*# set up Python environment: numpy for numerical routines, and matplotlib*

**import numpy as np**

**import matplotlib.pyplot as plt**

*# display plots in this notebook*

%**matplotlib** inline **import matplotlib.patches as patches**

*# set display defaults*

plt.rcParams['figure.figsize'] = (10, 10) *# large images*

plt.rcParams['image.interpolation'] = 'nearest' *# don't interpolate: show*

plt.rcParams['image.cmap'] = 'gray' *# use grayscale output rather than a*

*# The caffe module needs to be on the Python path; # we'll add it here explicitly.*

**import sys**

caffe\_root = '../../' *# this file should be run from { caffe\_root}/example*

sys.path.insert(0, caffe\_root + 'python')

**import caffe**

**import math**

*# If you get "No module named \_caffe", either you have not built pycaffe*

*o*

**def** sigmoid( p ):

**return** 1.0 / (1 + math.exp(-p \* 1.0))

**def** overlap(x1, w1, x2, w2): *#x1 ,x2 are two box center x*

left = max(x1 - w1 / 2.0, x2 - w2 / 2.0)

right = min(x1 + w1 / 2.0, x2 + w2 / 2.0)

**return** right - left

**def** cal\_iou(box, truth):

w = overlap(box[0], box[2], truth[0], truth[2])

h = overlap(box[1], box[3], truth[1], truth[3])

**if** w < 0 **or** h < 0:

**return** 0

inter\_area = w \* h

union\_area = box[2] \* box[3] + truth[2] \* truth[3] - inter\_area

**return** inter\_area \* 1.0 / union\_area

**def** apply\_nms(boxes, thres):

sorted\_boxes = sorted(boxes,key=**lambda** d: d[7])[::-1]

p = dict()

**for** i **in** range(len( sorted\_boxes )):

**if** i **in** p:

**continue**

truth = sorted\_boxes[i]

**for** j **in** range(i+1, len( sorted\_boxes )):

**if** j **in** p:

**continue**

box = sorted\_boxes[j]

iou = cal\_iou(box, truth)

**if** iou >= thres:

p[j] = 1

res = list()

**for** i **in** range(len( sorted\_boxes )):

**if** i **not in** p:

res.append(sorted\_boxes[i])

**return** res

**def** show\_boxes( boxes ):

label\_name = {0: "bg", 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" }

w = image.shape[1]

h = image.shape[0]

plt.imshow(image)

currentAxis = plt.gca()

colors = plt.cm.hsv(np.linspace(0, 1, 21)).tolist()

**for** box **in** boxes:

x\_min = int(round((box[0]-box[2]/2.0) \* w))

x\_max = int(round((box[0]+box[2]/2.0) \* w))

y\_min = int(round((box[1]-box[3]/2.0) \* h))

y\_max = int(round((box[1]+box[3]/2.0) \* h))

**if** x\_min < 0:

x\_min = 0

**if** x\_max > w:

x\_max = w

**if** y\_min < 0:

y\_min = 0

**if** y\_max > h:

y\_max = h

display\_txt = '{:0.2f}, {}'.format(box[7], label\_name[box[4]])

coords = (x\_min, y\_min), x\_max-x\_min+1, y\_max-y\_min+1

color = colors[box[4]]

currentAxis.add\_patch(plt.Rectangle(\*coords, fill=False, edgecolor r=color, linewidth=2))

currentAxis.text(x\_min, y\_min, display\_txt, bbox={'facecolor': color, 'alpha':0.5})

plt.show()

caffe.set\_mode\_cpu()

model\_def = './good/good\_deploy.prototxt'

model\_weights = './good/change\_iter\_45000.caffemodel'

net = caffe.Net(model\_def, *# defines the structure of the model*

model\_weights, *# contains the trained weights*

caffe.TEST) *# use test mode (e.g., don't perform dropo*

*# load the mean ImageNet image (as distributed with Caffe) for subtraction*

*#mu = np.load(caffe\_root + 'python/caffe/imagenet/ilsvrc\_2012\_mean.npy')*

*#mu = mu.mean(1).mean(1) # average over pixels to obtain the mean (BGR)*

*p*

*#print 'mean-subtracted values:', zip('BGR', mu)*

mu = np.array([105, 117, 123])

*# create transformer for the input called 'data'*

transformer = caffe.io.Transformer({'data': net.blobs['data'].data.shape})

transformer.set\_transpose('data', (2,0,1)) *# move image channels to outer*

transformer.set\_mean('data', mu) *# subtract the dataset-mean va*

transformer.set\_raw\_scale('data', 255) *# rescale from [0, 1] to [0, 2* transformer.set\_channel\_swap('data', (2,1,0)) *# swap channels from RGB to*

net.blobs['data'].reshape(1, *# batch size*

3, *# 3-channel (BGR) images*

416, 416) *# image size is 227x227*

image = caffe.io.load\_image('darknet\_v3/dog.jpg')

transformed\_image = transformer.preprocess('data', image)

net.blobs['data'].data[...] = transformed\_image

*### perform classification*

output = net.forward()

res = output['conv\_reg'][0] *# the output probability vector for the first*

swap = np.zeros((13\*13,5,25))

*#change*

index = 0 **for** h **in** range(13):

**for** w **in** range(13):

**for** c **in** range(125):

swap[h\*13+w][c/25][c%**25**] = res[c][h][w]

biases = [1.08,1.19,3.42,4.41,6.63,11.38,9.42,5.11,16.62,10.52]

boxes = list() **for** h **in** range(13):

**for** w **in** range(13):

**for** n **in** range(5):

box = list() ;

cls = list() ;

s = 0;

x = (w + sigmoid(swap[h\*13+w][n][0])) / 13.0; *#center x*

y = (h + sigmoid(swap[h\*13+w][n][1])) / 13.0; *#center y*

ww = (math.exp(swap[h\*13+w][n][2])\*biases[2\*n]) / 13.0; *#w*

hh = (math.exp(swap[h\*13+w][n][3])\*biases[2\*n+1]) / 13.0; *#h*

obj\_score = sigmoid(swap[h\*13+w][n][4]) ;

**for** p **in** range(20):

cls.append(swap[h\*13+w][n][5+p]);

large = max( cls);

**for** i **in** range(len( cls )):

cls[i] = math.exp(cls[i] - large);

s = sum( cls);

**for** i **in** range(len( cls )):

cls[i] = cls[i] \* 1.0 / s;

box.append(x); *#0*

box.append(y); *#1*

box.append(ww); *#2*

box.append(hh); *#3*

box.append(cls.index(max(cls))+1) *#4*

box.append(obj\_score); *#5*

box.append(max(cls)); *#6*

box.append(obj\_score \* max( cls ))

**if** box[5] \* box[6] > 0.8:

boxes.append(box);

res = apply\_nms(boxes, 0.5)

show\_boxes(res)

