AI VIETNAM
AI Insight Course

Object Localization and Detection

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2021

Global Pooling

Max pooling: Features are preserved

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

Data

2x2 max pooling $\left(\begin{array}{|c|c|c|c|} \hline v_1 & v_2 & v_3 & v_4 \\ \hline v_5 & v_6 & v_7 & v_8 \\ \hline v_9 & v_{10} & v_{11} & v_{12} \\ \hline v_{13} & v_{14} & v_{15} & v_{16} \\ \hline \end{array} \right) = \begin{array}{|c|c|} \hline m_1 & m_2 \\ \hline m_3 & m_4 \\ \hline \end{array}$

$m_1 = \max(v_1, v_2, v_5, v_6)$

$m_2 = \max(v_3, v_4, v_7, v_8)$

$m_3 = \max(v_9, v_{10}, v_{13}, v_{14})$

$m_4 = \max(v_{11}, v_{12}, v_{15}, v_{16})$

keras.layers.MaxPooling2D(pool_size=2)

Feature map (220x220)

max pooling (2x2) →

Feature map (110x110)

max pooling (2x2) →

Feature map (55x55)

Global Pooling

Max pooling

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

Data

2x2 max pooling (

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

) =

m_1	m_2
m_3	m_4

$$m_1 = \max(v_1, v_2, v_5, v_6)$$
$$m_2 = \max(v_3, v_4, v_7, v_8)$$
$$m_3 = \max(v_9, v_{10}, v_{13}, v_{14})$$
$$m_4 = \max(v_{11}, v_{12}, v_{15}, v_{16})$$

Global max pooling

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

Data

global max pooling (

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

) =

m

$$m = \max(v_1, v_2, \ldots, v_{16})$$

keras.layers.GlobalMaxPool2D()

Global Pooling

Average pooling: Features are preserved

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

Data

$$\text{average pooling}\left(\begin{array}{|c|c|c|c|}\hline v_1 & v_2 & v_3 & v_4 \\ \hline v_5 & v_6 & v_7 & v_8 \\ \hline v_9 & v_{10} & v_{11} & v_{12} \\ \hline v_{13} & v_{14} & v_{15} & v_{16} \\ \hline \end{array}\right) = \begin{array}{|c|c|}\hline m_1 & m_2 \\ \hline m_3 & m_4 \\ \hline \end{array}$$

$$m_1 = \text{mean}(v_1, v_2, v_5, v_6)$$
$$m_2 = \text{mean}(v_3, v_4, v_7, v_8)$$
$$m_3 = \text{mean}(v_9, v_{10}, v_{13}, v_{14})$$
$$m_4 = \text{mean}(v_{11}, v_{12}, v_{15}, v_{16})$$

keras.layers. AveragePooling2D (pool_size=2)

Feature map (220x220)

Average Pooling (2x2)

Feature map (110x110)

Average Pooling (2x2)

Feature map (55x55)

Global Pooling

Average pooling

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

Data

average pooling (

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

) =

m_1	m_2
m_3	m_4

$m_1 = \text{mean}(v_1, v_2, v_5, v_6)$

$m_2 = \text{mean}(v_3, v_4, v_7, v_8)$

$m_3 = \text{mean}(v_9, v_{10}, v_{13}, v_{14})$

$m_4 = \text{mean}(v_{11}, v_{12}, v_{15}, v_{16})$

Global average pooling

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

Data

global average pooling (

v_1	v_2	v_3	v_4
v_5	v_6	v_7	v_8
v_9	v_{10}	v_{11}	v_{12}
v_{13}	v_{14}	v_{15}	v_{16}

