by git. I've compressed it to get it under the limit so before you run the code you have to uncompress the file in the folder `coursedata/exercise-10-data/weights`.

1.1 Exercise 1 - Object detection with SSD: Single Shot MultiBox Object Detector, in PyTorch

The goal of this task is to learn the basics of deep learning based object detection with SSD by experimenting with the provided code and by reading the original [Single Shot MultiBox Detector](#) publication by Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang, and Alexander C. Berg from 2016.

Read the research paper linked above and experiment with the provided sample code according to the instructions below. Then answer the questions a), b), c) below and return your answers. Note that scientific publications are written for domain experts and some details may be challenging to understand if necessary background information is missing. However, don't worry if you don't understand all details. You should be able to grasp the overall idea and answer the questions even if some details would be difficult to understand.

The code implements the following steps to demonstrate SSD:

1. Build the SSD300 architecture and load pretrained weights on the VOC07 trainval dataset
2. Load 4 random sample images from the VOC07 dataset
3. Preprocess the images to the correct input form and show the preprocessed images
4. Run the sample images through the SSD network, parse the detections and show the results

a) In the beginning of the publication the authors argue about the relevance and novelty of their work. Summarise their main arguments and the evidence they present to support their arguments. Type your answer here:

The authors present a method for detecting objects in images using a single deep neural network called SSD. Generally, the model split the bounding boxes into a set of default boxes on different scales for each feature map location.

- The method generates scores for object categories presence in each default box and produces adjustments to better match the object shape. It combines predictions from multiple feature maps with different resolutions to detect different size objects.
- The model eliminates proposal generation and subsequent pixel or feature resampling stages, making it easy to train and straightforward to integrate into detection-involved systems. Specifically, SSD achieves a result of 59 FPS with mAP 74.3% on VOC2007 test, compared to Faster R-CNN 7 FPS with mAP 73.2% or YOLO 45 FPS with mAP 63.4%.
- The model also achieves high accuracy even with relatively low resolution input. Compared to methods utilizing an additional object proposal step, SSD is much faster and improves the speed vs accuracy trade-off.
- The model is put into test against the Faster R-CNN model by comparing performance on the PASCAL VOC, COCO, and ILSVRC datasets.

b) SSD consists of two networks: a “truncated base network” and a network of added “convolutional feature layers to the end of the truncated base network.” What is the purpose of the base network? Which base network do the authors of the SSD publication use, and what dataset was used to train the base network? Type your answer here:

- The base network is based on a standard architecture for classifying high quality image. The author use the VGG-16 network as their base network, and they train the base network with the ILSVRC CLS-LOC dataset.

c) SSD has it's own loss function, defined in chapter 2.2 in the original publication. What are the two attributes this loss function observes? How are these defined (short

explanation without any formulas is sufficient) and how do they help the network to minimise the object detection error? Type your answer here:

- The loss function is defined as a weighted sum with the two attributes being the localization loss (loc) and the confidence loss (conf). The localization loss is a Smooth L1 loss between the predicted box (I) and the ground truth box (g); this loss would help in predicting the loss in relation to default box shapes. The confidence loss is the softmax loss over multiple class confidences (c); this loss would help in predicting the presence of a class instance in each box.

1.1.1 1. Load the SSD architecture and the pretrained weights

```
[3]: net = build_ssd('test', 300, 21)      # initialize SSD
net.load_weights(data_dir+'/weights/ssd300_mAP_77.43_v2.pth')
```

Loading weights into state dict...
Finished!

