**《Fundamentals of Programming》**

**Curriculum Design Report**

Subject ： Computer Science And Technology

Specialty ：Computer Science

Class ： 19lq CST

Student’s Name ： Patwary Mohammad Sazzad Hossain

School Number ： 1911521212

Teacher ：

20 20 / 12 / 19 ～ 20 20 / 12 / 29 /

**Produced by School of Computer**

**Menu**

**1. Requirements Analysis**

**2. General Design**

**2.1 YOLO detector**

**2.2 SSD detector:**

**2.3 COCO dataset**

**3. Detailed Design, Tests and Analyses of Program Running Results**

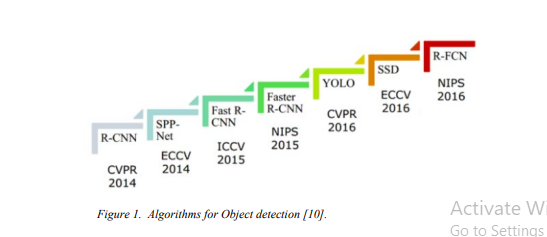
**4. Complimentary Close**

**5. References**

**The Main Body**

**1. Requirements Analysis**

Object detection, detecting instances of semantic objects, is a computer vision and image processing technology . A decade ago, distinction any object such as a person, animal, building, car, human face and so on, via computer was considered an unattainable task, especially like the difference between a cat and a dog even with a significant advance in the state of artificial intelligence . However, in 2012, Alex Krizhevsky et al. designed a convolutional neural network model called AlexNet, which significantly improved the performances in labeling pictures (classification) in the ImageNet challenge. and outperformed all previous models. Because of this breakthrough in the convolution neural network, many other CNN models were springing up all over this field such as VGGNet, inceptionNet, and ResNet and so forth. Nonetheless, the convolution neural network has been studied since the 90s. Why had it taken so long to reach this turning point? In addition to the contribution to the development of CNN on classification from Alex Krizhevsky, there were also two catalysts leading CNN to this flourish, which were first, GPUs – giving lots of computing power and second, large dataset – increasing accuracy via training. Although these approaches are impressive, image classification is too simple than the human’s complex visual understanding because, in reality, a complicated scene we view are objects and different backgrounds overlapping each other. Except for classifying these objects, we need to identify their locations, boundaries, and differences, or relations. That is the reason why the term, object detection, come out in several years later from 2012. Recently years, object detection has been employed ubiquitously from the use of video surveillance, tracking object, and pedestrian recognition, just to name a few. There are multiple deep learning approaches for computer vision systems such as Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN), Single Shot MultiBox Detector (SSD), You Only Look Once (YOLO) and so on as the figure 1 shown below. Joseph Redmon, the designer of the YOLO algorithm, claimed that the detectors nowadays are able to detect objects with a highly accurate detection rate, at a level greater than 99 percent accuracy [7]. Figure 1.

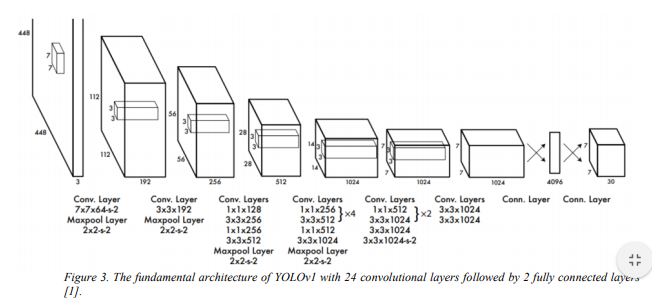


Algorithms for Object detection [10]. In this project, we are going to explain how to evaluate the performances for an object detector and analyze both the YOLO model (YOLOv2, and YOLOv3) and the SSD model in accuracy and speed. In addition, I will implement three object detectors in CPU (central processing unit) with python language in PyCharm, 8 an integrated development environment (IDE) design particularly for python by using the given pre-trained weight and configuration of the YOLOv2, YOLOv3, and SSD convolution neural network algorithms.

