Table of Contents

[Introduction/Challenge Description 1](#_Toc24721387)

[Proposed solution 2](#_Toc24721388)

[Network Architecture(RetinaNet) 2](#_Toc24721389)

[Feature Pyramid Network 3](#_Toc24721390)

[Focal Loss 3](#_Toc24721391)

[installation 4](#_Toc24721392)

[building dataset 5](#_Toc24721393)

[Anchors parameters 10](#_Toc24721394)

[Some Hyperparameters 11](#_Toc24721395)

[Image Augmentation 12](#_Toc24721396)

[Visualize augmentations 16](#_Toc24721397)

[More Hyperparameters 16](#_Toc24721398)

[Callbacks 17](#_Toc24721399)

[Download pretrained model 17](#_Toc24721400)

[Loading the model 18](#_Toc24721401)

[Training the Model 18](#_Toc24721402)

[Inference/Prediction 18](#_Toc24721403)

[Predict Image 22](#_Toc24721404)

[Total Map Achieved from Training Model is 23% 24](#_Toc24721405)

[Experiment and Results 25](#_Toc24721406)

[Future Work 26](#_Toc24721407)

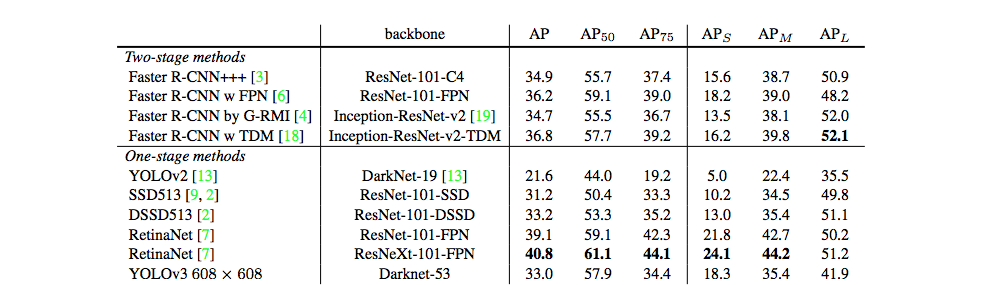
[References 26](#_Toc24721408)

# Introduction/Challenge Description

Development of object detection model to detect a smaller object/s in a given image.

# Proposed solution

I found several popular detectors including: OverFeat (Sermanet et al. 2013), R-CNN (Girshick et al. 2013), Fast R-CNN (Girshick 2015), SSD (Liu et al. 2016), R-FCN (Dai et al. 2016), YOLO (Redmon et al. 2016), Faster R-CNN (Ren et al. 2017) and RetinaNet (Lin, Goyal, et al. 2017). According to the paper, RetinaNet showed both ideal accuracy and speed compared to other detectors for small object detection while still keeping a very simple construct; plus, there is an [opensource implementation](https://github.com/fizyr/keras-retinanet) by Gaiser et al. (2018). Therefore, RetinaNet appears to be an ideal candidate for this project.



Retinanet Results on MS COCO

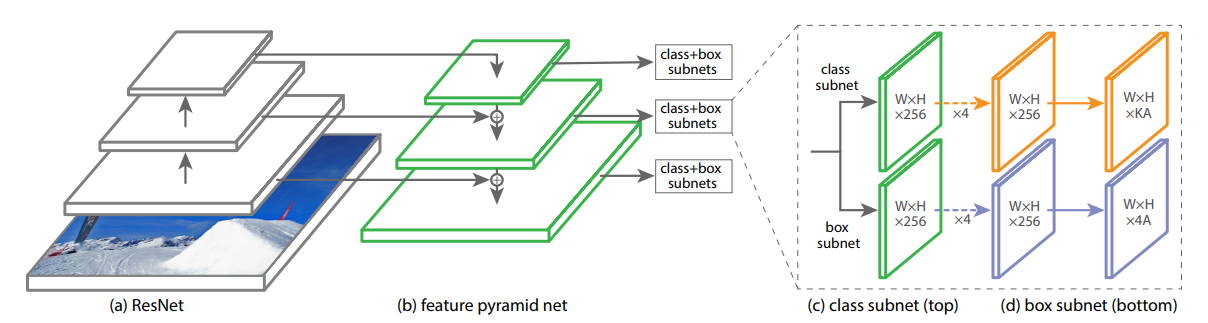
# Network Architecture(RetinaNet)

RetinaNet has been formed by making two improvements over existing single stage object detection models (like YOLO and SSD):

1. [Feature Pyramid Networks for Object Detection](https://arxiv.org/abs/1612.03144)
2. [Focal Loss for Dense Object Detection](https://arxiv.org/abs/1708.02002)

## Feature Pyramid Network

Pyramid networks have been used conventionally to identify objects at different scales. A Feature Pyramid Network (FPN) makes use of the inherent multi-scale pyramidal hierarchy of deep CNNs to create feature pyramids.