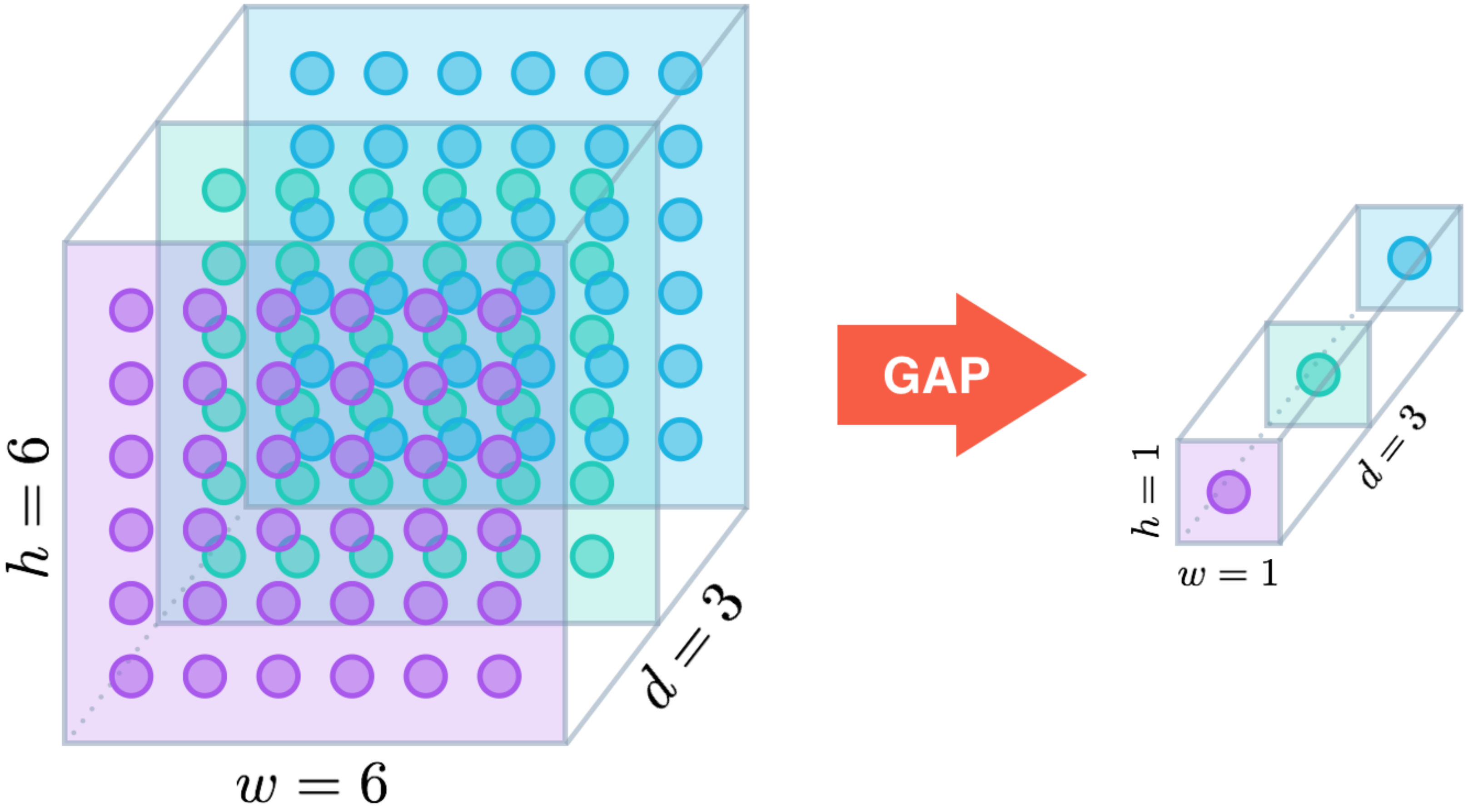
) = m

$m = \text{average}(v_1, v_2, \dots, v_{16})$

keras.layers.GlobalAveragePooling2D()

Object Localization

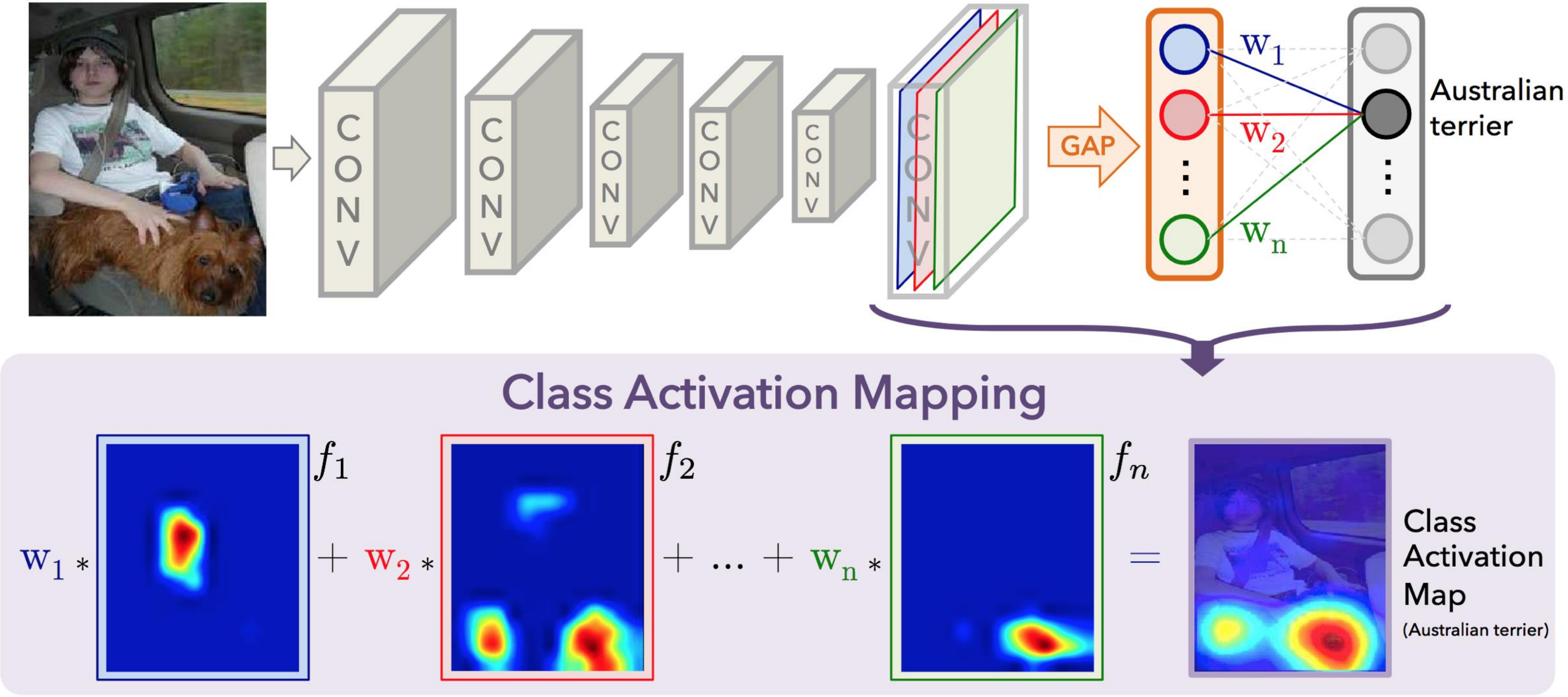
❖ Global Average Pooling



https://alexisbcook.github.io/2017/global-average-pooling-layers-for-object-localization/