**2.General Design**

**2.1 YOLO detector:**

You only look once (YOLO), a single convolutional neural network different from prior detectors, frames object as a regression problem to spatially separated bounding boxes and associated class probabilities directly from full images in one evaluation [1]. The full network YOLOv1 architecture with 24 convolutional layers and 2 fully connected layers is shown below in figure 3. In our project, we are using YOLOv3, rather than using YOLOv1, as our model to detect objects.



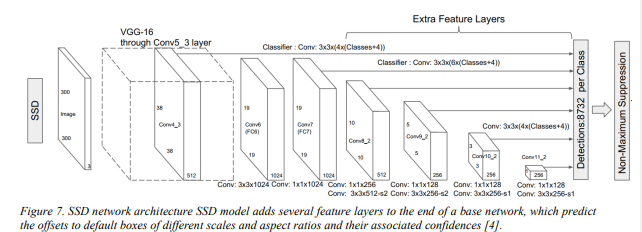
**2.2 SSD detector:**

Single Shot MultiBox Detector (SSD), a single convolutional neural network, is not complex compared with other methods providing object proposals that are removed by SSD. The SSD architecture in figure 7 is separated into two parts, a base network contributing high quality image classification applied to the front (SSD takes the VGG16 network as the base network to extract feature maps.), and several convolutional feature layers added afterward to predict objects detection. There are several features in the SSD model [4]:

• Multi-scale feature maps for detection: As for the rear network. The sizes of convolutional feature layers added decrease gradually, which allows us to predict objects at different scales. Besides, each feature layer uses a different convolutional model to predict detections.

• Convolutional predictors for detection: Each added feature layer or any existing feature layer from the base network can generate a fixed set of detection predictions using a set of convolutional filters that are displayed on the top of SSD architecture in Figure 7. For a feature layer with the size m × n and p channel, SSD applies small convolution filters that are 3 × 3 to calculate the location and class scores for each cell to make predictions for a fixed set of the default bounding box. Each 15 contains its own boundary with offset shape to its default box and scores for all classes plus one for no object present (a class 0 is reserved for no object). The YOLO model, instead of using a convolutional filter, adopts an intermediate fully connected layer that is discarded by SSD.

• Default boxes and aspect ratios: A set of default bounding boxes with each feature map cell are associated together in terms of multiple feature maps at the front network. The position of each default boxes relative to its corresponding cell is fixed. At each grid cell in feature map, the offsets relative to the default box shapes in the cell, and also the per-class scores that indicate the presence of a class instance in each of those boxes are predicted. To be Specific, for each box out of k at a given location, c class scores and the 4 offsets are calculated relative to the original default box shape. These results in a total of (c + 4)k filters that are applied around each location in the feature map, yielding (c + 4)kmn outputs for a m × n feature map.”



**2.3 COCO dataset**

Common Object in Context (COCO) is a large-scale image dataset created for object detection, segmentation, person key-points detection, stuff, and caption. Its dataset contains 330K images, 1.5 million object instances along with greater than 200K labeled. All of the objects are classified into 80 categories. It also provides Matlab, Python, and Lua APIs, which support programmer to load, parse, and visualize the annotations of all images. More information like data, paper, annotation format, are described in the COCO website. Taking the annotation for object detection as an example, the format looks like below [19]:

Annotation {

"id" : int,

"image\_id" : int,

"category\_id" : int,

"segmentation" : RLE or [polygon],

"area" : float,

"bbox" : [x, y, width, height],

"iscrowd" : 0 or 1, }

Categories [{ "id" : int,

"name" : str,

"supercategory" : str, }]

**3. Detailed Design, Tests And Analyses of Program Running Results**

**Object Detection - COCO**



This algorithm is able to discover not only what's in an image, but where it is too! It discovers the location within an image and generates a [bounding box annotation](https://en.wikipedia.org/wiki/Minimum_bounding_box). This algorithm uses the latest pre-trained models from Google's [Tensorflow](https://www.tensorflow.org/" \t "_blank) [object detection](https://github.com/tensorflow/models/tree/master/object_detection) project. All available models were all trained on the [Common Objects in Context](http://cocodataset.org/) Dataset.