The one-stage RetinaNet network architecture uses a Feature Pyramid Network (FPN) backbone on top of a feedforward ResNet architecture (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN while running at faster speeds.

## Focal Loss

Focal Loss is an improvement on cross-entropy loss that helps to reduce the relative loss for well-classified examples and putting more focus on hard, misclassified examples.

The focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.



Focal Loss Function

If you are further interested in the finer details of the model, I’ll suggest reading the original papers and this very helpful and descriptive blog ‘[The intuition behind RetinaNet](https://medium.com/@14prakash/the-intuition-behind-retinanet-eb636755607d)’.

# installation

* pip install imgaug
* pip install numpy scipy h5py
* pip install scikit-learn
* pip install tensorflow-gpu
* pip install opencv-contrib-python
* pip install keras-resnet
* pip install keras==2.1.3
* pip install tqdm

# Install Retinanet

* Download Link :https://github.com/fizyr/keras-retinanet
* cd keras-retinanet
* python setup.py build\_ext --inplace

# Install cudnn and cuda

* https://www.codingforentrepreneurs.com/blog/install-tensorflow-gpu-windows-cuda-cudnn(cudn)

# building dataset

* Download Bosch Small Traffic Lights Dataset <https://hci.iwr.uni-heidelberg.de/node/6132>
* Download contains Images and Yaml file.
* Convert Bosch Small Traffic Lights Dataset Annotations to Pascal VOC Format as image detail is given in Yaml file.
* python BoschTrafficYamlFileToXMLConverter.py train.yaml train\_out\_folder
* python BoschTrafficYamlFileToXMLConverter.py test.yaml test\_out\_folder

|  |
| --- |
| """  This script Converts Yaml annotations to Pascal .xml Files  of the Bosch Small Traffic Lights Dataset.  Example usage:  python bosch\_to\_pascal.py input\_yaml out\_folder  """  import os  import sys  import yaml  from lxml import etree  import os.path  import xml.etree.cElementTree as ET  def write\_xml(savedir, image, imgWidth, imgHeight,  depth=3, pose="Unspecified"):  boxes = image['boxes']  impath = image['path']  imagename = impath.split('/')[-1]  print("imagename&&&&&",imagename)  print("imagpath&&&&&",impath)  currentfolder = savedir.split("\\")[-1]  annotation = ET.Element("annotaion")  ET.SubElement(annotation, 'folder').text = str(currentfolder)  ET.SubElement(annotation, 'filename').text = str(impath)  imagename = imagename.split('.')[0]  size = ET.SubElement(annotation, 'size')  ET.SubElement(size, 'width').text = str(imgWidth)  ET.SubElement(size, 'height').text = str(imgHeight)  ET.SubElement(size, 'depth').text = str(depth)  ET.SubElement(annotation, 'segmented').text = '0'  for box in boxes:  obj = ET.SubElement(annotation, 'object')  ET.SubElement(obj, 'name').text = str(box['label'])  ET.SubElement(obj, 'pose').text = str(pose)  ET.SubElement(obj, 'occluded').text = str(box['occluded'])  ET.SubElement(obj, 'difficult').text = '0'  bbox = ET.SubElement(obj, 'bndbox')  ET.SubElement(bbox, 'xmin').text = str(box['x\_min'])  ET.SubElement(bbox, 'ymin').text = str(box['y\_min'])  ET.SubElement(bbox, 'xmax').text = str(box['x\_max'])  ET.SubElement(bbox, 'ymax').text = str(box['y\_max'])  xml\_str = ET.tostring(annotation)  root = etree.fromstring(xml\_str)  xml\_str = etree.tostring(root, pretty\_print=True)      save\_path = os.path.join(savedir, imagename + ".xml")    print("savepath is:::::::",save\_path)  print("xml string is::::::",xml\_str)  with open(save\_path, 'wb') as temp\_xml:  print("in writing the file")  temp\_xml.write(xml\_str)  if \_\_name\_\_ == '\_\_main\_\_':  if len(sys.argv) < 3:  print(\_\_doc\_\_)  sys.exit(-1)  yaml\_path = sys.argv[1]  out\_dir = sys.argv[2]  print("output directory is:::::",out\_dir)  images = yaml.load(open(yaml\_path, 'rb').read())  for image in images:  write\_xml(out\_dir, image, 1280, 720, depth=3, pose="Unspecified") |