Object Localization

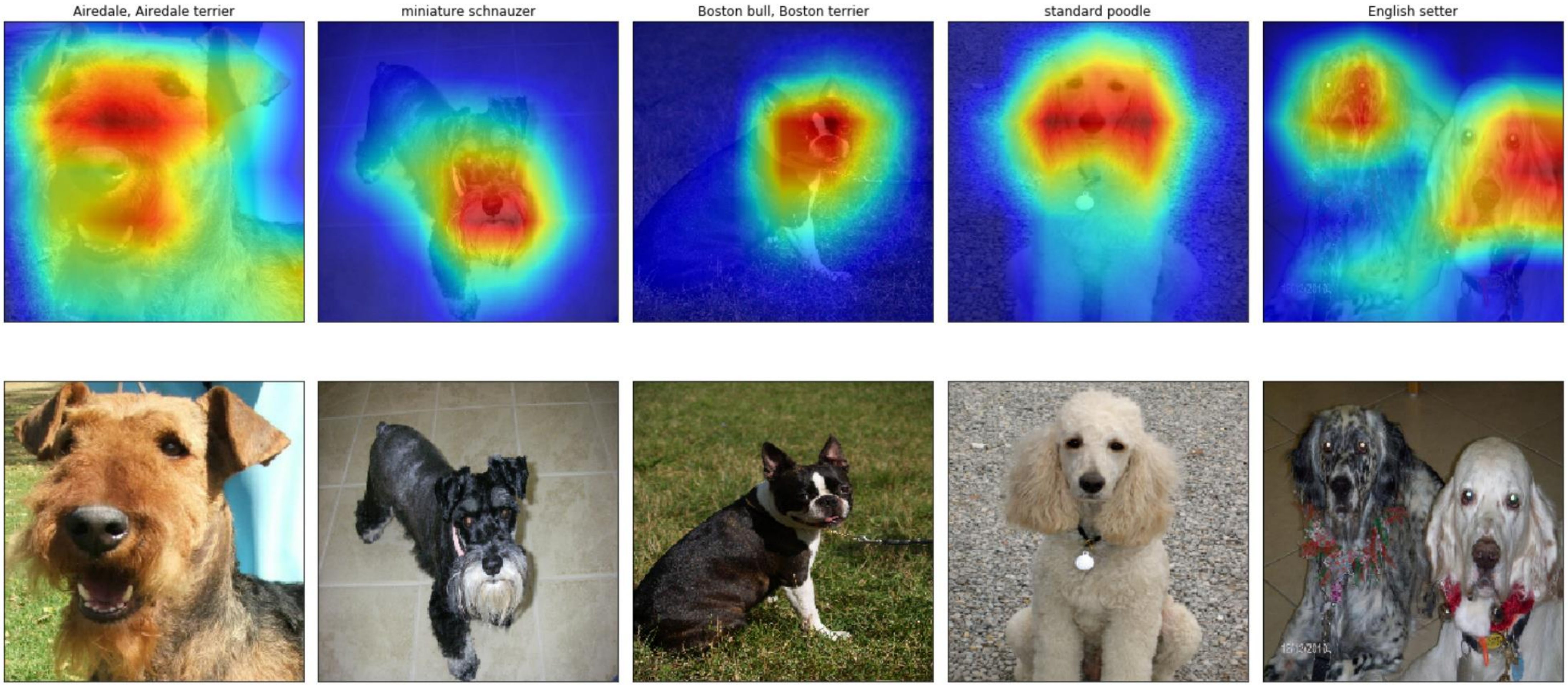
❖ Class Heat Map



$$w_1 \cdot f_1 + w_2 \cdot f_2 + \ldots + w_{2048} \cdot f_{2048}.$$

Object Localization

❖ Class Heat Map



ObjectLocalization.ipynb

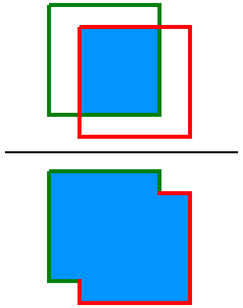
Object Detection

Object Detection Metrics

❖ Intersection Over Union (IOU)

$$IoU = \frac{B_p \cap B_{gt}}{B_p \cup B_{gt}}$$

$$IoU = \frac{area\ of\ overlap}{area\ of\ union} =$$



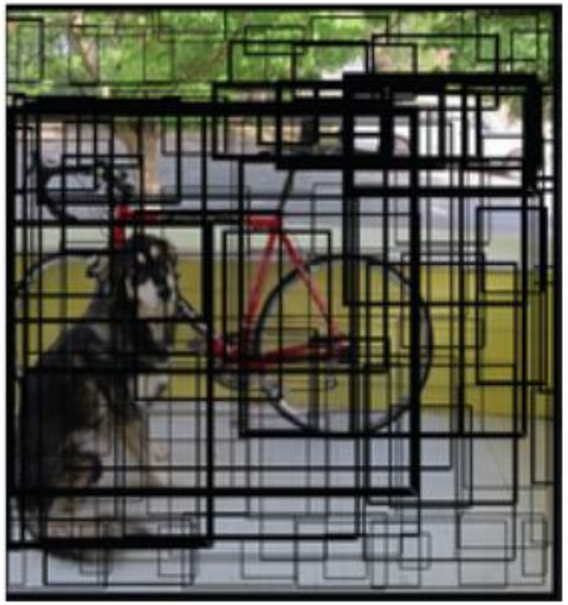
True Positive (TP): A correct detection.
Detection with IOU ≥ *threshold*

False Positive (FP): A wrong detection.
Detection with IOU < *threshold*

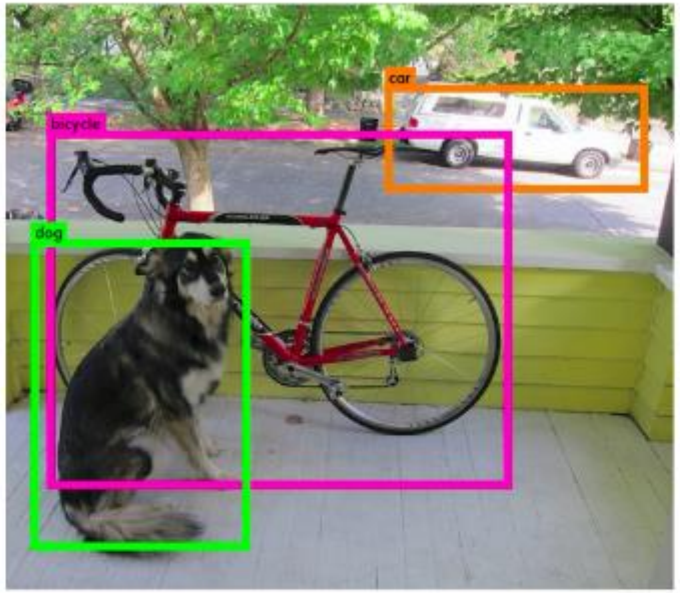
threshold: depending on the metric,
usually set to 50%, 75% or 95%.

