Proposal: Detect objects like bicycle/people in image

1. Domain Background

Object detection is an important computer vision task that allows a model to classify a label to the identified object and localize it in the given image. I am motivated by the idea that a camera based sensing device can go through the visuals of our physical surroundings and interpret the world like humans do. This has a major impact on how we as society interpret objects, resources and interact with them.

I would like to understand how vision sensing can process the vision signals and derive useful information like label, texture, orientation etc. from it. **I like outdoors and specially bicycling and would like to understand the detect objects and bicyclists on streets.**

Wikipedia describes this task as "***Object detection*** *is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance*"

A seminal work in this domain is named "Viola–Jones object detection framework" which was published by Viola, P. and Jones, M. in the proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition

Some of the recent works focus on Neural Network and Deep Learning based approaches. It is well described in a review by Rohith Gandhi in a towardsdatascience.com blog titled "R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms"

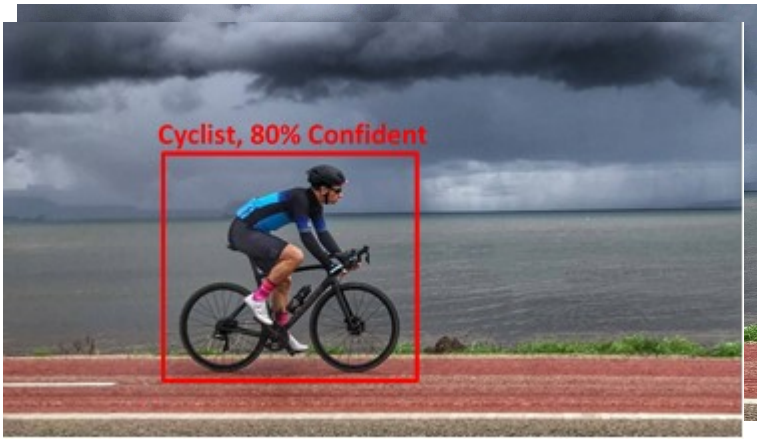
We can use one of these algorithms based on their benchmark to propose a solution.

2. Problem Statement

Given an outdoor image frame, detect bicyclists and all the other objects in view.

- Detection of bicyclist should localize its position with respect to the image frame
- Draw a bounding box in the output image frame with a label classification confidence score.

We measure the confidence score in percentage for classification



3. Solution Statement

To detect bicyclist in an image we use a computer vision task called object detection. The solution code should be able to run as a python inference file on an EC2 instance or can be called from a lambda function as an endpoint. We will use the EC2 instance approach and output the image file with labels.

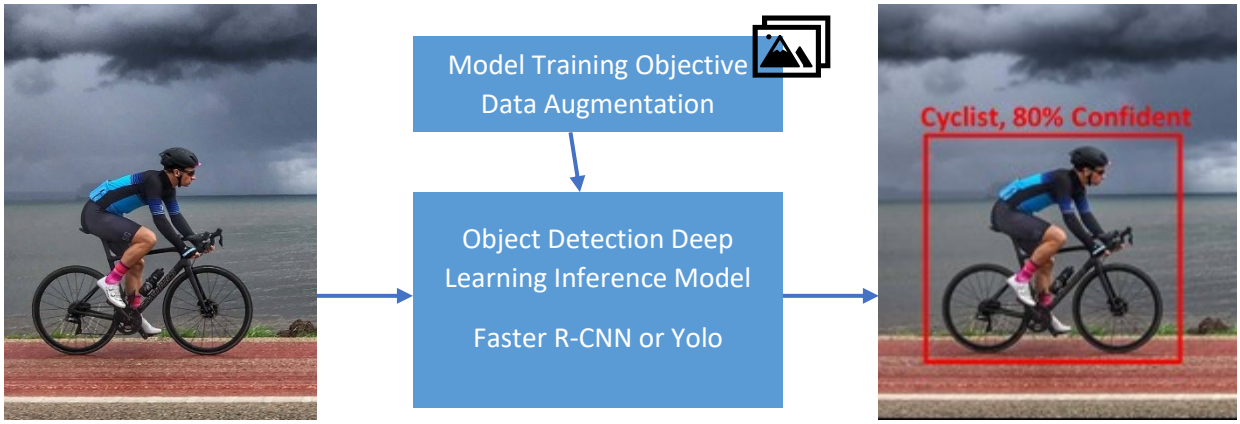


Figure: Input image on the left and output image with object detection results on the right

- **Step 1**: We will use multiple training images with object position annotation to architect a deep learning object detection model
- **Step 2**: We train the deep learning model with augmented dataset with objective that minimizes the loss between predicted bounding box label and actual ground truth bounding box label
- **Step 3**: We select the model that optimizes the loss function
- **Step 4**: We setup the model for inference and demonstrate the output results

Intend to use AWS Sagemaker JupyterLab instance to develop this solution and attempt to deploy this as an endpoint.