1.1.2 2. Load Sample Images

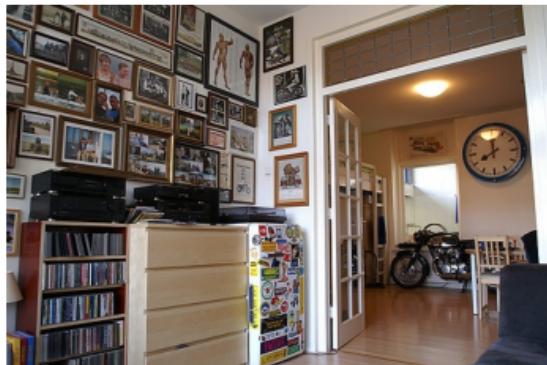
```
[4]: images = []
for i in range(4):
    if docker:
        # if local storage use a 300 image subset of the test set
        img_id = np.random.randint(1,high = 300)
        image = cv2.imread(data_dir+'/voc_images/'+f"{{img_id:06d}}.jpg", cv2.
    ↪IMREAD_COLOR)
    else:
        # if JupyterLab use the full test set
        # here we specify year (07 or 12) and dataset ('test', 'val', 'train')
        testset = VOCDetection(data_dir+'/VOCdevkit', [('2007', 'val')], None, ↪
    ↪VOCAnnotationTransform())
        img_id = np.random.randint(0,high = len(testset))
        image = testset.pull_image(img_id)

    rgb_image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    images.append(rgb_image)

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10,10))
ax = axes.ravel()
ax[0].imshow(images[0])
ax[0].axis('off')
ax[1].imshow(images[1])
ax[1].axis('off')
ax[2].imshow(images[2])
ax[2].axis('off')
ax[3].imshow(images[3])
ax[3].axis('off')
plt.tight_layout()
```

```
plt.suptitle("Randomly sampled images from the dataset", fontsize=20)
plt.subplots_adjust(top=0.95)
plt.show()
```

Randomly sampled images from the dataset



1.1.3 3 Pre-process the input images

Using the torchvision package, we can apply multiple built-in transforms. At test time we resize our image to 300x300, subtract the dataset's mean rgb values, and swap the color channels for input to SSD300.

```
[5]: def preprocess_inputs(images):
    preprocessed_images = []
    for image in images:
```

```
x = cv2.resize(image, (300, 300)).astype(np.float32)
x -= (104.0, 117.0, 123.0)
x = x.astype(np.float32)
preprocessed_images.append(x)
return preprocessed_images
```

```
[6]: preprocessed_images = preprocess_inputs(images)

fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(10,10))
ax = axes.ravel()
ax[0].imshow(preprocessed_images[0])
ax[0].axis('off')
ax[1].imshow(preprocessed_images[1])
ax[1].axis('off')
ax[2].imshow(preprocessed_images[2])
ax[2].axis('off')
ax[3].imshow(preprocessed_images[3])
ax[3].axis('off')
plt.tight_layout()
plt.suptitle("Preprocessed input images", fontsize=20)
plt.subplots_adjust(top=0.95)
plt.show()
```

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).

