CSM-392:
UG PROJECT

Topic : Object Detection and Localization using YOLO v3

Submitted by:
Name: Aditya Kulkarni
Roll no: 20124054
Branch: Mathematics and Computing
Email ID: aditya.kulkarni.cd.mat20@iitbhu.ac.in

Table of Contents:

Abstract

- The goal of this project was to build an object detection system that uses YOLOv3 to detect and localize objects within a bounding box.
- The major area of study was Convolutional Neural Networks (CNNs).
- The study involved the readings of various types of CNNs, different architectures comprising CNNs, their working and the intuition behind it.

Procedure

Planning

- After formulating the work plan back in January 2023, I set up a reasonable schedule to work on the project regularly to complement my exploration of the field of computer vision.
- The link to the plan can be found out here: UG: Project Layout.
- This plan consisted of studying and researching along with coding and testing.
- The work plan had been duly submitted to **Prof SK Pandey**, and this project has been done under his mentorship and guidance.

Convolutional Neural Networks

- I started off by studying Convolutional Neural Networks with the help of the famous
- Deep Learning Specialization on Coursera.
- There I learned about the need and functioning of CNNs and how they fulfilled problems faced by the traditional Multi Layer Perceptrons (MLPs).

- CNNs are typically grids of parameters that traverse the image in a set manner and essentially capture the information available in the cell it traverses, i.e. say we have a 64x64 image and a 3x3 convolution layer (a 3x3 grid of parameters).
- It starts traversing from the upper left corner of the image and moves along each 3x3 portion of the image, performing matrix scalar product at each step and finally producing a smaller output containing information regarding the data captured by the CNN.

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

- Convolutional layer
- Pooling layer
- Fully-connected (FC) layer

Convolutional Layer

The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs. It requires a few components, which are input data, a filter, and a feature map. Let's assume that the input will be a color image, which is made up of a matrix of pixels in 3D. This means that the input will have three dimensions—a height, width, and depth—corresponding to RGB in an image. We also have a feature detector, also known as a kernel or a filter, which will move across the receptive fields of the image, checking if the feature is present.
This process is known as a convolution.

Pooling Layer

Pooling layers, also known as downsampling, conduct dimensionality reduction, reducing the number of parameters in the input. Similar to the convolutional layer, the pooling operation sweeps a filter across the entire

input, but the difference is that this filter does not have any weights. Instead, the kernel applies an aggregation function to the values within the receptive field, populating the output array. There are two main types of pooling:

Max pooling: As the filter moves across the input, it selects the pixel with the maximum value to send to the output array. As an aside, this approach tends to be used more often compared to average pooling.

Average pooling: As the filter moves across the input, it calculates the average value within the receptive field to send to the output array.
While a lot of information is lost in the pooling layer, it also has a number of benefits to the CNN. They help to reduce complexity, improve efficiency, and limit risk of overfitting.

Fully Connected Layers

The name of the full-connected layer aptly describes itself. As mentioned earlier, the pixel values of the input image are not directly connected to the output layer in partially connected layers. However, in the fully-connected layer, each node in the output layer connects directly to a node in the previous layer.

This layer performs the task of classification based on the features extracted through the previous layers and their different filters. While convolutional and pooling layers tend to use ReLu functions, FC layers usually leverage a softmax activation function to classify inputs appropriately, producing a probability from 0 to 1.

Data Augmentation

- The problem of lack of data can be tackled by data augmentation, i.e. generating new data from already existing one.
- This can be done by tuning by colour of the image, flipping it horizontally and vertically (given it doesn't change the identity of the image), crop it, shift it, tilt etc.
- There are numerous ways to do so, and hence the problem with lack of data is resolved.

Object Detection and YOLO

Onto the final step of the project, object detection involves not only detecting what object is in an image, but also where it is located. This is done with the help of bounding boxes that encompass the object in the output image. The training for this is slightly different from that of an object classifier. While an object classifier outputs one of the various possible classes of the object after running a CNN model over the image, an object detector does the same in addition to finding the coordinates of the center of the bounding box and its height and width. Thus, the output in this case is larger than the one for an object classifier since the location of the object detected needs to be stored too.