* Perform above steps for Test and Train Data. Executing above code create train and test xml file .
* Next, let’s write a Python script that will read all the image paths and annotations and output the three CSVs that are required during training and evaluating the model.
* train.csv — This file will hold all the annotations for training in the format: <path/to/image>,<xmin>,<ymin>,<xmax>,<ymax>,<label>  
  Each row will represent one bounding box, therefore, one image can be present in multiple rows depending on how many objects have been annotated in that image.
* test.csv — Similar to train.csv in format, this file will hold all the annotations for testing the model.
* classes.csv — A file with all unique class labels in the dataset with index assignments (starting from 0 and ignoring the background)
* For every image, find all the objects and iterate over each one of them. Then, find the bounding box (xmin, ymin, xmax, ymax) and the class label (name) for each object in the annotation. Do a cleanup by truncating any bounding box coordinate that falls outside the boundaries of the image. Also, do a sanity check if, by error, any minimum value is larger than the maximum value and vice-versa. If we find such values, we will ignore these objects and continue to the next one.

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """  Created on Thu Oct 31 23:06:04 2019  @author: Pawan  """  import math  import sys  import os  import xml.etree.ElementTree as ET  import csv  ANNOTATIONS\_FILE = "C:\\Users\\Pawan\\Documents\\ML\\annotations\_train\_modified2.csv"  CLASSES\_FILE = "C:\\Users\\Pawan\\Documents\\ML\\classes\_train\_modified2.csv"  DATASET\_DIR="C:\\Users\\Pawan\\Documents\\ML\\TRAIN\_XML\_FILE"  annotations = []  classes = set([])  for xml\_file in [f for f in os.listdir(DATASET\_DIR) if f.endswith(".xml")]:  tree = ET.parse(os.path.join(DATASET\_DIR, xml\_file))  root = tree.getroot()  file\_name = None  print("after root################")  for elem in root:  if elem.tag == 'filename':  fileSplit=elem.text.split("/")  ModifiedFileName="C:\\Users\\Pawan\\Downloads\\dataset\_train\_rgb\\rgb\\train"+"\\"+fileSplit[3]+"\\"+fileSplit[4]  file\_name = ModifiedFileName  if elem.tag == 'object':  obj\_name = None  coords = []  for subelem in elem:  if subelem.tag == 'name':  obj\_name = subelem.text  if subelem.tag == 'bndbox':  for subsubelem in subelem:    if((str(subsubelem)).find('min')>0):  coords.append(math.floor(float(subsubelem.text)))  else:  coords.append(math.ceil(float(subsubelem.text)))    if coords[0]>=coords[2] or coords[1] >= coords[3] :  continue    if coords[2]<=coords[0] or coords[3]<=coords[1] :  continue    item = [file\_name] + coords + [obj\_name]  annotations.append(item)  classes.add(obj\_name)  with open(ANNOTATIONS\_FILE, 'w') as f:  writer = csv.writer(f)  writer.writerows(annotations)  with open(CLASSES\_FILE, 'w') as f:  for i, line in enumerate(classes):  f.write('{},{}\n'.format(line,i)) |

# Anchors parameters

* Anchor parameters are used to decide how anchor boxes will be generated for the model.
* As we're dealing mostly small boxes with can be highly elongated, we'll change ratios and scales to fit our needs.

|  |
| --- |
| with open('config.ini','w') as f: f.write('[anchor\_parameters]\nsizes = 16 32 64 128 256 512\nstrides = 8 16 32 64 128\nratios = 0.001 0.1 0.442 0.476 0.5 1.0 2.102 2.26 3 4\nscales =0.1 0.2 0.3 0.4 0.498 0.506 0.625 0.639 1 1.2 1.6 1.8\n') |