Non-max Suppression

❖ Motivation



Multiple Bounding Boxes

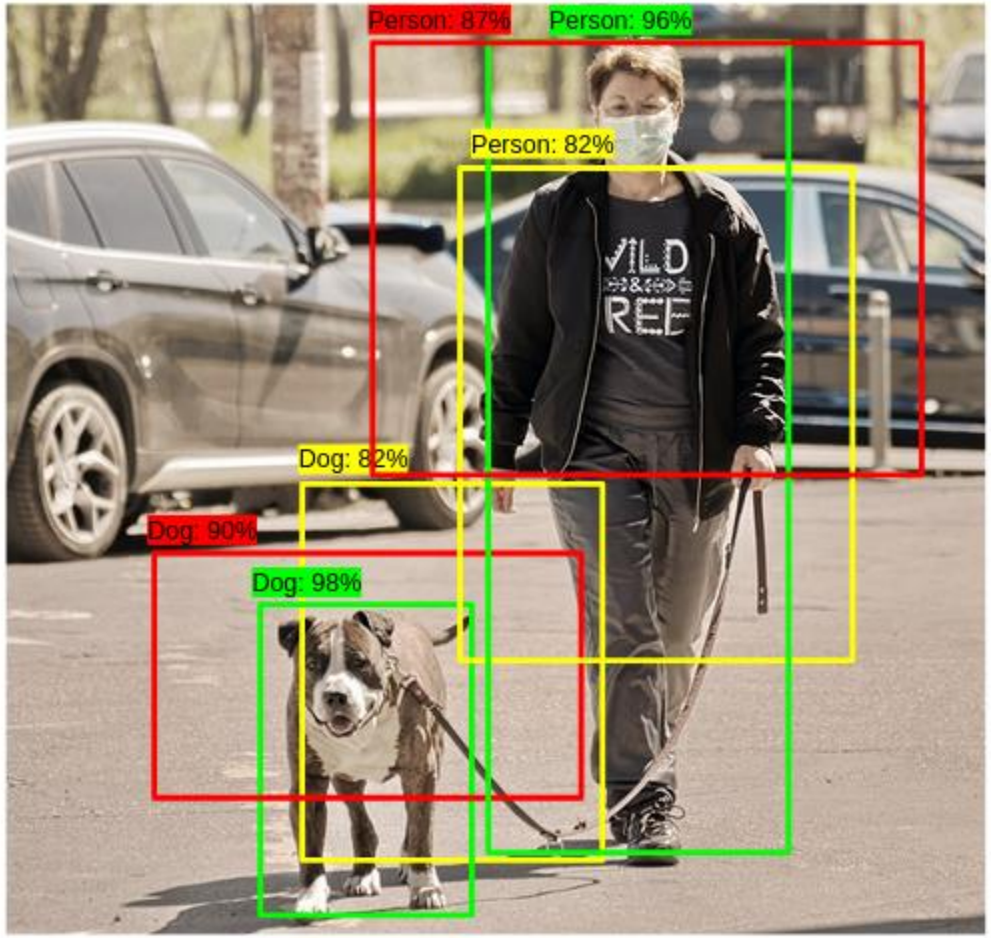


Final Bounding Boxes

https://pjreddie.com/darknet/yolov1/

Non-max Suppression

- Confidence score
- IoU



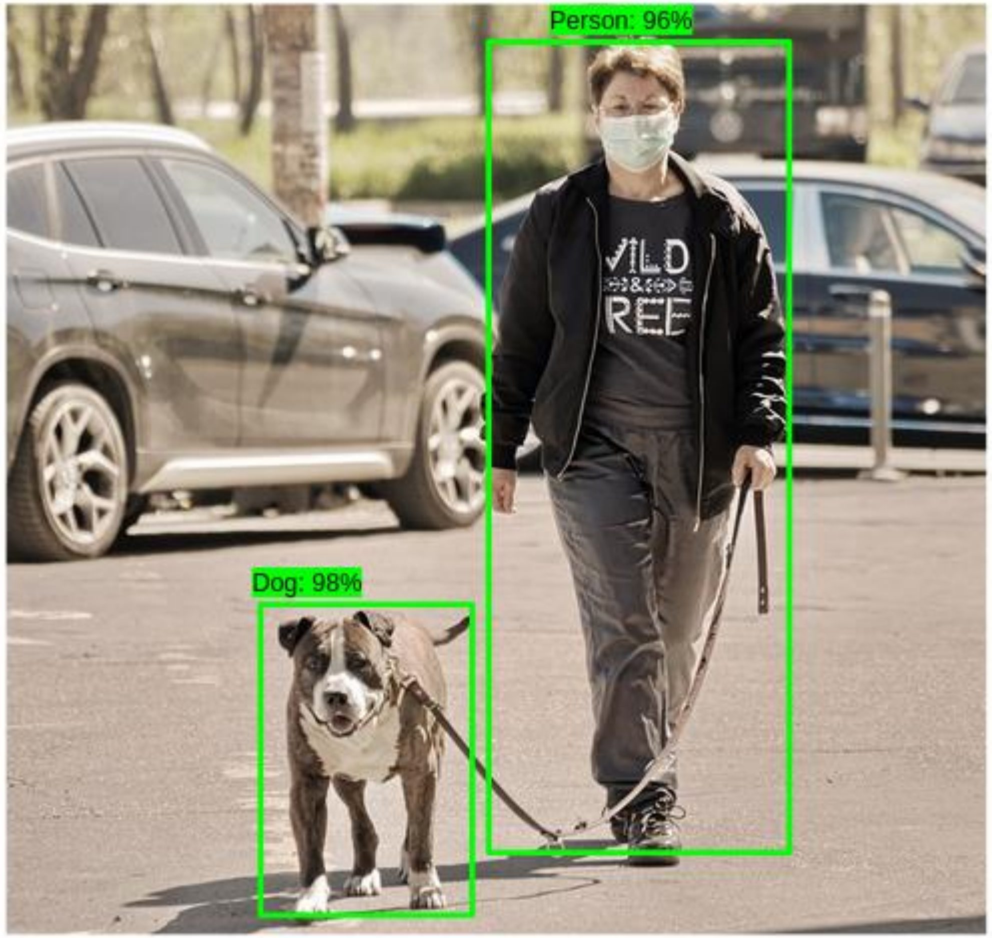
https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