Preprocessed input images



4. Run SSD on the sample images and show the results

```
[7]: def run_network(images, nrows, ncols, figsize = (10,10), threshold = 0.6, title=True):

    if nrows * ncols != len(images):
        print("Subgrid dimensions don't match with the number of images.")
        return

    preprocessed_images = preprocess_inputs(images)
    fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=figsize)
```

```

if len(images) != 1:
    ax = axes.ravel()
else:
    ax = [axes]

for it, input_image in enumerate(images):
    # Process data for the network
    # swap color channels
    x = preprocessed_images[it][:, :, ::-1].copy()
    # change the order
    x = torch.from_numpy(x).permute(2, 0, 1)
    # wrap the image in a Variable so it is recognized by PyTorch autograd
    xx = Variable(x.unsqueeze(0))
    if torch.cuda.is_available():
        xx = xx.cuda()

    # SSD Forward Pass
    y = net(xx)
    detections = y.data

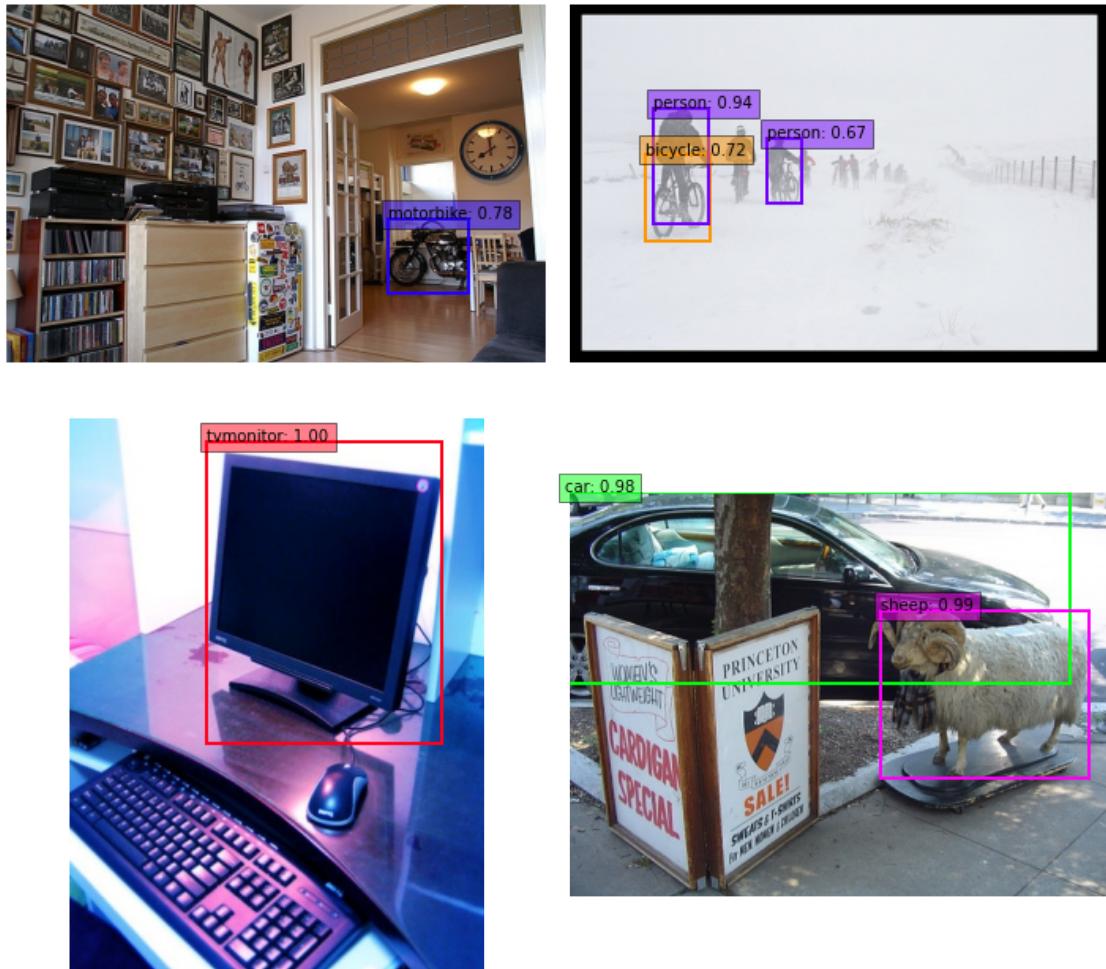
    # colormap for the bounding boxes
    colors = plt.cm.hsv(np.linspace(0, 1, 21)).tolist()

    # scale each detection back up to the image
    scale = torch.Tensor(input_image.shape[1:-1]).repeat(2)
    for i in range(detections.size(1)):
        j = 0
        while detections[0,i,j,0] >= threshold:
            score = detections[0,i,j,0]
            label_name = labels[i-1]
            display_txt = '%s: %.2f'%(label_name, score)
            pt = (detections[0,i,j,1:]*scale).cpu().numpy()
            coords = (pt[0], pt[1]), pt[2]-pt[0]+1, pt[3]-pt[1]+1
            color = colors[i]
            ax[it].add_patch(plt.Rectangle(*coords, fill=False, □
→edgecolor=color, linewidth=2))
            ax[it].text(pt[0], pt[1], display_txt, bbox={'facecolor':color, □
→'alpha':0.5})
            j+=1
        ax[it].imshow(images[it])
        ax[it].axis('off')
    plt.tight_layout()
    if title:
        plt.suptitle("Detection results at threshold: %1.2f" %threshold, □
→fontsize=20)
        plt.subplots_adjust(top=0.95)
    plt.show()