# Some Hyperparameters

* We will rescale our images to 672x672 for better precision

|  |
| --- |
| b = backbone('resnet50')  class args:  batch\_size =2  #64  config = read\_config\_file('config1111.ini')  #config=True  random\_transform = True # Image augmentation  annotations = "C:\\Users\\Pawan\\Documents\\ML\\annotations\_train\_modified2.csv"  val\_annotations = "C:\\Users\\Pawan\\Documents\\ML\\annotations\_test\_modified2.csv"  no\_resize=False  classes = "C:\\Users\\Pawan\\Documents\\ML\\classes\_train\_modified2.csv"  image\_min\_side = 672  image\_max\_side = 672  dataset\_type = 'csv'  tensorboard\_dir = 'C:\\Users\\Pawan\\Documents\\Tensorboard'  evaluation = True  snapshots = True  snapshot\_path = "C:\\Users\\Pawan\\Documents\\ML\\snapshots12"  backbone = 'resnet50'  #epochs = 70  epochs = 70  steps = 10755//(batch\_size)  weighted\_average = True  gpu=0  resize=True      train\_gen,valid\_gen = create\_generators(args,b.preprocess\_image) |

# Image Augmentation

* In addition to augmentations already done by keras-retinanet [here](https://github.com/fizyr/keras-retinanet/blob/master/keras_retinanet/bin/train.py#L227) , we'll use a package called imgaug to further augment the data.

|  |
| --- |
| sometimes = lambda aug: iaa.Sometimes(0.5, aug)  # Define our sequence of augmentation steps that will be applied to every image.  seq = iaa.Sequential(  [  #  # Execute 1 to 9 of the following (less important) augmenters per  # image. Don't execute all of them, as that would often be way too  # strong.  #  iaa.SomeOf((1, 9),  [  # Blur each image with varying strength using  # gaussian blur (sigma between 0 and .5),  # average/uniform blur (kernel size 1x1)  # median blur (kernel size 1x1).  iaa.OneOf([  iaa.GaussianBlur((0,0.5)),  iaa.AverageBlur(k=(1)),  iaa.MedianBlur(k=(1)),  ]),  # Sharpen each image, overlay the result with the original  # image using an alpha between 0 (no sharpening) and 1  # (full sharpening effect).  iaa.Sharpen(alpha=(0, 0.25), lightness=(0.75, 1.5)),  # Add gaussian noise to some images.  # In 50% of these cases, the noise is randomly sampled per  # channel and pixel.  # In the other 50% of all cases it is sampled once per  # pixel (i.e. brightness change).  iaa.AdditiveGaussianNoise(  loc=0, scale=(0.0, 0.01\*255), per\_channel=0.5  ),  # Either drop randomly 1 to 10% of all pixels (i.e. set  # them to black) or drop them on an image with 2-5% percent  # of the original size, leading to large dropped  # rectangles.  iaa.OneOf([  iaa.Dropout((0.01, 0.1), per\_channel=0.5),  iaa.CoarseDropout(  (0.03, 0.15), size\_percent=(0.02, 0.05),  per\_channel=0.2  ),  ]),  # Add a value of -5 to 5 to each pixel.  iaa.Add((-5, 5), per\_channel=0.5),  # Change brightness of images (85-115% of original value).  iaa.Multiply((0.85, 1.15), per\_channel=0.5),  # Improve or worsen the contrast of images.  iaa.ContrastNormalization((0.75, 1.25), per\_channel=0.5),  # Convert each image to grayscale and then overlay the  # result with the original with random alpha. I.e. remove  # colors with varying strengths.  iaa.Grayscale(alpha=(0.0, 0.25)),  # In some images distort local areas with varying strength.  sometimes(iaa.PiecewiseAffine(scale=(0.001, 0.01)))  ],  # do all of the above augmentations in random order  random\_order=True  )  ],  # do all of the above augmentations in random order  random\_order=True  )  #######################################################################################  def augment\_train\_gen(train\_gen,visualize=False):  '''  Creates a generator using another generator with applied image augmentation.  Args  train\_gen : keras-retinanet generator object.  visualize : Boolean; False will convert bounding boxes to their anchor box targets for the model.  '''  imgs = []  boxes = []  targets = []  size = train\_gen.size()  idx = 0  while True:  while len(imgs) < args.batch\_size:  image = train\_gen.load\_image(idx % size)  annotations = train\_gen.load\_annotations(idx % size)  image,annotations = train\_gen.random\_transform\_group\_entry(image,annotations)  imgs.append(image)  boxes.append(annotations['bboxes'])  targets.append(annotations)  idx += 1  if visualize:  imgs = seq.augment\_images(imgs)  imgs = np.array(imgs)  boxes = np.array(boxes)  yield imgs,boxes  else:  imgs = seq.augment\_images(imgs)  imgs,targets = train\_gen.preprocess\_group(imgs,targets)  imgs = train\_gen.compute\_inputs(imgs)  targets = train\_gen.compute\_targets(imgs,targets)  imgs = np.array(imgs)  yield imgs,targets  imgs = []  boxes = []  targets = [] |