Non-max Suppression

❖ Procedure

Select the bounding box with the highest confidence score

Remove all the other boxes with high overlap



Non-max Suppression

❖ Procedure

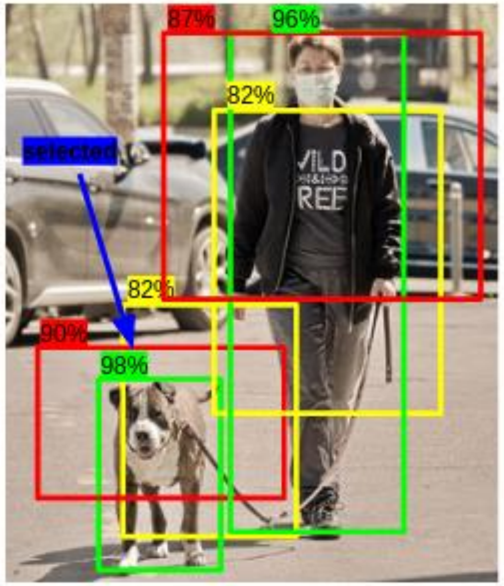
Step 1: Select the box with highest objectiveness score

Step 2: Then, compare the overlap (intersection over union) of this box with other boxes

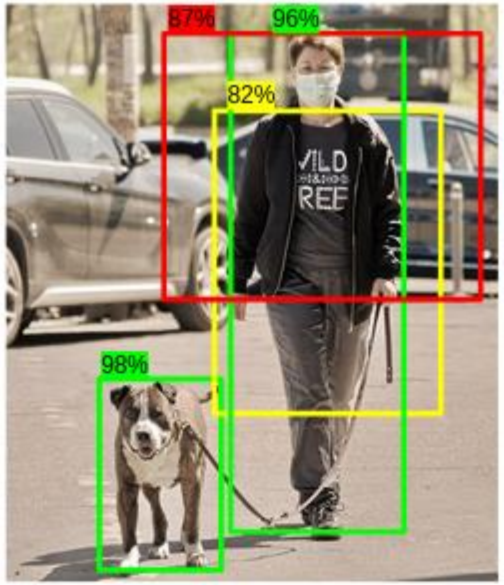
Step 3: Remove the bounding boxes with overlap (intersection over union) >50%

Step 4: Then, move to the next highest objectiveness score

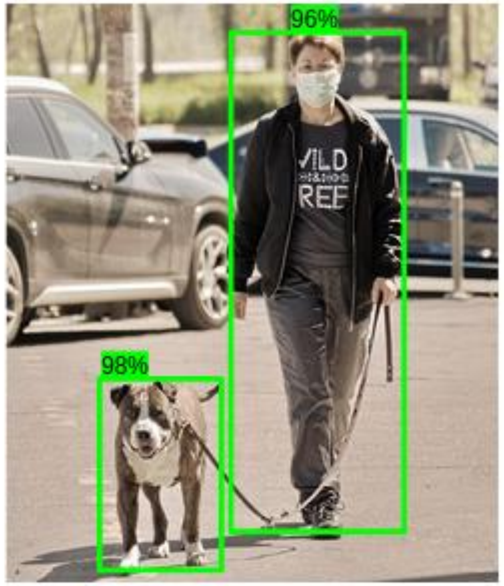
Step 5: Finally, repeat steps 2-4



Step 1: Selecting Bounding box with highest score



Step 3: Delete Bounding box with high overlap



Step 5: Final Output

Non-max-Suppression.ipynb

VOC2007 Dataset

❖ **20 categories**

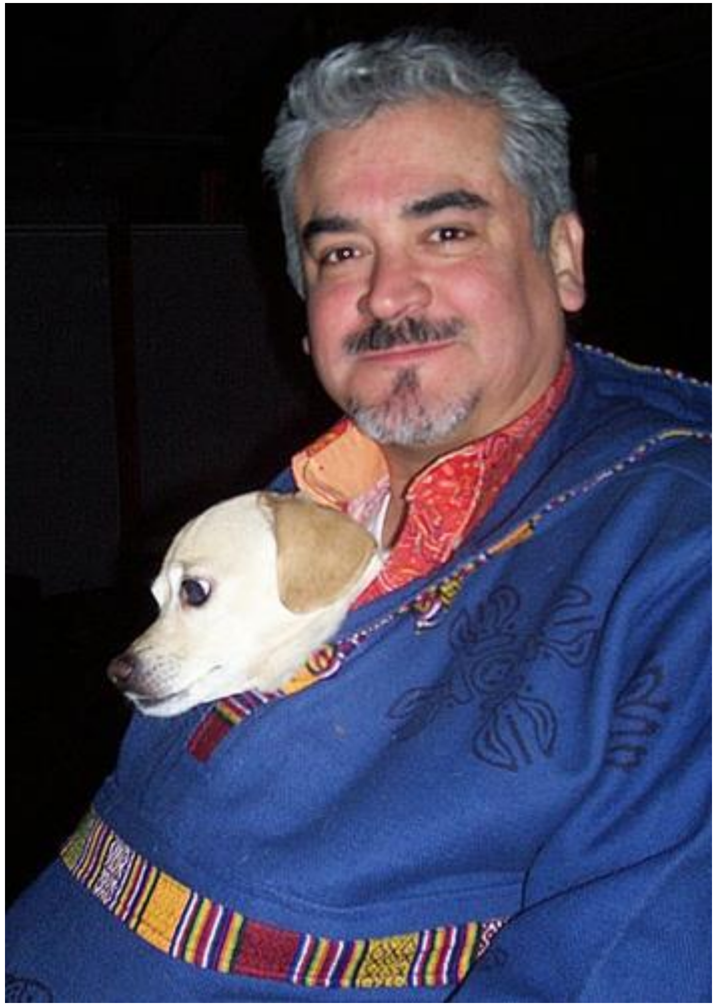
Person: person
Animal: bird, cat, cow, dog, horse, sheep
Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor

VOC2007 Dataset

❖ **20 categories**

VOC2007 Dataset

- **Example**




```xml
<annotation>
    <folder>VOC2007</folder>
    <filename>000001.jpg</filename>
    <source>
        <database>The VOC2007 Database</database>
        <annotation>PASCAL VOC2007</annotation>
        <image>flickr</image>
        <flickrid>341012865</flickrid>
    </source>
    <owner>
        <flickrid>Fried Camels</flickrid>
        <name>Jinky the Fruit Bat</name>
    </owner>
    <size>
        <width>353</width>
        <height>500</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>dog</name>
        <pose>Left</pose>
        <truncated>1</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>48</xmin>
            <ymin>240</ymin>
            <xmax>195</xmax>
            <ymax>371</ymax>
        </bndbox>
    </object>
    <object>
        <name>person</name>
        <pose>Left</pose>
        <truncated>1</truncated>
        <difficult>0</difficult>
        <bndbox>
            <xmin>8</xmin>
            <ymin>12</ymin>
            <xmax>352</xmax>
            <ymax>498</ymax>
        </bndbox>
    </object>
</annotation>
```