4. Datasets & Inputs

We will use a sampled version of the MS COCO dataset (https://cocodataset.org) since our storage and compute needs are to be kept low.

What is COCO? COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features. However, we will use the object instance detection from its annotated 1.5 million object instances.

Figure: Typical objects annotated with bounding boxes in the MS COCO dataset (includes bicycles)

We will describe the annotated dataset now

The data itself has various categories in annotation. Let us look at one of the images and annotation sections of the JSON files that are made available with the image dataset. The dataset is split in 70:20:10 for train, eval and test sets.


```
"image": [{
    "id": 1342334,
    "width": 640,
    "height": 640,
    "file_name: "ford-t.jpg",
    "license": 1,
    "date_captured": "2022-02-01  15:13"
}]
```

```
"annotations": [{
    "segmentation": [[34, 55, 10, 71, 76, 23, 98, 43, 11, 8]],
    "area": 600.4,
    "iscrowd": 1,
    "Image_id:" 122214,
    "bbox": [473.05, 395.45, 38.65, 28.92],
    "category_id": 15,
    "id": 934
}]
```


Figure: JSON format for raw image metadata and corresponding annotations with label & position

Our deep learning model will read training data files with these annotation labels and bounding box during the training phase and then later in the evaluation phase we understand the models overfitness to the data and do model selection.

5. Benchmark Model

Current object detectors can be divided into two categories: Networks separating the tasks of determining the location of objects and their classification, where Faster R-CNN is one of the most famous ones, and networks which predict bounding boxes and class scores at once, with the YOLO and SSD networks being famous architectures.

We would like to approach by using a SSD (single shot detection) or Yolo based approach to detect objects in the given image.

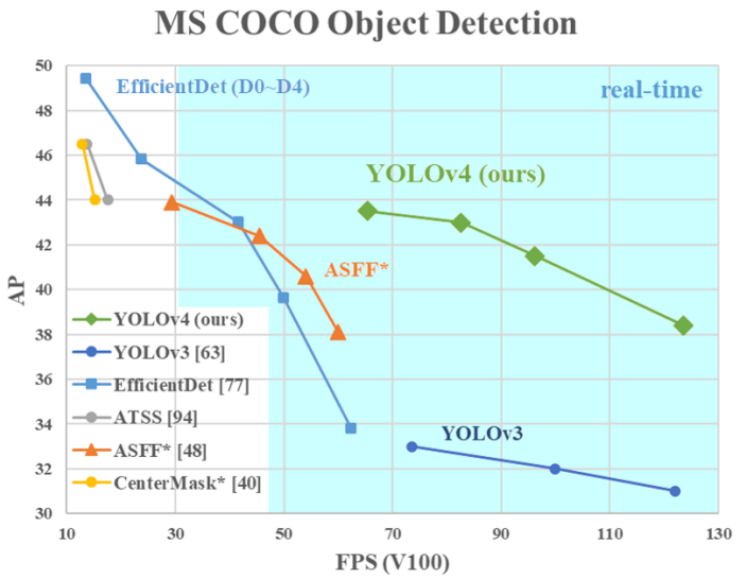


Figure: Model Comparison (YoloV4 - https://arxiv.org/pdf/2004.10934.pdf)