```

```
[8]: # Filter outputs with confidence scores lower than a threshold. The default
      ↪threshold is 60%.
run_network(images, nrows=2, ncols=2, figsize=(10,10), threshold=0.6)
```

Detection results at threshold: 0.60



1.2 Exercise 2 - Evaluating the SSD network on some more challenging input images

To better detect challenging inputs the authors have implemented data augmentation described in chapter 2.2 and 3.2 of the publication and in [reference 14](#). Read the chapters in the publication and selectively read the main points of reference 14 and answer the following questions.

1.2.1 2.1

a) Based on the results (test images1) below what kind of objects are easier for the network to detect? Type your answer here:

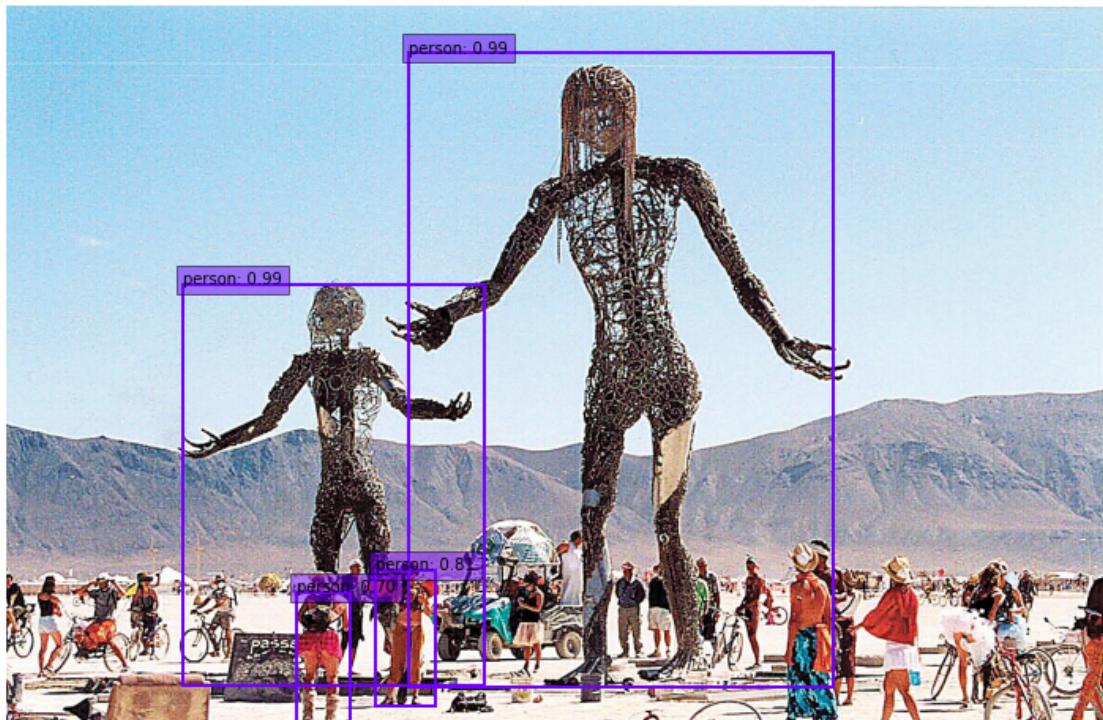
It seems that bigger objects with clear separation are easier for the network to detect.

b) Would switching from input size 300x300 to 512x512 make the problem better, worse, or have no impact? Elaborate on why or why not. Would designing a object detector that uses HD resolution of 1920x1080, or even higher, images as inputs be a good idea? Elaborate on why or why not. Type your answer here:

SSD model uses multiple layer for prediction and can achieve considerable accuracy with relatively low resolution input. If the input size is larger (for example 512x512), the problem may return higher classification accuracy. However, if the resolution is too high (such as 1920x1080), it will require a larger network that is computationally expensive and therefore not recommended.

```
[16]: # Photo credit: to burningman.org. Used under the fair use principles for  
→transformative educational purposes.  
image11 = cv2.imread(data_dir+'/test_images/img11.jpg', cv2.IMREAD_COLOR)  
images1=[cv2.cvtColor(image11, cv2.COLOR_BGR2RGB)]  
run_network(images1, nrows=1, ncols=1, figsize=(10,10), threshold=0.6)
```

Detection results at threshold: 0.60



1.2.2 2.2

a) Based on the results (test images2) below elaborate why the detector has trouble detecting the objects? Type your answer here:

It seems that the detector has trouble detecting objects that are tightly distributed and greatly overlap.

b) Have the authors of the SSD publication tried to mitigate this problem? If yes, briefly explain how. Type your answer here:

The authors performed data augmentation to make the model more robust to various input object sizes by randomly sampling each training image by:

- * The entire original input image
- * Sampling of a patch whose minimum Jaccard overlap with the objects is 01, 0.3, 0.5, 0.7, or 0.9
- * Random sampling of a patch After that, each sampled patch is resized appropriately and horizontally flipped with a 0.5 probability.

```
[10]: # Photo credit: free wallpaper image. Used under the fair use principles for  
    ↪transformative educational purposes.  
image21 = cv2.imread(data_dir+'/test_images/img21.jpg', cv2.IMREAD_COLOR)  
# Photo credit: David Dodman, KNOM. Used under the fair use principles for  
    ↪transformative educational purposes.  
image22 = cv2.imread(data_dir+'/test_images/img22.jpg', cv2.IMREAD_COLOR)  
images2=[cv2.cvtColor(image21, cv2.COLOR_BGR2RGB),cv2.cvtColor(image22, cv2.  
    ↪COLOR_BGR2RGB)]  
run_network(images2, nrows=2, ncols=1, figsize=(25,25), threshold=0.6)
```

Detection results at threshold: 0.60



1.2.3 2.3

- a) Based on the results (test images3) below, elaborate why the detector has trouble detecting the objects? Type your answer here:

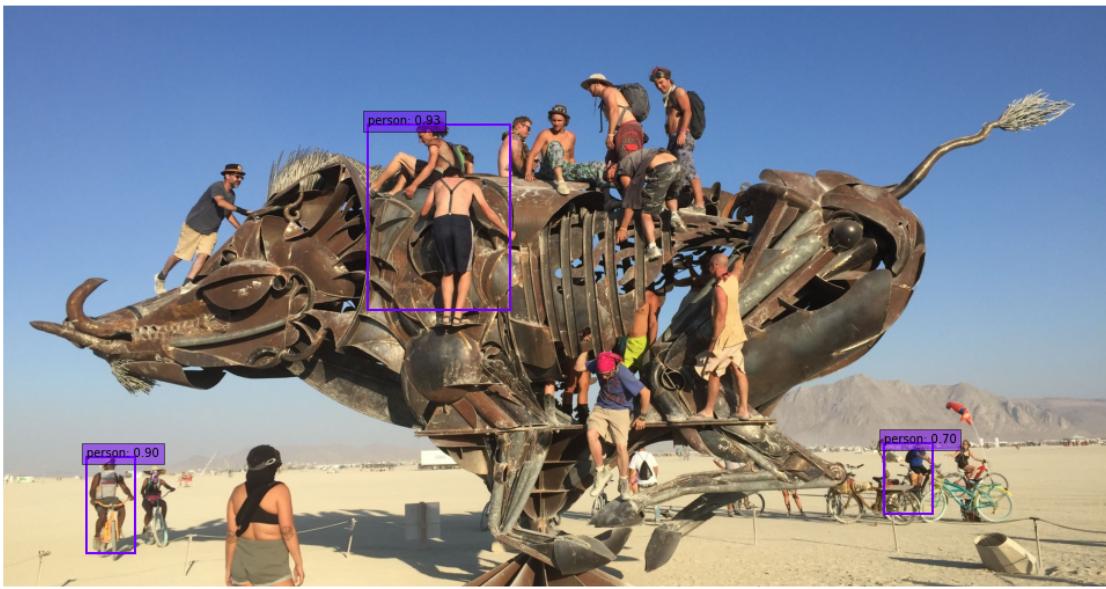
It seems that the detector has trouble detecting small objects and occlusions of objects. The model might skip the smaller object since it is already included in the bigger bounding box that correspond to the bigger object laying underneath.

