Objects

person

horse

cow

cat

elephant

giraffe

motorcycle

car

bus

train

truck

boat

traffic light

fire hydrant,

keyboard

mouse

building

sign

table

bus

window

tree

wall

rock

snow

hat

apple

**RetinaNet**

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, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange,

broccoli, carrot, hot dog, pizza, donot, cake, chair, couch, potted plant, bed,

dining table, toilet, tv, laptop, mouse, remote, keyboard, cell phone, microwave,

oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair dryer,

toothbrush.

**Parameters**

75 convolutional layers

convolutional layer with stride 2 is used to downsample the feature map

No fully connected layers and no pooling layers

This structure makes it possible to deal with images with any sizes

number of classes – 80

kernel size is 1 x 1 x 255

input image of size 416 x 416.

YOLO v3 makes prediction at three scales, which are precisely given by downsampling the dimensions of the input image by 32, 16 and 8 respectively.

YOLO v3, in total uses 9 anchor boxes. Three for each scale

K-Means clustering to generate 9 anchors.

YOLO v3 predicts 10x the number of boxes predicted by YOLO v2.

object confidence and class predictions in YOLO v3 are now predicted through logistic regression.

each class score is predicted using logistic regression and a threshold is used to predict multiple labels for an object. Classes with scores higher than this threshold are assigned to the box.

If the IoU between the prediction and the ground truth box is less than 0.5, the prediction is classified as a mislocalisation and marked a false positive.

For other priors with overlap greater than a predefined threshold (default 0.5), they incur no cost.

The function that is used to compute this error is known as Loss Function

Pr(object)\*IoU

Pr(classi | object)

**the width and height of the anchors are (10×13),(16×30),(33×23),(30×61),(62×45),(59× 119),(116 × 90),(156 × 198),(373 × 326).**

**Inside yolo.h5 file**

import h5py

import numpy as np

with h5py.File('drive/My Drive/yolo.h5', 'r') as hdf:

ls= list(hdf.keys())

print(ls)

data = hdf.get('model\_weights')

print(data)

model\_weights = np.array(data)

print(model\_weights)

output

['model\_weights']

<HDF5 group "/model\_weights" (252 members)>

['add\_1' 'add\_10' 'add\_11' 'add\_12' 'add\_13' 'add\_14' 'add\_15' 'add\_16'

'add\_17' 'add\_18' 'add\_19' 'add\_2' 'add\_20' 'add\_21' 'add\_22' 'add\_23'

'add\_3' 'add\_4' 'add\_5' 'add\_6' 'add\_7' 'add\_8' 'add\_9'

'batch\_normalization\_1' 'batch\_normalization\_10' 'batch\_normalization\_11'

'batch\_normalization\_12' 'batch\_normalization\_13'

'batch\_normalization\_14' 'batch\_normalization\_15'

'batch\_normalization\_16' 'batch\_normalization\_17'

'batch\_normalization\_18' 'batch\_normalization\_19' 'batch\_normalization\_2'

'batch\_normalization\_20' 'batch\_normalization\_21'

'batch\_normalization\_22' 'batch\_normalization\_23'

'batch\_normalization\_24' 'batch\_normalization\_25'

'batch\_normalization\_26' 'batch\_normalization\_27'

'batch\_normalization\_28' 'batch\_normalization\_29' 'batch\_normalization\_3'

'batch\_normalization\_30' 'batch\_normalization\_31'

'batch\_normalization\_32' 'batch\_normalization\_33'

'batch\_normalization\_34' 'batch\_normalization\_35'

'batch\_normalization\_36' 'batch\_normalization\_37'

'batch\_normalization\_38' 'batch\_normalization\_39' 'batch\_normalization\_4'

'batch\_normalization\_40' 'batch\_normalization\_41'

'batch\_normalization\_42' 'batch\_normalization\_43'

'batch\_normalization\_44' 'batch\_normalization\_45'

'batch\_normalization\_46' 'batch\_normalization\_47'

'batch\_normalization\_48' 'batch\_normalization\_49' 'batch\_normalization\_5'

'batch\_normalization\_50' 'batch\_normalization\_51'

'batch\_normalization\_52' 'batch\_normalization\_53'

'batch\_normalization\_54' 'batch\_normalization\_55'

'batch\_normalization\_56' 'batch\_normalization\_57'

'batch\_normalization\_58' 'batch\_normalization\_59' 'batch\_normalization\_6'

'batch\_normalization\_60' 'batch\_normalization\_61'

'batch\_normalization\_62' 'batch\_normalization\_63'

'batch\_normalization\_64' 'batch\_normalization\_65'

'batch\_normalization\_66' 'batch\_normalization\_67'

'batch\_normalization\_68' 'batch\_normalization\_69' 'batch\_normalization\_7'

'batch\_normalization\_70' 'batch\_normalization\_71'