VOC2007 Dataset

❖ Data Processing

'aeroplane': 0	'bus': 5	'diningtable': 10	'pottedplant': 15
'bicycle': 1	'car': 6	'dog': 11	'sheep': 16
'bird': 2	'cat': 7	'horse': 12	'sofa': 17
'boat': 3	'chair': 8	'motorbike': 13	'train': 18
'bottle': 4	'cow': 9	'person': 14	'tv/monitor': 19

VOC2007 Dataset

❖ Data Processing

Name

- Annotations
- ImageSets
- JPEGImages
- SegmentationClass
- SegmentationObject

Name

- 000001.xml
- 000002.xml
- 000003.xml
- 000004.xml
- 000005.xml
- 000006.xml
- 000007.xml
- 000008.xml
- 000009.xml
- 000010.xml

Name

- 000001.jpg
- 000002.jpg
- 000003.jpg
- 000004.jpg
- 000005.jpg
- 000006.jpg
- 000007.jpg
- 000008.jpg
- 000009.jpg
- 000010.jpg

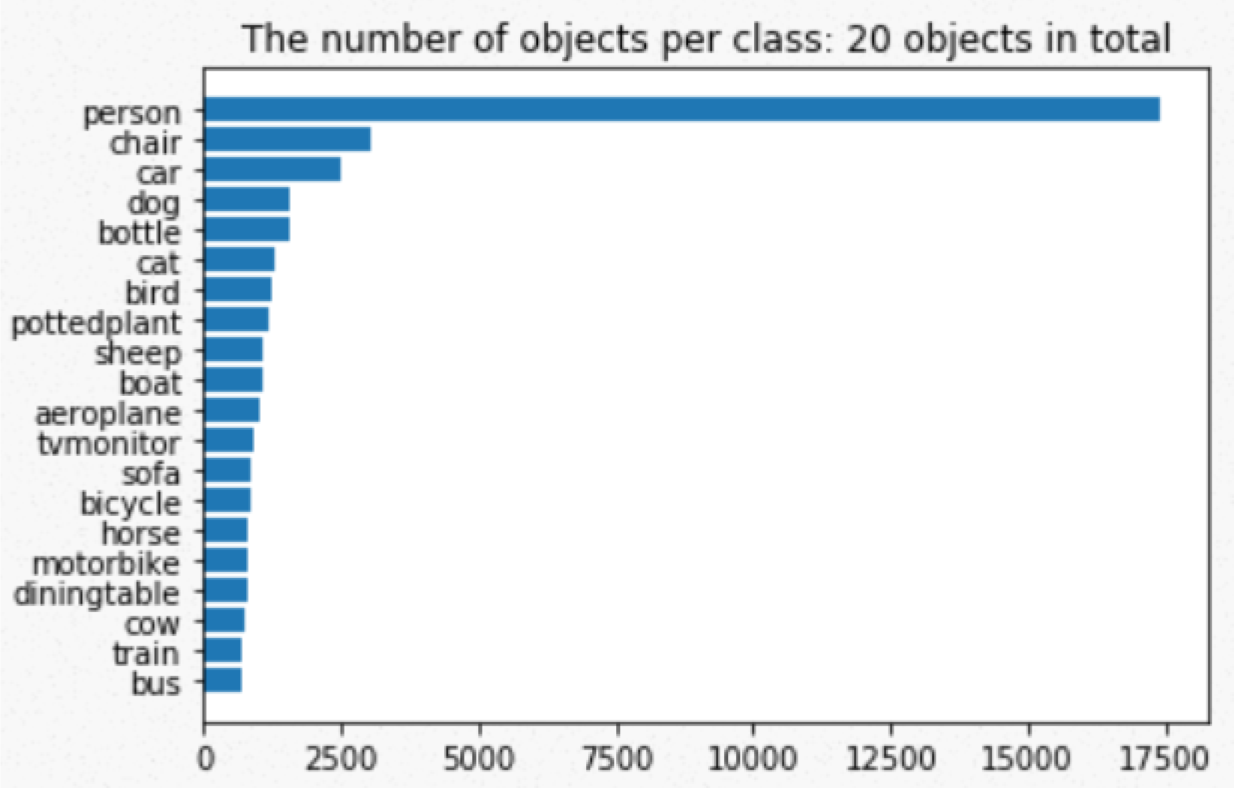
VOC2007 Dataset

❖ Data Processing

```
VOCdevkit/VOC2007/JPEGImages/000153.jpg 237,147,358,191,6
VOCdevkit/VOC2007/JPEGImages/000154.jpg 59,76,367,266,3
VOCdevkit/VOC2007/JPEGImages/000159.jpg 234,48,286,124,14 1,16,498,333,6
VOCdevkit/VOC2007/JPEGImages/000161.jpg 104,34,446,390,6 68,195,121,288,6
VOCdevkit/VOC2007/JPEGImages/000162.jpg 306,227,380,299,19 196,143,309,369,14
VOCdevkit/VOC2007/JPEGImages/000163.jpg 52,22,308,328,14 26,108,456,396,13
VOCdevkit/VOC2007/JPEGImages/000164.jpg 114,154,369,348,13 292,49,446,370,14
VOCdevkit/VOC2007/JPEGImages/000171.jpg 1,290,128,407,11 94,21,375,491,14
VOCdevkit/VOC2007/JPEGImages/000173.jpg 106,64,270,297,14 109,64,288,464,12
VOCdevkit/VOC2007/JPEGImages/000174.jpg 143,5,426,333,14
VOCdevkit/VOC2007/JPEGImages/000187.jpg 1,95,240,336,19
VOCdevkit/VOC2007/JPEGImages/000189.jpg 65,39,459,346,2
VOCdevkit/VOC2007/JPEGImages/000192.jpg 116,64,356,375,14
VOCdevkit/VOC2007/JPEGImages/000193.jpg 80,4,500,375,14 1,29,227,375,14
VOCdevkit/VOC2007/JPEGImages/000194.jpg 86,36,239,224,12 115,19,203,136,14 279,77,298,132,14
VOCdevkit/VOC2007/JPEGImages/000198.jpg 160,134,286,239,18
```