A list of all available labels is below.

**Table of Contents**

* [I/O](https://algorithmia.com/algorithms/deeplearning/ObjectDetectionCOCO/docs#io)
* [labels](https://algorithmia.com/algorithms/deeplearning/ObjectDetectionCOCO/docs#labels)
* [Examples](https://algorithmia.com/algorithms/deeplearning/ObjectDetectionCOCO/docs#examples)
* [credits](https://algorithmia.com/algorithms/deeplearning/ObjectDetectionCOCO/docs#credits)
* [Algorithm Console](https://algorithmia.com/algorithms/deeplearning/ObjectDetectionCOCO/docs#console)

**I/O**

**Input**

{

"image": String,

"output": String,

"max\_boxes": Integer,

"min\_score": Float,

"model": String

}

* image - ***(required)*** - a hosted image file, may be a web url (http, https) or a data connector URI (data://, s3://, etc).
* output - ***(optional)*** - the output data connector URI (data://, s3://, etc) for the resultant annotated image. *If output is not provided, only the bounding box data is returned.*
* max\_boxes - ***(optional)*** - the maximum number of bounding boxes to return in the results. *If max\_boxes is not defined, it defaults to 20*
* min\_score - ***(optional)*** - the minimum score threshold for bounding box annotations, if a prediction's confidence is less than this minimum, it's not returned in the results. \_If min\_score is not defined, it defaults to 0.5
* models - ***(optional)*** - the pre-trained object detection model to use, may be any of the following:

| **Model name** | **compute time per image** | **COCO accuracy (mAP)** |
| --- | --- | --- |
| ssd\_mobilenet\_v1 | 4.78 | 21 |
| ssd\_inception\_v2 | 8.75 | 24 |
| rfcn\_resnet101 | 10.25 | 30 |
| faster\_rcnn\_resnet101 | 11.05 | 32 |
| faster\_rcnn\_inception\_resnet\_v2\_atrous | 16.78 | 37 |

*when model is not defined, defaults to ssd\_mobilenet\_v1*

**Alternatively you can just pass a url directly to the algorithm as a string.**

**Output**

{

"image": String,

"boxes": [

{

"coordinates": {

"x0": Float,

"y0": Float,

"x1": Float,

"y1": Float

},

"label": String,

"confidence": Float

},

...

]

}

* image - The bounding box annotated image (if output was defined) data connector URI.
* boxes - a list of all detected objects and their bounding boxes.
* coordinates - the absolute cartesian coordinates of the bounding box found in the specimen image.
* label - the predicted label/class for the detected object
* confidence - the confidence of the class prediction (0 -> 1)

**Examples**

**Example 1 - Street car**

**Input**



{

"image":"http://i.imgur.com/k67kjlB.jpg",

"output":"data://.algo/temp/streetcar.png",

"model":"ssd\_inception\_v2"

}

**Output**



{

"boxes":[

{

"confidence":0.9845596551895142,

"coordinates":{

"x0":276,

"x1":490,

"y0":170,

"y1":386

},

"label":"train"

},

{

"confidence":0.8447293043136597,

"coordinates":{

"x0":36,

"x1":132,

"y0":280,

"y1":314

},

"label":"car"

},

{

"confidence":0.7735579609870911,

"coordinates":{

"x0":520,

"x1":555,

"y0":273,

"y1":305

},

"label":"car"

}

],

"image":"data://.algo/temp/streetcar.png"