b) Have the authors of the SSD publication tried to mitigate this problem? If yes, briefly explain how. Type your answer here:

The authors perform the data augmentation process as described in Part 2.2).

```
[11]: # Photo credit: Duncan Rawlinson - Duncan.co photo from flickr. Creative Commons license. https://creativecommons.org/licenses/by-nc/2.0/
image31 = cv2.imread(data_dir+'/test_images/img31.jpg', cv2.IMREAD_COLOR)
# Photo credit: Karri Huhtanen from flickr. Creative Commons license. https://creativecommons.org/licenses/by-nc/2.0/
image32 = cv2.imread(data_dir+'/test_images/img32.jpg', cv2.IMREAD_COLOR)
images3=[cv2.cvtColor(image31, cv2.COLOR_BGR2RGB),cv2.cvtColor(image32, cv2.COLOR_BGR2RGB)]
run_network(images3, nrows=2, ncols=1, figsize=(15,15), threshold=0.6)
```

Detection results at threshold: 0.60



1.2.4 2.4

- a) Based on the results (test images4) below elaborate why the detector has trouble detecting objects in the first/upper image, but is able to detect objects in the second/lower image which contains the left part of the upper image? Note: you can double click the upper image to show it in actual size. Type your answer here:

It seems that the model has trouble detecting smaller size objects, combining with the fact that the first image has high density of objects that confuses the model into inaccurate different-sized objects.

b) Have the authors of the SSD publication tried to mitigate this problem? If yes, briefly explain how. Type your answer here:

The author present a method for choosing scales and aspect ratios for default boxes. They design the tiling of default boxes so that specific feature maps learn to be responsive to particular scales of the objects, which may explain the better detection in the lower scaled version of the image. The formula for the scale of the default boxes for each of m feature maps is:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1)$$