VOC2007 Dataset

❖ Statistics



	train		val		trainval	
	Images	**Objects**	**Images**	**Objects**	**Images**	**Objects**
Aeroplane	112	151	126	155	238	306
Bicycle	116	176	127	177	243	353
Bird	180	243	150	243	330	486
Boat	81	140	100	150	181	290
Bottle	139	253	105	252	244	505
Bus	97	115	89	114	186	229
Car	376	625	337	625	713	1250
Cat	163	186	174	190	337	376
Chair	224	400	221	398	445	798
Cow	69	136	72	123	141	259
Diningtable	97	103	103	112	200	215
Dog	203	253	218	257	421	510
Horse	139	182	148	180	287	362
Motorbike	120	167	125	172	245	339
Person	1025	2358	983	2332	2008	4690
Pottedplant	133	248	112	266	245	514
Sheep	48	130	48	127	96	257
Sofa	111	124	118	124	229	248
Train	127	145	134	152	261	297
Tvmonitor	128	166	128	158	256	324
Total	2501	6301	2510	6307	5011	12608

VOC2007 Dataset

❖ Data Processing

```python
import matplotlib.pyplot as plt
import cv2
import numpy as np

# read an image
image = cv2.imread('VOCdevkit/VOC2007/JPEGImages/000026.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
print(image.shape)

# draw bounding boxes
color = (255, 0, 0)
thickness = 2
image = cv2.rectangle(image, (90,125), (337,212),  color, thickness)

# plot image
fig = plt.figure()
plt.imshow(image/255.0)
```

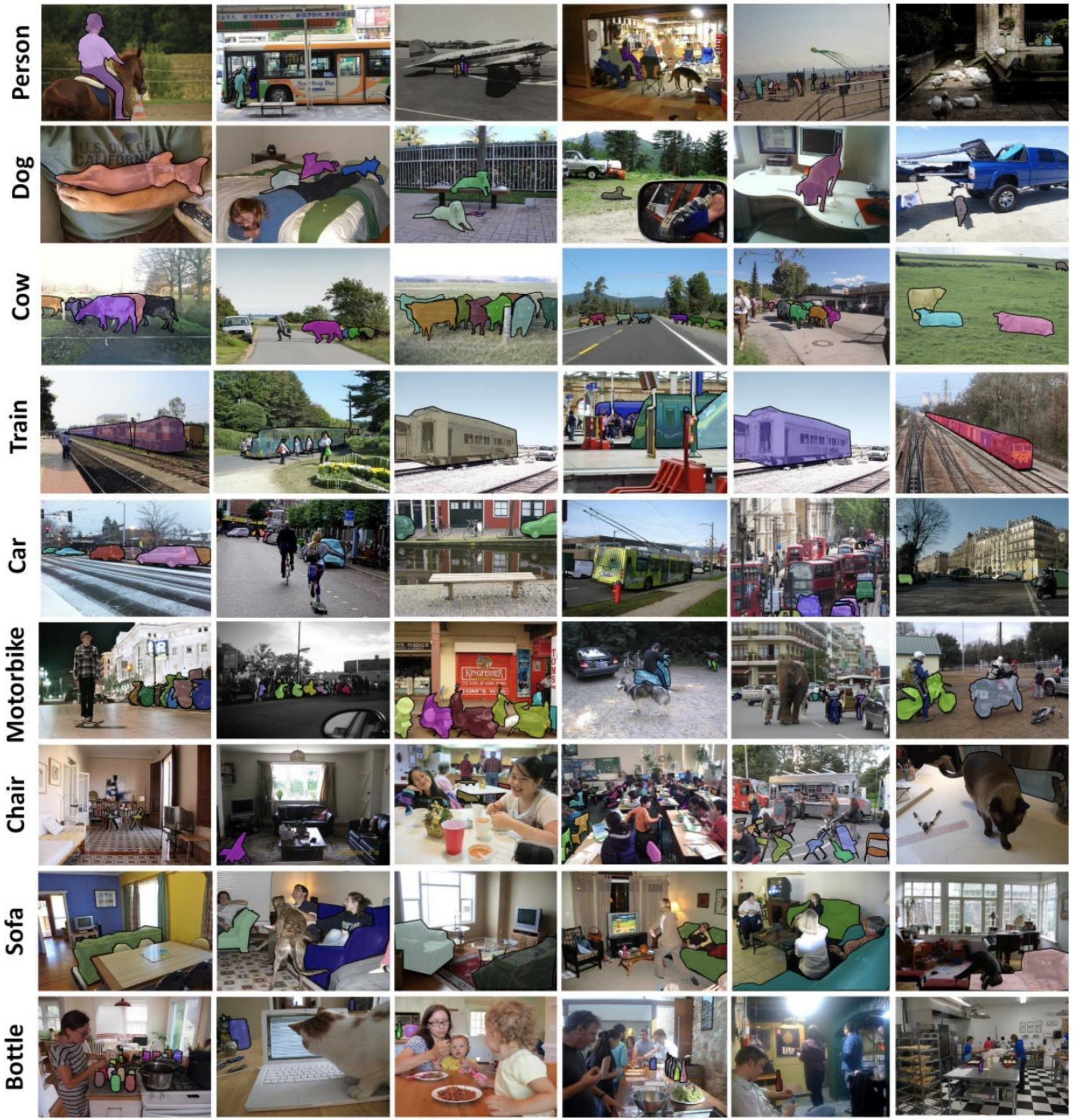
```
(333, 500, 3)
```

VOC2007Dataset.ipynb

COCO Dataset

❖ **Common Objects in Context**

121,408 images
883,331 object annotations
80 classes



COCO Dataset

❖ Common Objects in Context

person	10,777
car	1,918
chair	1,771
book	1,129
bottle	1,013
cup	895
diningtable	695
traffic light	634
bowl	623
handbag	540
bird	427
boat	424
truck	414
bench	411
umbrella	407
cow	372
backpack	371
banana	370
motorbike	367
carrot	365
sheep	354
pottedplant	342
wine glass	341

donut	328
kite	327
knife	325
bicycle	314
broccoli	312
cake	310
suitcase	299
tvmonitor	288
orange	285
pizza	284
bus	283
remote	283
vase	274
horse	272
clock	267
surfboard	267
zebra	266
cell phone	262
sofa	261
sports ball	260
spoon	253
elephant	252
tie	252
skis	241
apple	236
giraffe	232
laptop	231
sink	225
tennis racket	225
dog	218
fork	215
cat	202
teddy bear	190