I read the research papers of some of the object detectors such as U-Net and EfficientDet.

YOLOv3 is one such object detector that passes over the image once, and fine-tunes preset bounding boxes (also known as anchor boxes) according

to the locations of the objects it detects as the model passes over the image. This way, it is able to detect the object in one pass and is hence rightfully named, “You Only Look Once”.

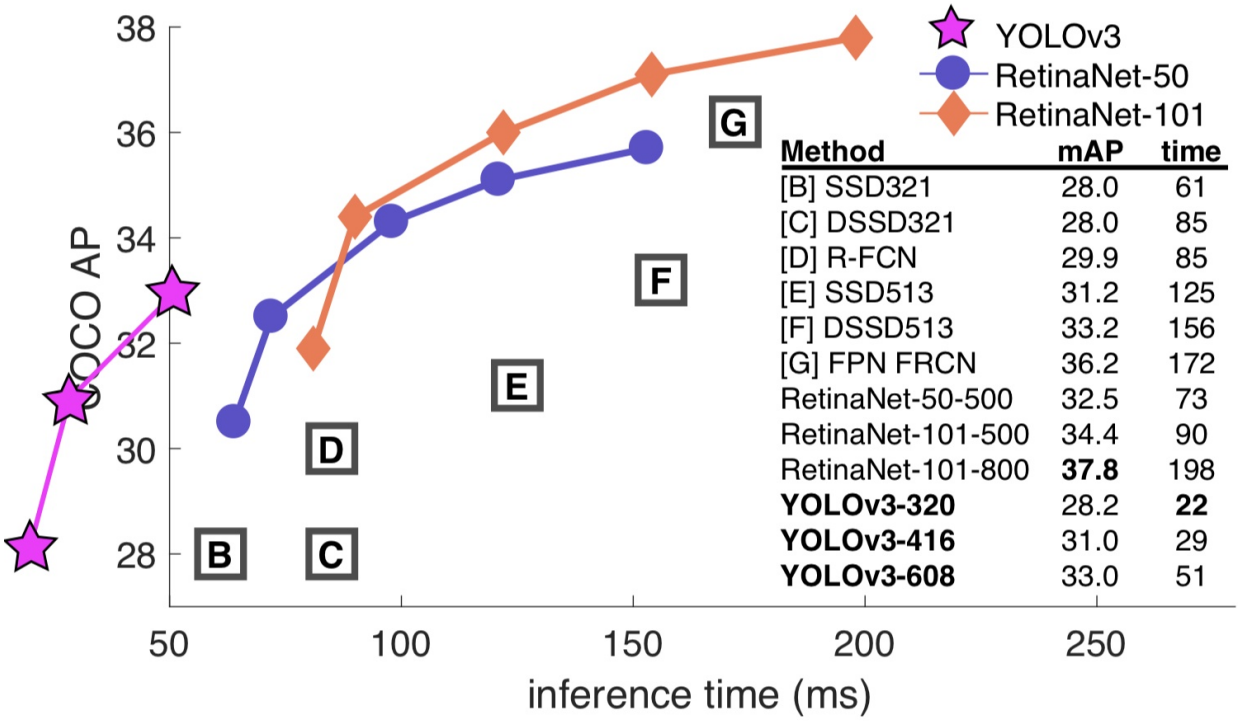
I developed a localized and personalized version of YOLOv3 myself by writing the code of the model myself and then applying the pre-trained weights instead of randomized ones.

Finally, I ran the model over a video and received satisfactory results.

Introduction to YOLO Algorithm

In 2015, Redmon J et al. Proposed the YOLO network, which is characterized by combining the candidate box generation and classification regression into a single step.

In April 2018, the author released the third version of YOLOv3.The mAP-50 on the COCO dataset increased from 44.0% of YOLOv2 to 57.9%. Compared with RetinaNet with 61.1% mAP, RetinaNet by default has an input size of 500x500. In this case, the detection speed is about 98 ms/frame, while YOLOv3 has 29 ms/frame when the input size is 416x416.