with

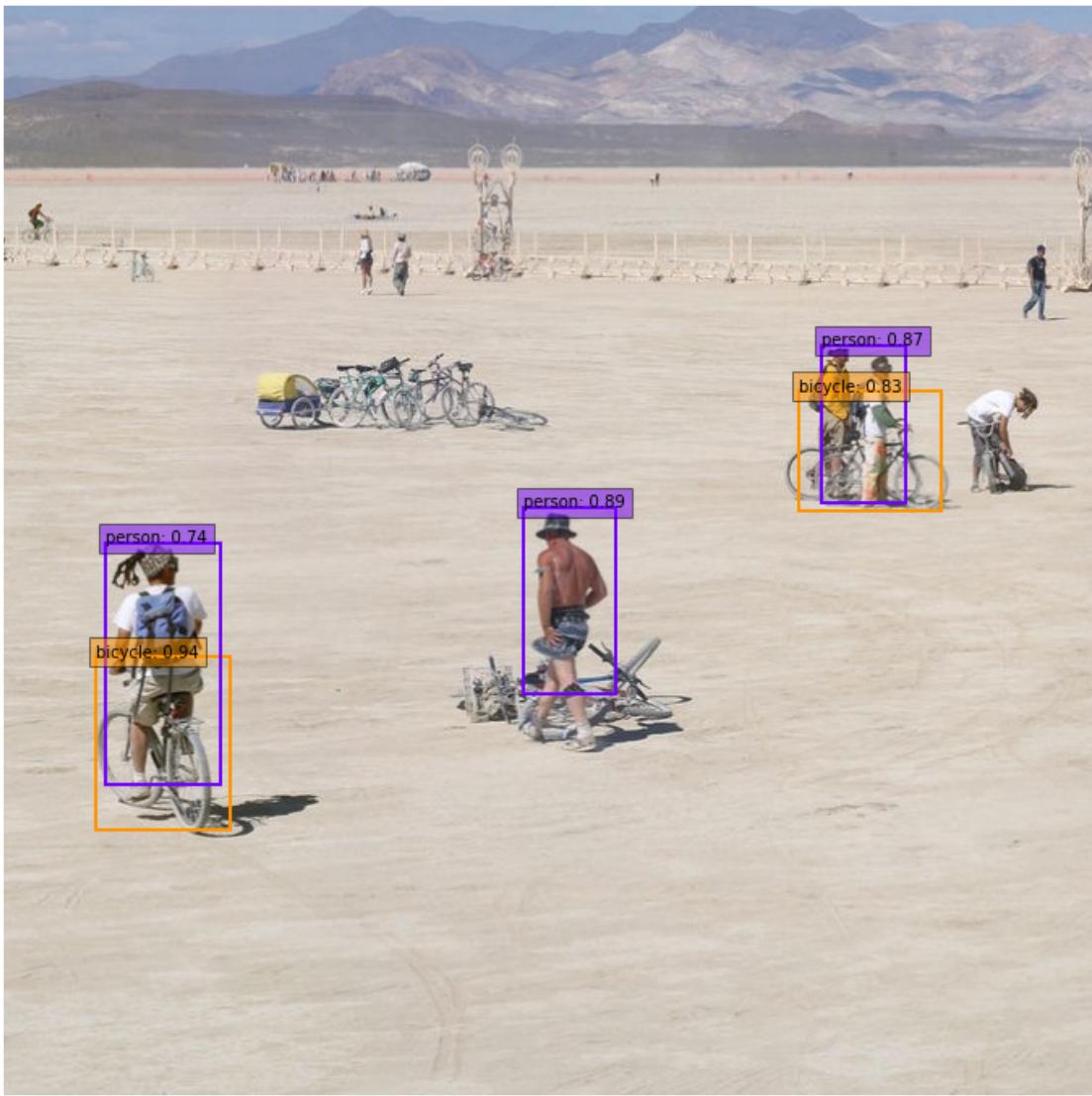
$$k \in [1, m]$$

```
[12]: # Photo credit: Brad Templeton. Used under the fair use principles for  
→transformative educational purposes.  
image41 = cv2.imread(data_dir+'/test_images/img41.jpg', cv2.IMREAD_COLOR)  
image42 = cv2.imread(data_dir+'/test_images/img42.jpg', cv2.IMREAD_COLOR)  
images4 = [cv2.cvtColor(image41, cv2.COLOR_BGR2RGB), cv2.cvtColor(image42, cv2.  
→COLOR_BGR2RGB)]  
# run the first image without title information to show it properly  
run_network([images4[0]], nrows=1, ncols=1, figsize=(100,100), threshold=0.6,  
→title=False)
```



```
[23]: run_network([images4[1]], nrows=1, ncols=1, figsize=(10,10), threshold=0.6)
```

Detection results at threshold: 0.60



1.2.5 2.5

- a) One of the test images in (images5) was perhaps accidentally flipped vertically and as can be seen the detector has trouble detecting objects if there's a significant rotation. Have the authors of the SSD publication tried to mitigate this problem? If yes, briefly explain how. If no, propose a naive method to mitigate it and explain why it usually might not be necessary or a good idea(more harm than good). Type your answer here:

The authors only proposed random horizontal flip. A naive method to mitigate this problem may be adding the random vertical flip in the data augmentation process as well as rotation and sampling.