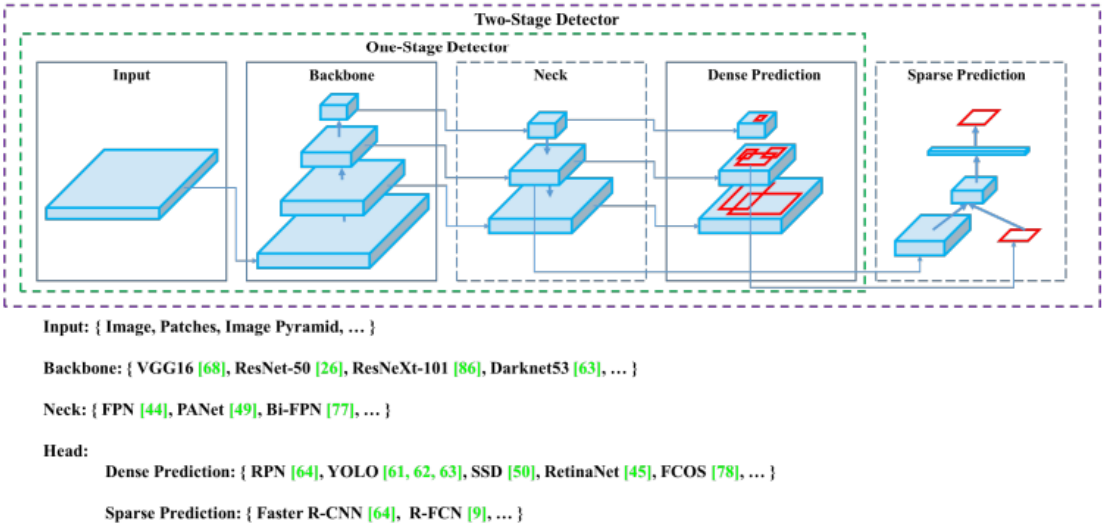


Figure: Model Comparison (YoloV4 - https://arxiv.org/pdf/2004.10934.pdf)

Details on how typical object detectors are setup is mentioned in the figure above and for a small model size and low latency we can choose a Yolo based approach to detect bicyclists in the image.

6. Evalutation Metrics

Detection Evaluation *metrics* used by COCO as per their web page is linked here (https://cocodataset.org/#detection-eval)

We will be using the Average Precision metric to specify the benchmark or characterizing the performance of the object detector model on COCO.

```
Average Precision (AP):
  AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
  AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
  AP^IoU=.75          % AP at IoU=.75 (strict metric)
```

1. AP is averaged over all categories. Traditionally, this is called "mean average precision" (mAP). No distinction between AP and mAP (and likewise AR and mAR) and assume the difference is clear from context.
2. AP (averaged across all 10 IoU thresholds and all 80 categories) will determine the best score.
3. AP is averaged over multiple Intersection over Union (IoU) values.

We can use MS COCO codebase (https://github.com/cocodataset/cocoapi) for loading images and evaluating based on this dataset.

7. Theoretical Workflow

There are three major steps in the DNN based object detection. We describe the workflow in-order of steps to be performed:

a. **Dataset Pre-processing and Model architecture**
   - Dataset setup with images sampled for annotated bicycle labels
   - Split of training/evaluation/test dataset for selected samples. (Typically, 80:10:10). We will use this for object detection model training with the training and evaluation dataset.
   - Common object detectors have the following component (shown in benchmark model figure 2 and in Yolov4 paper)
     - Input image augmented or sized to 640x480
     - Backbone like Resnet 18 or larger
     - Prediction Neck/Head focused on objectives like regression, classification for loss based optimization
     - As per Yolo we have three partial categories losses: confidence/log loss, location/regression loss, category/class loss

b. **Yolo v4 (One of the yolo variants) model training objective**
   - Theoretically described in terms of a loss function as below

$$\mathcal{L}(\hat{z}, z) = \mathcal{L}_{CIoU} - \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} \left[\hat{C}_i \log (C_i) + \left(1 - \hat{C}_i\right) \log (1 - C_i) \right] - \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{\text{noobj}} \left[\hat{C}_i \log (C_i) + \left(1 - \hat{C}_i\right) \log (1 - C_i) \right] - \sum_{i=0}^{S^2} I_{ij}^{obj} \sum_{c \in \text{ classes}} [\hat{p}_i(c) \log (p_i(c)) + (1 - \hat{p}_i(c)) \log (1 - p_i(c))]$$

   - **CIoU Loss** instead of MSE for the regression terms (x,y,w,h). CIoU stands for Complete Intersection over Union, and is not so far from the MSE loss. It proposes to compare width and height a bit more interestingly (consistency between aspect ratios), but it keeps the MSE for the comparison between bounding box centers. This is the first term in the equation above.
   - **Box-per-cell level prediction** instead of cell level prediction for the class probabilities, so a slightly different penalization for the classification terms. where I^{obj}_i denotes if object appears in cell ***i*** and I^{obj}_{ij} denotes that the j^{th} bounding box predictor in cell ***i*** is "responsible" for that prediction
   - **Binary cross entropy** for the objectness and classification scores across all classes. This is the last term in the loss function expansion.

c. **Model Inference**
   - The DNN model is set to eval mode and weights are frozen

- Given a new image, the model will use the model weights to output a array of the size of class identifier and (x,y,w,h) attributes for multiple detected labels.
- We use this information to overlay bounding box on the image using OpenCV and present the results.

In general, we will use a PyTorch based object detector model from open source projects. This will help provide a verifiable evaluation on a standard CPU/GPU training machine.