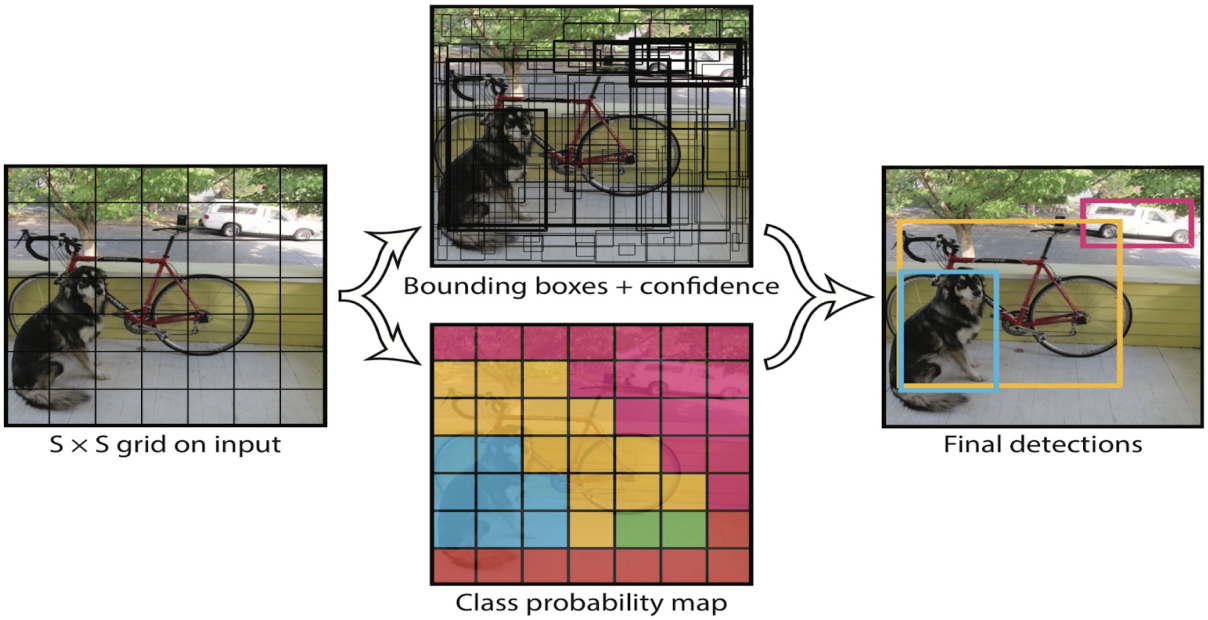


Source: [Original YOLO v3 Paper]

Hence, we see that YOLO v3 has achieved a great accuracy at the premise of maintaining speed.

YOLO v3 Idea

The object detection problem is a regression problem in the YOLO algorithm and it divides the image into an S × S grid. If the center of a target falls into a grid, the grid is responsible for detecting the target.



Each grid will output a bounding box, confidence, and class probability map. Among them:

- The bounding box contains four values: x, y, w, h, (x, y) represents the center of the box. (W, h) defines the width and height of the box.
- Confidence indicates the probability of containing objects in this prediction box, which is the IoU value between the prediction box and the actual box.
- The class probability indicates the class probability of the object, and the YOLOv3 uses a two-class method.

YOLO v3 Architecture

YOLO (You Only Look Once) applies a single forward pass neural network to the whole image and predicts the bounding boxes and their class probabilities as well. This technique makes YOLO quite fast without losing a lot of accuracies.

YOLOv3 has 53 convolutional layers called Darknet-53 is shown in the bellow image. The figure is mainly composed of Convolutional and Residual structures. It should be noted that the last three layers Avgpool, Connected, and softmax layer, are used for classification training on the Imagenet dataset. When using the Darknet-53 layer to extract features from the picture, these three layers are not used.