'batch\_normalization\_72' 'batch\_normalization\_8' 'batch\_normalization\_9'

'concatenate\_1' 'concatenate\_2' 'conv2d\_1' 'conv2d\_10' 'conv2d\_11'

'conv2d\_12' 'conv2d\_13' 'conv2d\_14' 'conv2d\_15' 'conv2d\_16' 'conv2d\_17'

'conv2d\_18' 'conv2d\_19' 'conv2d\_2' 'conv2d\_20' 'conv2d\_21' 'conv2d\_22'

'conv2d\_23' 'conv2d\_24' 'conv2d\_25' 'conv2d\_26' 'conv2d\_27' 'conv2d\_28'

'conv2d\_29' 'conv2d\_3' 'conv2d\_30' 'conv2d\_31' 'conv2d\_32' 'conv2d\_33'

'conv2d\_34' 'conv2d\_35' 'conv2d\_36' 'conv2d\_37' 'conv2d\_38' 'conv2d\_39'

'conv2d\_4' 'conv2d\_40' 'conv2d\_41' 'conv2d\_42' 'conv2d\_43' 'conv2d\_44'

'conv2d\_45' 'conv2d\_46' 'conv2d\_47' 'conv2d\_48' 'conv2d\_49' 'conv2d\_5'

'conv2d\_50' 'conv2d\_51' 'conv2d\_52' 'conv2d\_53' 'conv2d\_54' 'conv2d\_55'

'conv2d\_56' 'conv2d\_57' 'conv2d\_58' 'conv2d\_59' 'conv2d\_6' 'conv2d\_60'

'conv2d\_61' 'conv2d\_62' 'conv2d\_63' 'conv2d\_64' 'conv2d\_65' 'conv2d\_66'

'conv2d\_67' 'conv2d\_68' 'conv2d\_69' 'conv2d\_7' 'conv2d\_70' 'conv2d\_71'

'conv2d\_72' 'conv2d\_73' 'conv2d\_74' 'conv2d\_75' 'conv2d\_8' 'conv2d\_9'

'input\_1' 'leaky\_re\_lu\_1' 'leaky\_re\_lu\_10' 'leaky\_re\_lu\_11'

'leaky\_re\_lu\_12' 'leaky\_re\_lu\_13' 'leaky\_re\_lu\_14' 'leaky\_re\_lu\_15'

'leaky\_re\_lu\_16' 'leaky\_re\_lu\_17' 'leaky\_re\_lu\_18' 'leaky\_re\_lu\_19'

'leaky\_re\_lu\_2' 'leaky\_re\_lu\_20' 'leaky\_re\_lu\_21' 'leaky\_re\_lu\_22'

'leaky\_re\_lu\_23' 'leaky\_re\_lu\_24' 'leaky\_re\_lu\_25' 'leaky\_re\_lu\_26'

'leaky\_re\_lu\_27' 'leaky\_re\_lu\_28' 'leaky\_re\_lu\_29' 'leaky\_re\_lu\_3'

'leaky\_re\_lu\_30' 'leaky\_re\_lu\_31' 'leaky\_re\_lu\_32' 'leaky\_re\_lu\_33'

'leaky\_re\_lu\_34' 'leaky\_re\_lu\_35' 'leaky\_re\_lu\_36' 'leaky\_re\_lu\_37'

'leaky\_re\_lu\_38' 'leaky\_re\_lu\_39' 'leaky\_re\_lu\_4' 'leaky\_re\_lu\_40'

'leaky\_re\_lu\_41' 'leaky\_re\_lu\_42' 'leaky\_re\_lu\_43' 'leaky\_re\_lu\_44'

'leaky\_re\_lu\_45' 'leaky\_re\_lu\_46' 'leaky\_re\_lu\_47' 'leaky\_re\_lu\_48'

'leaky\_re\_lu\_49' 'leaky\_re\_lu\_5' 'leaky\_re\_lu\_50' 'leaky\_re\_lu\_51'

'leaky\_re\_lu\_52' 'leaky\_re\_lu\_53' 'leaky\_re\_lu\_54' 'leaky\_re\_lu\_55'

'leaky\_re\_lu\_56' 'leaky\_re\_lu\_57' 'leaky\_re\_lu\_58' 'leaky\_re\_lu\_59'

'leaky\_re\_lu\_6' 'leaky\_re\_lu\_60' 'leaky\_re\_lu\_61' 'leaky\_re\_lu\_62'

'leaky\_re\_lu\_63' 'leaky\_re\_lu\_64' 'leaky\_re\_lu\_65' 'leaky\_re\_lu\_66'

'leaky\_re\_lu\_67' 'leaky\_re\_lu\_68' 'leaky\_re\_lu\_69' 'leaky\_re\_lu\_7'