# Visualize augmentations

|  |
| --- |
| i = 0  for imgs,boxes in augment\_train\_gen(train\_gen,visualize=True):  if i > skip\_batches:  fig=plt.figure(figsize=(24,96))  columns = 2  rows = 8  for i in range(1, columns\*rows + 1):  draw\_boxes(imgs[i], boxes[i], (0, 255, 0), thickness=1)  fig.add\_subplot(rows, columns, i)  plt.imshow(cv2.cvtColor(imgs[i],cv2.COLOR\_BGR2RGB))  plt.show()    else:  i += 1 |

# More Hyperparameters

* we'll use learning rate of 0.001 and freeze weights for the backbone

|  |
| --- |
| model, training\_model, prediction\_model = create\_models(  backbone\_retinanet=b.retinanet,  num\_classes=train\_gen.num\_classes(),  weights=None,  multi\_gpu=True,  freeze\_backbone=True,  lr=1e-3,  config=args.config  ) |

# Callbacks

|  |
| --- |
| callbacks = create\_callbacks(  model,  training\_model,  prediction\_model,  valid\_gen,  args,  ) |

# Download pretrained model

* We download a pretrained model on COCO dataset and load it's weights
* Download Link: <https://github.com/fizyr/keras-retinanet/releases/download/0.5.1/resnet50_coco_best_v2.1.0.h5>

# Loading the model

|  |
| --- |
| training\_model.load\_weights('C:\\Users\\Pawan\\Documents\\ML\\snapshots11\\resnet50\_csv\_01.h5',  skip\_mismatch=True,by\_name=True) |

# Training the Model

|  |
| --- |
| training\_model.fit\_generator(generator=augment\_train\_gen(train\_gen),  steps\_per\_epoch=args.steps,  epochs=args.epochs,  verbose=1,  callbacks=callbacks,) |