}

**Example 2 - Dog Park**

**Input**



{

"image":"http://i.imgur.com/1IWZX69.jpg",

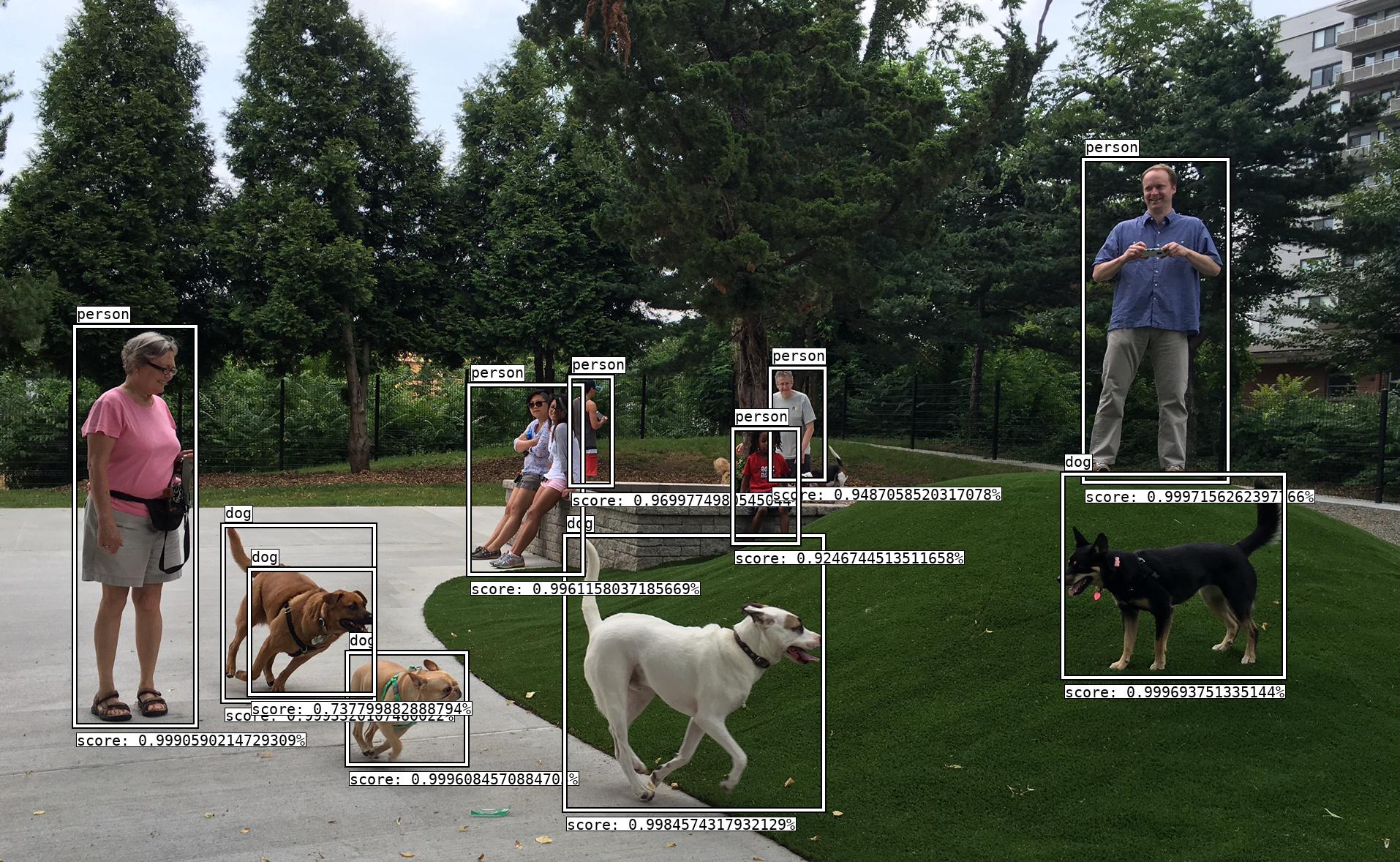
"output":"data://.algo/temp/dog\_park.png",

"min\_score":0.7,

"model":"faster\_rcnn\_resnet101"

}

**Output**



{

"boxes":[

{

"confidence":0.9997156262397766,

"coordinates":{

"x0":1484,

"x1":1675,

"y0":221,

"y1":655

},

"label":"person"