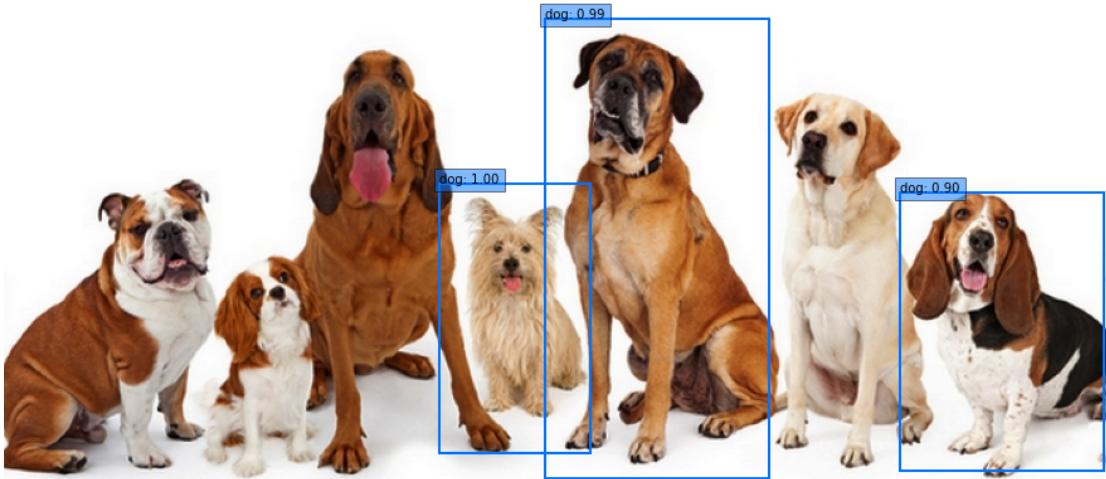
b) Is a convolutional network naturally, without specifically addressing the issue, able to detect objects if there are small rotations? If yes, briefly explain which part of the network helps to mitigate the effects of rotation and why. (Hint: what layers are sometimes between two convolutional layers) Type your answer here:

Naturally, a convolutional network with max-pooling layers may help mitigate the effect of small changes in shift and rotation, but only when the changes are small enough for the max-pooling layers to get the maximum value of feature maps.

[19]: # Photo credit: Susan Schmitz / shutterstock. Used under the fair use
→ principles for transformative educational purposes.

```
image51 = cv2.imread(data_dir+'/test_images/img51.jpg', cv2.IMREAD_COLOR)
image52 = cv2.imread(data_dir+'/test_images/img52.jpg', cv2.IMREAD_COLOR)
images5=[cv2.cvtColor(image51, cv2.COLOR_BGR2RGB),cv2.cvtColor(image52, cv2.
→COLOR_BGR2RGB)]
run_network(images5, nrows=2, ncols=1, figsize=(15,15), threshold=0.9)
```

Detection results at threshold: 0.90



1.2.6 2.6

- a) Incrementally lower the detection confidence threshold, run SSD on the test images and observe the results. What are the upsides and downsides of lowering the confidence threshold? How could you measure the effect of the confidence threshold? Assume you have some object detection task that you want to apply the detector to. What kind of object detection tasks could benefit from a lower confidence threshold and

what kind of tasks need a high confidence threshold? Type your answer here: Lowering the confidence threshold: * Upsides: Detecting more objects * Downsides: Lower classification confidence and accuracy * Effect measurement: Looking at classification accuracy * Usage: lower confidence threshold is good for tasks that attempt to detect as many objects as possible, while higher confidence threshold is good for tasks that emphasize accuracy and precision.

b) Watch the following YouTube video of a detector that is similar to SSD called YOLO V2 (You Only Look Once) and use the knowledge from previous tasks to answer the following questions: In what object detection tasks is a state of the art object detector especially useful and able to perform better than a human? In what object detection tasks is a human better than a state of the art object detector? Give examples of both. Type your answer here:

- State-of-the-art detectors are better at identifying multiple objects at the same time including when there are many objects in the same image and the scale is small (objects appear far away in the image). For example, in Part 2.4, the model can recognize many objects like bicycles and people at the same time while human cannot.
- Humans are better at recognizing the objects in spite of occlusions or transformation of the objects like rotation. For example, in Part 2.2, human can recognize the overlapping objects like the cat and the hat while the model cannot.