'leaky\_re\_lu\_70' 'leaky\_re\_lu\_71' 'leaky\_re\_lu\_72' 'leaky\_re\_lu\_8'

'leaky\_re\_lu\_9' 'up\_sampling2d\_1' 'up\_sampling2d\_2' 'zero\_padding2d\_1'

'zero\_padding2d\_2' 'zero\_padding2d\_3' 'zero\_padding2d\_4'

'zero\_padding2d\_5']

**Inside retinanet.h5 file**

import h5py

import numpy as np

with h5py.File('drive/My Drive/resnet50\_coco\_best\_v2.0.1.h5', 'r') as hdf:

ls= list(hdf.keys())

print(ls)

data = hdf.get('model\_weights')

print(data)

model\_weights = np.array(data)

print(model\_weights)

Output

['model\_weights']

<HDF5 group "/model\_weights" (215 members)>

['C3\_reduced' 'C4\_reduced' 'C5\_reduced' 'C6\_relu' 'P3' 'P3\_merged' 'P4'

'P4\_merged' 'P4\_upsampled' 'P5' 'P5\_upsampled' 'P6' 'P7' 'anchors\_0'

'anchors\_1' 'anchors\_2' 'anchors\_3' 'anchors\_4' 'bn2a\_branch1'

'bn2a\_branch2a' 'bn2a\_branch2b' 'bn2a\_branch2c' 'bn2b\_branch2a'

'bn2b\_branch2b' 'bn2b\_branch2c' 'bn2c\_branch2a' 'bn2c\_branch2b'

'bn2c\_branch2c' 'bn3a\_branch1' 'bn3a\_branch2a' 'bn3a\_branch2b'

'bn3a\_branch2c' 'bn3b\_branch2a' 'bn3b\_branch2b' 'bn3b\_branch2c'

'bn3c\_branch2a' 'bn3c\_branch2b' 'bn3c\_branch2c' 'bn3d\_branch2a'

'bn3d\_branch2b' 'bn3d\_branch2c' 'bn4a\_branch1' 'bn4a\_branch2a'

'bn4a\_branch2b' 'bn4a\_branch2c' 'bn4b\_branch2a' 'bn4b\_branch2b'

'bn4b\_branch2c' 'bn4c\_branch2a' 'bn4c\_branch2b' 'bn4c\_branch2c'

'bn4d\_branch2a' 'bn4d\_branch2b' 'bn4d\_branch2c' 'bn4e\_branch2a'

'bn4e\_branch2b' 'bn4e\_branch2c' 'bn4f\_branch2a' 'bn4f\_branch2b'

'bn4f\_branch2c' 'bn5a\_branch1' 'bn5a\_branch2a' 'bn5a\_branch2b'

'bn5a\_branch2c' 'bn5b\_branch2a' 'bn5b\_branch2b' 'bn5b\_branch2c'

'bn5c\_branch2a' 'bn5c\_branch2b' 'bn5c\_branch2c' 'bn\_conv1' 'boxes'

'classification' 'classification\_submodel' 'concatenate\_1' 'conv1'

'conv1\_relu' 'input\_1' 'nms' 'padding2a\_branch2b' 'padding2b\_branch2b'

'padding2c\_branch2b' 'padding3a\_branch2b' 'padding3b\_branch2b'

'padding3c\_branch2b' 'padding3d\_branch2b' 'padding4a\_branch2b'

'padding4b\_branch2b' 'padding4c\_branch2b' 'padding4d\_branch2b'

'padding4e\_branch2b' 'padding4f\_branch2b' 'padding5a\_branch2b'

'padding5b\_branch2b' 'padding5c\_branch2b' 'padding\_conv1' 'pool1'

'regression' 'regression\_submodel' 'res2a' 'res2a\_branch1'

'res2a\_branch2a' 'res2a\_branch2a\_relu' 'res2a\_branch2b'

'res2a\_branch2b\_relu' 'res2a\_branch2c' 'res2a\_relu' 'res2b'

'res2b\_branch2a' 'res2b\_branch2a\_relu' 'res2b\_branch2b'

'res2b\_branch2b\_relu' 'res2b\_branch2c' 'res2b\_relu' 'res2c'

'res2c\_branch2a' 'res2c\_branch2a\_relu' 'res2c\_branch2b'

'res2c\_branch2b\_relu' 'res2c\_branch2c' 'res2c\_relu' 'res3a'

'res3a\_branch1' 'res3a\_branch2a' 'res3a\_branch2a\_relu' 'res3a\_branch2b'

'res3a\_branch2b\_relu' 'res3a\_branch2c' 'res3a\_relu' 'res3b'

'res3b\_branch2a' 'res3b\_branch2a\_relu' 'res3b\_branch2b'

'res3b\_branch2b\_relu' 'res3b\_branch2c' 'res3b\_relu' 'res3c'