},

{

"confidence":0.999693751335144,

"coordinates":{

"x0":1456,

"x1":1752,

"y0":651,

"y1":922

},

"label":"dog"

},

{

"confidence":0.9996084570884703,

"coordinates":{

"x0":478,

"x1":634,

"y0":895,

"y1":1041

},

"label":"dog"

},

{

"confidence":0.9993320107460022,

"coordinates":{

"x0":308,

"x1":508,

"y0":721,

"y1":954

},

"label":"dog"

},

{

"confidence":0.9990590214729308,

"coordinates":{

"x0":105,

"x1":264,

"y0":449,

"y1":988

},

"label":"person"

},

{

"confidence":0.9984574317932128,

"coordinates":{

"x0":775,

"x1":1122,

"y0":735,

"y1":1103

},

"label":"dog"

},

{

"confidence":0.9961158037185668,

"coordinates":{

"x0":644,

"x1":793,

"y0":529,

"y1":781

},

"label":"person"

},

{

"confidence":0.9699774980545044,

"coordinates":{

"x0":782,

"x1":833,

"y0":518,

"y1":660

},

"label":"person"

},

{

"confidence":0.9487058520317078,

"coordinates":{

"x0":1056,

"x1":1124,

"y0":506,

"y1":652

},

"label":"person"

},

{

"confidence":0.9246744513511658,

"coordinates":{

"x0":1005,

"x1":1088,

"y0":589,

"y1":739

},

"label":"person"

},

{

"confidence":0.737799882888794,

"coordinates":{

"x0":344,

"x1":508,

"y0":781,

"y1":945

},

"label":"dog"

}

],

"image":"data://.algo/temp/dog\_park.png"

}

**Labels**

The dataset that this algorithm was trained on has 90 possible labels, here's a list for easy reference:

- person

- bicycle

- car

- motorcycle

- airplane

- bus

- train

- truck

- boat

- traffic light

- fire hydrant

- stop sign

- parking meter

- bench

- bird

- cat

- dog

- horse

- sheep

- cow

- elephant

- bear

- zebra

- giraffe

- backpack

- umbrella

- handbag

- tie

- suitcase

- frisbee

- skis

- snowboard

- sports ball

- kite

- baseball bat

- baseball glove

- skateboard

- surfboard

- tennis racket

- bottle

- wine glass

- cub

- fork

- knife

- spoon

- bowl

- banana

- apple

- sandwich

- orange

- broccoli

- carrot

- hot dog

- pizza

- donut

- cake

- chair

- couch

- potted plant

- bed

- dining table

- toilet

- tv

- laptop

- mouse

- remote

- keyboard

- cell phone

- microwave

- oven

- toaster

- sink

- refigerator

- book

- clock

- vase

- scissors

- teddy bear

- hair drier

- toothbrush

**Credits**

**4.Complimentary Close**

According to our implementation results of one image and video, and real time object detection. It is quite obvious that YOLO performed more accurate than SSD because some objects in the same image, or video were not detected by SSD pre-trained model but were detected by YOLO pre-trained model simply through observation. However, is also very clear for speed in the image and video detection between two models but opposite in contrast to accuracy that SSD surpassed the YOLO model. Admittedly. YOLO has better performance in accuracy while SSD has a better performance in speed. If users care more about the accuracy of detection, it is better to choose YOLO, while SSD might be applied if speed is taken into account. Or if users need high accuracy, but a slight decrease in accuracy can be acceptable, then it might be a good alternative to switch to the YOLOv2 model. There is always a tradeoff between speed and accuracy depending on what we need and focus on. In this project, we did a small-scale evaluation of the rate in speed and accuracy through image, video, and real time instead of going through the process of mAP calculation due to the time, financial limitation. In order to process a large scale of the dataset, we have to possess a very powerful and expensive GPU to deal with it, and it still takes a great time to complete the process. Besides, if we are based on the same dataset such as all the same pictures and data from COCO, the result of the evaluation of performance in accuracy will be almost the same as previous studies due to the same model trained with the same images. These results of the mAP studied by researchers previously can be searched online. Therefore, we decide to use the random images, videos and real-time camera to measure the performance between two detectors directly. However, these data do not have a ground truth box marked in an annotation file. That is another reason we do not follow the abovementioned steps to compute the mAP. Hence, in the future we might consider evaluates mAP through a dataset with its provided ground truth to verify our output if we would like to prove that we can get resemble consequence, or further plan on creating our own small dataset and manually draw ground truth box with label attached for each object, so we can evaluate our own training model with the YOLO or SSD architectures and gradually increase our data to a certain reasonable level, which might be considered to be more objective. safety, law and culture can be considered and evaluated When you design it.)