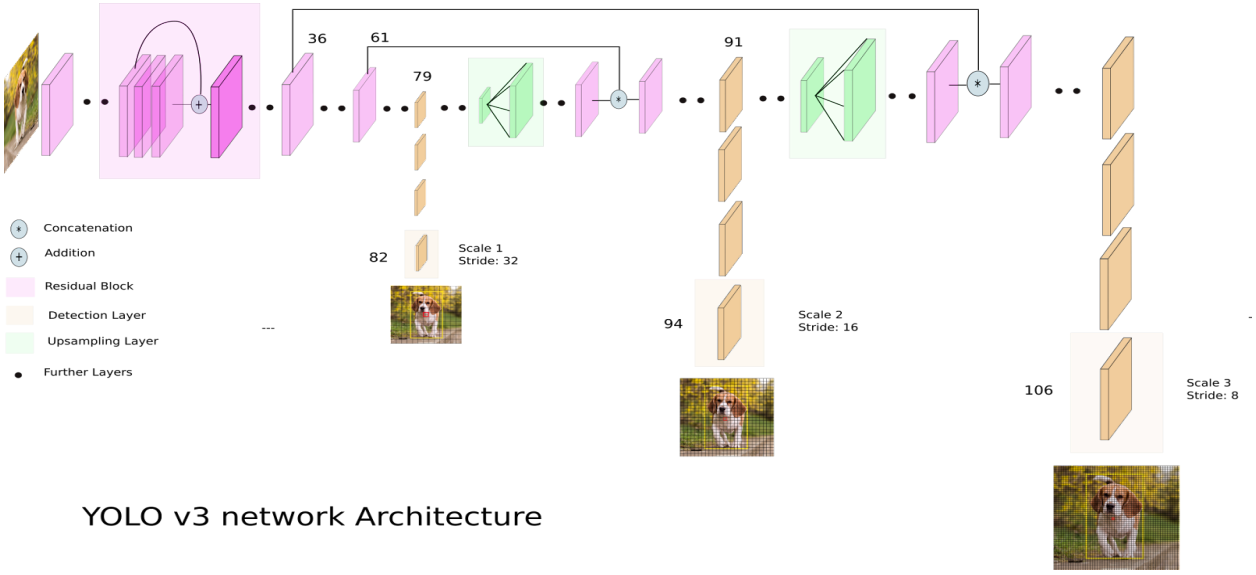
Following is the layer structure in the YOLO v3

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Darknet-53 Implemented in code:

```
def darknet53(input_data):
    input_data = convolutional(input_data, (3, 3,  3,  32))
    input_data = convolutional(input_data, (3, 3, 32,  64),
downsample=True)

    for i in range(1):
        input_data = residual_block(input_data,  64,  32, 64)

    input_data = convolutional(input_data, (3, 3,  64, 128),
downsample=True)

    for i in range(2):
        input_data = residual_block(input_data, 128,  64, 128)

    input_data = convolutional(input_data, (3, 3, 128, 256),
downsample=True)

    for i in range(8):
        input_data = residual_block(input_data, 256, 128, 256)

    route_1 = input_data
    input_data = convolutional(input_data, (3, 3, 256, 512),
downsample=True)

    for i in range(8):
        input_data = residual_block(input_data, 512, 256, 512)

    route_2 = input_data
    input_data = convolutional(input_data, (3, 3, 512, 1024),
downsample=True)

    for i in range(4):
        input_data = residual_block(input_data, 1024, 512, 1024)

    return route_1, route_2, input_data
```