# Inference/Prediction

|  |
| --- |
| import os  from os import listdir, walk  from os.path import join  import numpy as np  import keras  import math  import tensorflow as tf  from keras\_retinanet.utils.visualization import draw\_boxes  from sklearn.model\_selection import train\_test\_split  from imgaug import augmenters as iaa  import matplotlib.pyplot as plt  from keras\_retinanet.utils.gpu import setup\_gpu  from keras\_retinanet.utils.image import read\_image\_bgr, preprocess\_image, resize\_image  from tqdm import tqdm  from keras\_retinanet.bin.train import create\_generators,create\_models,create\_callbacks  from keras\_retinanet.models import backbone,load\_model,convert\_model  from keras\_retinanet.utils.config import read\_config\_file,parse\_anchor\_parameters  from keras\_retinanet.utils.visualization import draw\_boxes  from sklearn.model\_selection import train\_test\_split  from imgaug import augmenters as iaa  import matplotlib.pyplot as plt  from keras\_retinanet.utils.gpu import setup\_gpu  import tensorflow.compat.v1 as tf  tf.disable\_v2\_behavior()  config = tf.ConfigProto()  config.gpu\_options.allow\_growth = True  sess = tf.Session(config=config)  b = backbone('resnet50')  class args:  batch\_size =4  config = read\_config\_file('config.ini')  random\_transform = True # Image augmentation  annotations = "C:\\Users\\Pawan\\Documents\\ML\\annotations\_train\_modified2.csv"  val\_annotations = "C:\\Users\\Pawan\\Documents\\ML\\annotations\_test\_modified2.csv"  no\_resize=False  classes = "C:\\Users\\Pawan\\Documents\\ML\\classes\_train\_modified2.csv"  image\_min\_side = 672  image\_max\_side = 672  dataset\_type = 'csv'  tensorboard\_dir = 'C:\\Users\\Pawan\\Documents\\Tensorboard'  evaluation = True  snapshots = True  snapshot\_path = "C:\\Users\\Pawan\\Documents\\ML\\snapshots12"  backbone = 'resnet50'  epochs = 100  steps = 10755//(batch\_size)  gpu=0  resize=True    train\_gen,valid\_gen = create\_generators(args,b.preprocess\_image)  model, training\_model, prediction\_model = create\_models(  backbone\_retinanet=b.retinanet,  num\_classes=train\_gen.num\_classes(),  weights=None,  multi\_gpu=True,  freeze\_backbone=True,  lr=1e-9,  config=args.config  )    training\_model.load\_weights("C:\\Users\\Pawan\\Documents\\ML\\snapshots12\\resnet50\_csv\_07.h5")  infer\_model = convert\_model(training\_model,anchor\_params=parse\_anchor\_parameters(read\_config\_file('C:\\Users\\Pawan\\Documents\\config.ini')))  def test\_gen(image\_ids, bs = 2, size=672,test = True):  imgs = []  scale = None  idx = 0  if test:  path = 'C:\\Users\\Pawan\\Downloads\\dataset\_test\_rgb\\rgb\\test\\'  else:  path = 'C:\\Users\\Pawan\\Downloads\\dataset\_test\_rgb\\rgb\\test\\'    while idx < len(image\_ids):  if len(imgs) < bs:  imgs.append(resize\_image(preprocess\_image(read\_image\_bgr(path + image\_ids[idx] + '.png')),min\_side=size,max\_side=size)[0])  if scale is None:  scale = resize\_image(preprocess\_image(read\_image\_bgr(path + image\_ids[idx] + '.png')),min\_side=size,max\_side=size)[1]  idx += 1  else:  yield np.array(imgs),scale  imgs = []      if len(imgs) > 0:  yield np.array(imgs),scale  print("###########################################################################################")  \_,\_,image\_ids = next(walk('C:\\Users\\Pawan\\Downloads\\dataset\_test\_rgb\\rgb\\test\\'))  image\_ids = [i[:-4] for i in image\_ids]  image\_ids = sorted(image\_ids)  #print("image\_ids",image\_ids)  iter\_num = 0  test\_bs = 2  for imgs,scale in tqdm(test\_gen(image\_ids,bs=test\_bs),total=math.ceil(len(image\_ids)/test\_bs)):  boxes, scores, labels = infer\_model.predict\_on\_batch(imgs)  boxes /= scale  for img\_num in range(len(imgs)):  with open('C:\\Users\\Pawan\\Music\\Paku5\\' + image\_ids[(iter\_num\*test\_bs) + img\_num] + '.txt', 'w') as f:  for box, score, label in zip(boxes[img\_num], scores[img\_num], labels[img\_num]):  # scores are sorted so we can break  if score < 0:  break  f.write(f'{label + 1} {score} {int((box[1]))} {int((box[0]))} {int((box[3]))} {int((box[2]))} \n')  iter\_num += 1 |