**5. References**

1. J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You only look once: Unified, real-time object detection". Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

2. J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger". Proceedings of the IEEE conference on computer vision and pattern recognition, arXiv:1612.08242 [cs.CV], 2017.

3. J. Redmon and A. Farhadi, "Yolov3: An incremental improvement". Preprint arXiv:1804.02767 [cs.CV], 2018.

4. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European Conference on Computer Vision (ECCV), 2016, pp. 21–37.

5. Pulli, Kari; Baksheev, Anatoly; Kornyakov, Kirill; Eruhimov, Victor (1 April 2012). "Realtime Computer Vision with OpenCV". Queue: 40:40–40:56. doi:10.1145/2181796.2206309 (inactive 2019- 07-14).

6. Object detection, <https://en.wikipedia.org/wiki/Object_detection>

7. How computers learn to recognize objects instantly, https://www.ted.com/talks/joseph\_redmon\_how\_a\_computer\_learns\_to\_recognize\_objects\_instantly# t-12277, April 2017, Redmon, Joseph’s speech in TED. 8. Joseph Chet Redmon’s personal websit, <https://pjreddie.com/>

8. Joseph Chet Redmon’s personal websit, https://pjreddie.com/

9. Intel unveils the intel neural compute stick 2 https://newsroom.intel.com/news/intel-unveils-intelneural-compute-stick-2/#gs.sbwxdp, Nov 14 th, 2018.

10. Beginner’s Guide to Object Detection Algorithms, https://towardsdatascience.com/beginners-guide-toobject-detection-algorithms-6620fb31c375, April 27 th, 2019.

11. Object Detection YOLO v1, v2, v3, https://medium.com/@venkatakrishna.jonnalagadda/objectdetection-yolo-v1-v2-v3-c3d5eca2312a, Jan 30 th, 2019.

12. A Beginner’s Guide to Object Detection, https://www.datacamp.com/community/tutorials/objectdetection-guide, April 19, 2018

13. OpenCV, <https://en.wikipedia.org/wiki/OpenCV>

14. Object Detection and Image Classification with YOLO, https://www.kdnuggets.com/2018/09/objectdetection-image-classification-yolo.html, Michal Maj, Sep, 2018.

15. An overview of object detection: one-stage methods, https://www.jeremyjordan.me/object-detectionone-stage/#ssd, Jeremy Jordan, July 11th, 2018. 16. Understanding YOLO, https://hackernoon.com/understanding-yolo-f5a74bbc7967, Mauricio Menegaz, October 14th, 2019.

17. mAP (mean Average Precision) for Object Detection, https://medium.com/@jonathan\_hui/map-meanaverage-precision-for-object-detection-45c121a31173, Joanthan Hui, March 6th, 2018

18. An Introduction to Evaluation Metrics for Object Detection, https://blog.zenggyu.com/en/post/2018- 12-16/an-introduction-to-evaluation-metrics-for-object-detection/, Nick Zeng, Dec 6th, 2018

19. Common Objects in Context, <http://cocodataset.org/>

20. Pyimagesearch, https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-objectdetection/, Adrian Rosebrock

21. GluonCV: a deep learning toolkit for computer vision, https://gluoncv.mxnet.io/model\_zoo/detection.html#id1

**Thank you**