Structure of YOLO v3



YOLO v3 network Architecture

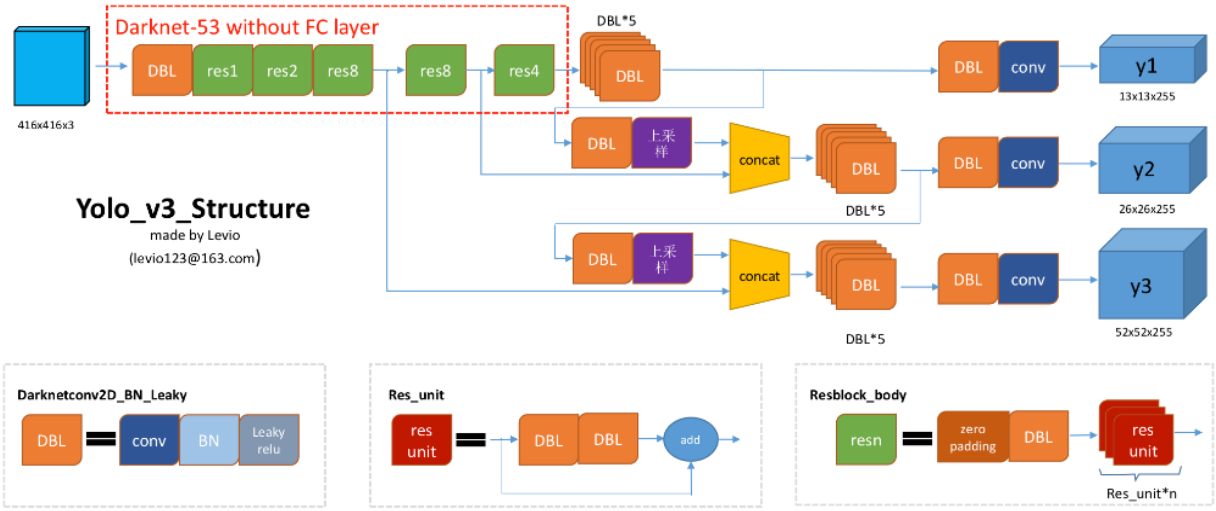
From the above architecture image, you can see that YOLO makes detection in 3 different scales to accommodate various objects sizes by using strides of 32, 16, and 8. This means, if we feed an input image of size 416x416, YOLOv3 will make detection on the scale of 13x13, 26x26, and 52x52.

YOLOv3 downsamples the input image into 13 x 13 and predicts the 82nd layer for the first scale. The 1st detection scale yields a 3-D tensor of size 13x13x255.

After that, YOLOv3 takes the feature map from layer 79 and applies one convolutional layer before upsampling it by a factor of 2 to have a size of 26 x 26. This upsampled feature map is then concatenated with the feature map from layer 61. The concatenated feature map is subjected to a few more convolutional layers until the 2nd detection scale is performed at layer 94. The second prediction scale produces a 3-D tensor of size 26 x 26 x 255.

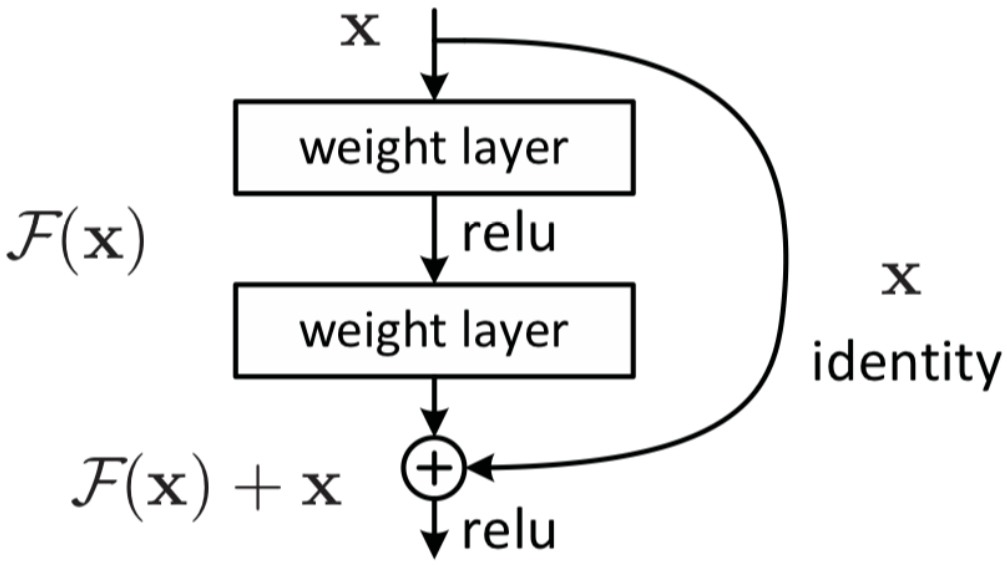
But still, seeing Darknet-53 and Yolo v3 structure, we can't fully understand all layers. This is why I have one more figure with the overall architecture of the YOLOv3 network.