Object Detection

❖ Use as blackbox

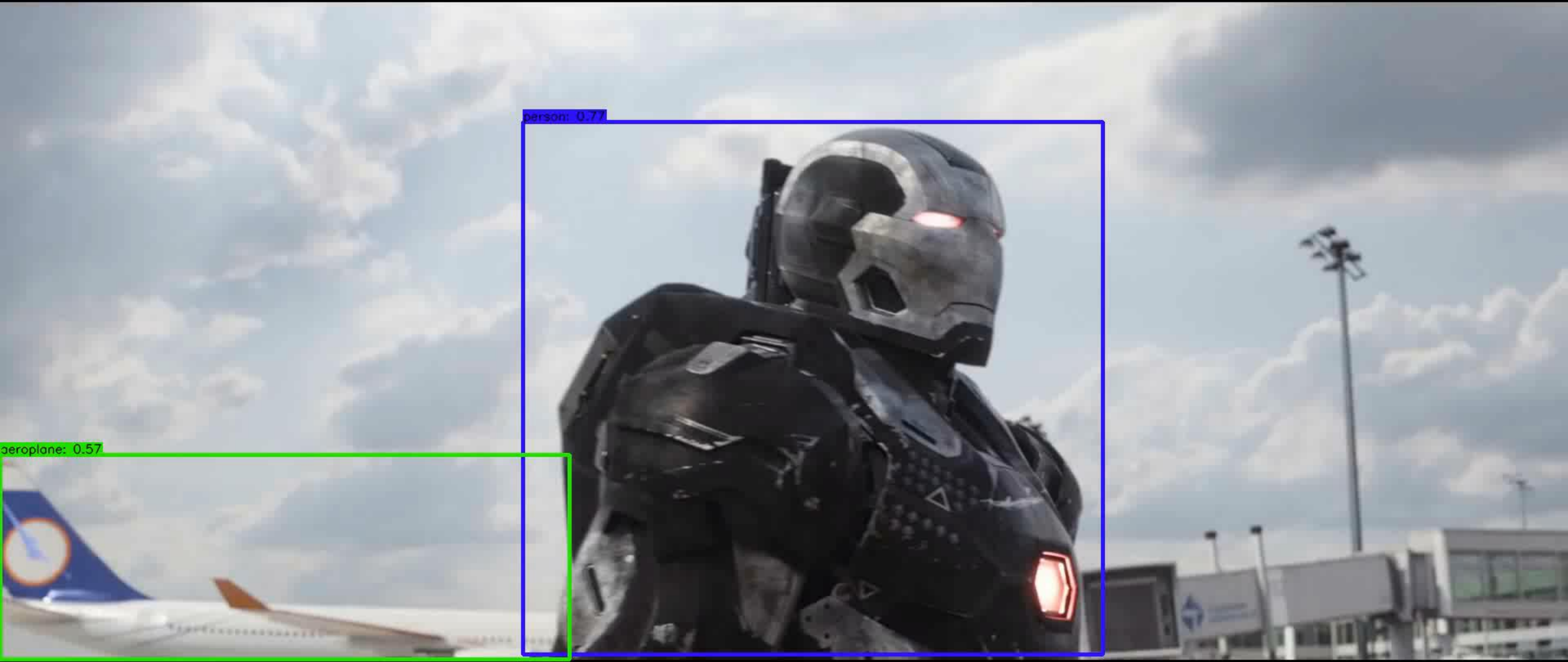
yolov4-custom-functions

license MIT

A wide range of custom functions for YOLOv4, YOLOv4-tiny, YOLOv3, and YOLOv3-tiny implemented in TensorFlow, TFLite and TensorRT.

DISCLAIMER: This repository is very similar to my repository: tensorflow-yolov4-tflite. I created this repository to explore coding custom functions to be implemented with YOLOv4, and they may worsen the overal speed of the application and make it not optimized in respect to time complexity. So if you want to run the most optimal YOLOv4 code with TensorFlow than head over to my other repository. This one is to explore cool customizations and applications that can be created using YOLOv4!

https://github.com/theAIGuysCode/yolov4-custom-functions


person: 0.77
aeroplane: 0.57

Object Detection

❖ Use as blackbox

```
# load model
saved_model_loaded = tf.saved_model.load(flags_weights, tags=[tag_constants.SERVING])

# load image
original_image = cv2.imread(image_path)
original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)

image_data = cv2.resize(original_image, (input_size, input_size))
image_data = image_data / 255.

# get image name by using split method
image_name = image_path.split('/')[-1]
image_name = image_name.split('.')[0]
```

Object Detection

❖ Use as blackbox

```
#load model and detect objects
infer = saved_model_loaded.signatures['serving_default']
batch_data = tf.constant(images_data)
pred_bbox = infer(batch_data)
for key, value in pred_bbox.items():
    boxes = value[:, :, 0:4]
    pred_conf = value[:, :, 4:]

# run non max suppression on detections
boxes, scores, classes, valid_detections = tf.image.combined_non_max_suppression(
    boxes=tf.reshape(boxes, (tf.shape(boxes)[0], -1, 1, 4)),
    scores=tf.reshape(
        pred_conf, (tf.shape(pred_conf)[0], -1, tf.shape(pred_conf)[-1])),
    max_output_size_per_class=50,
    max_total_size=50,
    iou_threshold=flags_iou,
    score_threshold=flags_score
)
```

Object Detection

- ❖ **Use as black-box**
- ❖ **Detect for a specific object**

Thank you!