# Predict Image

|  |
| --- |
| from keras\_retinanet.utils.visualization import draw\_box, draw\_caption  from keras\_retinanet.utils.colors import label\_color  from keras\_retinanet.utils.image import read\_image\_bgr, preprocess\_image, resize\_image  import cv2  from os import walk  import matplotlib.pyplot as plt  \_,\_,image\_ids = next(walk('C:\\Users\\Pawan\\Downloads\\dataset\_test\_rgb\\rgb\\test\\'))  image\_ids = [i[:-4] for i in image\_ids]  image\_ids = sorted(image\_ids)  image\_ids=["28164"]  idx = 0  image\_id = 1  score\_thres = 0.1  for id in image\_ids:  # load image  #idx += 1  #if idx == image\_id:  image = read\_image\_bgr('C:\\Users\\Pawan\\Downloads\\dataset\_test\_rgb\\rgb\\test\\' + id + '.png')  # copy to draw on  draw = image.copy()  draw = cv2.cvtColor(draw, cv2.COLOR\_BGR2RGB)  # process image  boxes = [list(map(int,(line.split()[3],line.split()[2],line.split()[5],line.split()[4]))) for line in open('C:\\Users\\Pawan\\Music\\Paku5\\' + id + '.txt','r').readlines()]  scores = [float(line.split()[1]) for line in open('C:\\Users\\Pawan\\Music\\Paku5\\' + id + '.txt','r').readlines()]  labels = [int(line.split()[0]) - 1 for line in open('C:\\Users\\Pawan\\Music\\Paku5\\' + id + '.txt','r').readlines()]  print("scores is :::::",scores)  for box, score, label in zip(boxes, scores, labels):  if score < score\_thres:  break  color = label\_color(label)  draw\_box(draw, box, color=color,thickness=1)  caption = "{:.3f}".format(score)  draw\_caption(draw, box, caption)  plt.figure(figsize=(15, 15))  plt.axis('off')  plt.imshow(draw)  plt.show()  break |

# Total Map Achieved from Training Model is 23%

Running network: 100% (7147 of 7147) |###| Elapsed Time: 0:12:15 Time: 0:12:15

Parsing annotations: 100% (7147 of 7147) || Elapsed Time: 0:00:00 Time: 0:00:00

154 instances of class Yellow with average precision: 0.2792

7569 instances of class Green with average precision: 0.2607

0 instances of class RedStraightLeft with average precision: 0.0000

5321 instances of class Red with average precision: 0.1849

442 instances of class off with average precision: 0.0001

0 instances of class RedStraight with average precision: 0.0000

0 instances of class GreenLeft with average precision: 0.0000

0 instances of class GreenStraightLeft with average precision: 0.0000

0 instances of class RedLeft with average precision: 0.0000

0 instances of class GreenStraight with average precision: 0.0000

0 instances of class GreenStraightRight with average precision: 0.0000

0 instances of class GreenRight with average precision: 0.0000

0 instances of class RedRight with average precision: 0.0000

**mAP: 0.2225**

# Experiment and Results

|  |
| --- |
| scores is ::::: [0.571634829044342, 0.5125510692596436, 0.2827792763710022, 0.27717673778533936, 0.20916566252708435, 0.20307132601737976, 0.14584004878997803, 0.13446566462516785, 0.1333320140838623, 0.12682181596755981, 0.12143006920814514, 0.11091485619544983, 0.10256195068359375, 0.10223910212516785, 0.10082471370697021, 0.10029655694961548, 0.09380248188972473, 0.09279590845108032, 0.08195269107818604, 0.07910144329071045, 0.07907557487487793, 0.07834625244140625, 0.07830506563186646, 0.07813361287117004, 0.0770319402217865, 0.07637962698936462, 0.0732857882976532, 0.07205429673194885, 0.07170355319976807, 0.07074078917503357, 0.0706489086151123, 0.0674208402633667, 0.06622764468193054, 0.0644054114818573, 0.06420519948005676, 0.06354096531867981, 0.0634050965309143, 0.06336888670921326, 0.06277352571487427, 0.06230911612510681, 0.06214413046836853, 0.06147173047065735, 0.06101241707801819, 0.06061992049217224, 0.060408711433410645, 0.06028279662132263, 0.060002148151397705, 0.059241026639938354, 0.05922093987464905, 0.05859461426734924, 0.05789363384246826, 0.05667439103126526, 0.05634751915931702, 0.05634298920631409, 0.056228816509246826, 0.05550459027290344, 0.05545032024383545, 0.054631978273391724, 0.05440878868103027, 0.054337382316589355, 0.05411452054977417, 0.0536685585975647, 0.053560495376586914, 0.05317610502243042, 0.05317121744155884, 0.052974700927734375, 0.05244642496109009, 0.05169004201889038, 0.05163013935089111, 0.051077574491500854, 0.050680309534072876, 0.050320059061050415, 0.05017542839050293] |



# Future Work

To Improve MAP

# References

* <https://github.com/fizyr/keras-retinanet>
* <https://github.com/kunwar31/ESRI_Object_Detection>
* <https://medium.com/@14prakash/the-intuition-behind-retinanet-eb636755607d>
* <https://blog.zenggyu.com/en/post/2018-12-05/retinanet-explained-and-demystified/>