'res3c\_branch2a' 'res3c\_branch2a\_relu' 'res3c\_branch2b'

'res3c\_branch2b\_relu' 'res3c\_branch2c' 'res3c\_relu' 'res3d'

'res3d\_branch2a' 'res3d\_branch2a\_relu' 'res3d\_branch2b'

'res3d\_branch2b\_relu' 'res3d\_branch2c' 'res3d\_relu' 'res4a'

'res4a\_branch1' 'res4a\_branch2a' 'res4a\_branch2a\_relu' 'res4a\_branch2b'

'res4a\_branch2b\_relu' 'res4a\_branch2c' 'res4a\_relu' 'res4b'

'res4b\_branch2a' 'res4b\_branch2a\_relu' 'res4b\_branch2b'

'res4b\_branch2b\_relu' 'res4b\_branch2c' 'res4b\_relu' 'res4c'

'res4c\_branch2a' 'res4c\_branch2a\_relu' 'res4c\_branch2b'

'res4c\_branch2b\_relu' 'res4c\_branch2c' 'res4c\_relu' 'res4d'

'res4d\_branch2a' 'res4d\_branch2a\_relu' 'res4d\_branch2b'

'res4d\_branch2b\_relu' 'res4d\_branch2c' 'res4d\_relu' 'res4e'

'res4e\_branch2a' 'res4e\_branch2a\_relu' 'res4e\_branch2b'

'res4e\_branch2b\_relu' 'res4e\_branch2c' 'res4e\_relu' 'res4f'

'res4f\_branch2a' 'res4f\_branch2a\_relu' 'res4f\_branch2b'

'res4f\_branch2b\_relu' 'res4f\_branch2c' 'res4f\_relu' 'res5a'

'res5a\_branch1' 'res5a\_branch2a' 'res5a\_branch2a\_relu' 'res5a\_branch2b'

'res5a\_branch2b\_relu' 'res5a\_branch2c' 'res5a\_relu' 'res5b'

'res5b\_branch2a' 'res5b\_branch2a\_relu' 'res5b\_branch2b'

'res5b\_branch2b\_relu' 'res5b\_branch2c' 'res5b\_relu' 'res5c'

'res5c\_branch2a' 'res5c\_branch2a\_relu' 'res5c\_branch2b'

'res5c\_branch2b\_relu' 'res5c\_branch2c' 'res5c\_relu']

Yolov3 Configration file – Main website ( <https://pjreddie.com/darknet/yolo/> )

<https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg>

[net]

# Testing

# batch=1

# subdivisions=1

# Training

batch=64

subdivisions=16

width=608

height=608

channels=3

momentum=0.9

decay=0.0005

angle=0

saturation = 1.5

exposure = 1.5

hue=.1

learning\_rate=0.001

burn\_in=1000

max\_batches = 500200

policy=steps

steps=400000,450000

scales=.1,.1

[convolutional]

batch\_normalize=1

filters=32

size=3

stride=1

pad=1

activation=leaky

# Downsample

[convolutional]

batch\_normalize=1

filters=64

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=32

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=64

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

# Downsample

[convolutional]

batch\_normalize=1

filters=128

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=64

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=64

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

# Downsample

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

# Downsample

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

# Downsample

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=2

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

filters=1024

size=3

stride=1

pad=1

activation=leaky

[shortcut]

from=-3

activation=linear

######################

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=1024

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=1024

activation=leaky

[convolutional]

batch\_normalize=1

filters=512

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=1024

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=255

activation=linear

[yolo]

mask = 6,7,8

anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326

classes=80

num=9

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

[route]

layers = -4

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[upsample]

stride=2

[route]

layers = -1, 61

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=512

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=512

activation=leaky

[convolutional]

batch\_normalize=1

filters=256

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=512

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=255

activation=linear

[yolo]

mask = 3,4,5

anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326

classes=80

num=9

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

[route]

layers = -4

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[upsample]

stride=2

[route]

layers = -1, 36

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=256

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=256

activation=leaky

[convolutional]

batch\_normalize=1

filters=128

size=1

stride=1

pad=1

activation=leaky

[convolutional]

batch\_normalize=1

size=3

stride=1

pad=1

filters=256

activation=leaky

[convolutional]

size=1

stride=1

pad=1

filters=255

activation=linear

[yolo]

mask = 0,1,2

anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326

classes=80

num=9

jitter=.3

ignore\_thresh = .7

truth\_thresh = 1

random=1

Detection evaluation = <http://cocodataset.org/#detection-eval>

<https://github.com/cocodataset/cocoapi/blob/master/PythonAPI/pycocoEvalDemo.ipynb>

IoU measures how much overlap between 2 regions, This measures how good is our prediction in the object detector with the ground truth (the real object boundary).

<https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

<https://heartbeat.fritz.ai/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2-65fe59ac12d>