The picture below shows that the input picture of size 416x416 gets three branches after entering the Darknet-53 network. These branches undergo a series of convolutions, upsampling, merging, and other operations. Three feature maps with different sizes are finally obtained, with shapes of [13, 13, 255], [26, 26, 255] and [52, 52, 255]:



Residual Module

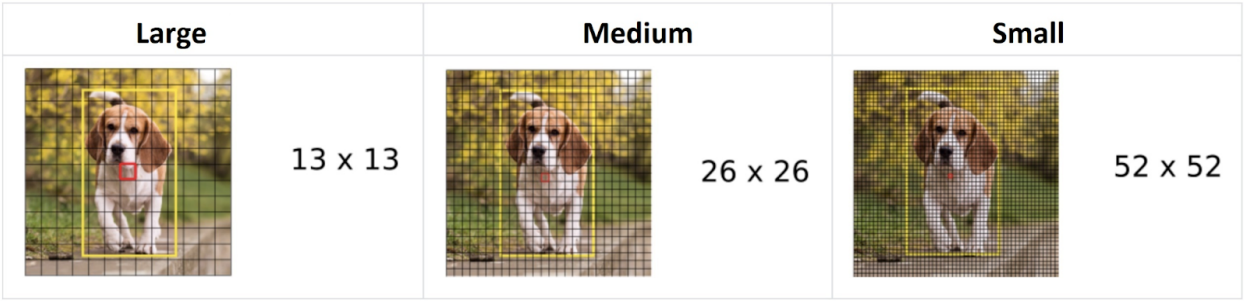
The most significant feature of the residual module is the use of a short cut mechanism (somewhat similar to the short circuit mechanism in the circuit) to alleviate the gradient disappearance problem caused by increasing the depth in the neural network, thereby making the neural network easier to optimize. It uses identity mapping to establish a direct correlation channel between input and output so that the network can concentrate on learning the residual between input and output.



YOLO v3 Detection

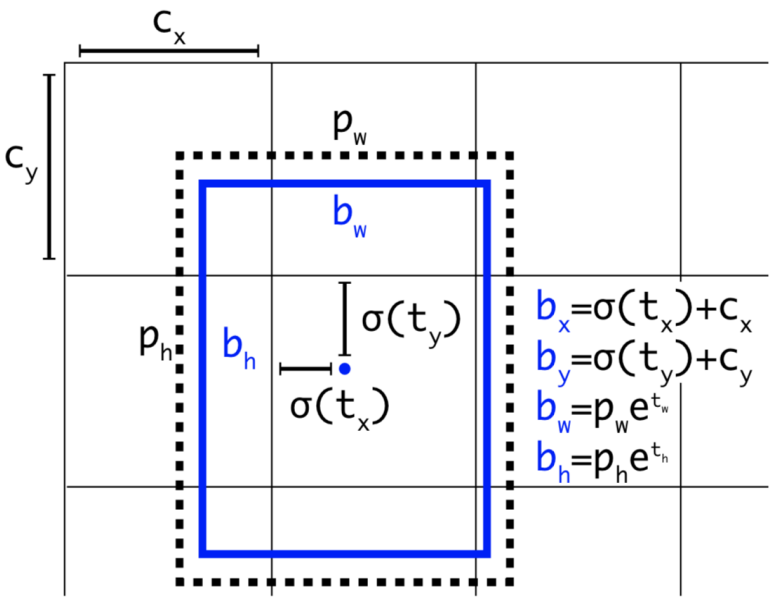
Multi-scale detection

YOLO performs coarse, medium, and fine meshing of the input image to predict large, medium, and small objects, respectively. If the input picture size is 416X416, then the coarse, medium, and fine grid sizes are 13x13, 26x26, and 52x52, respectively. In this way, it is scaled by 32, 16, and 8 times in length and width, respectively:



Dimensions of the Bounding Box

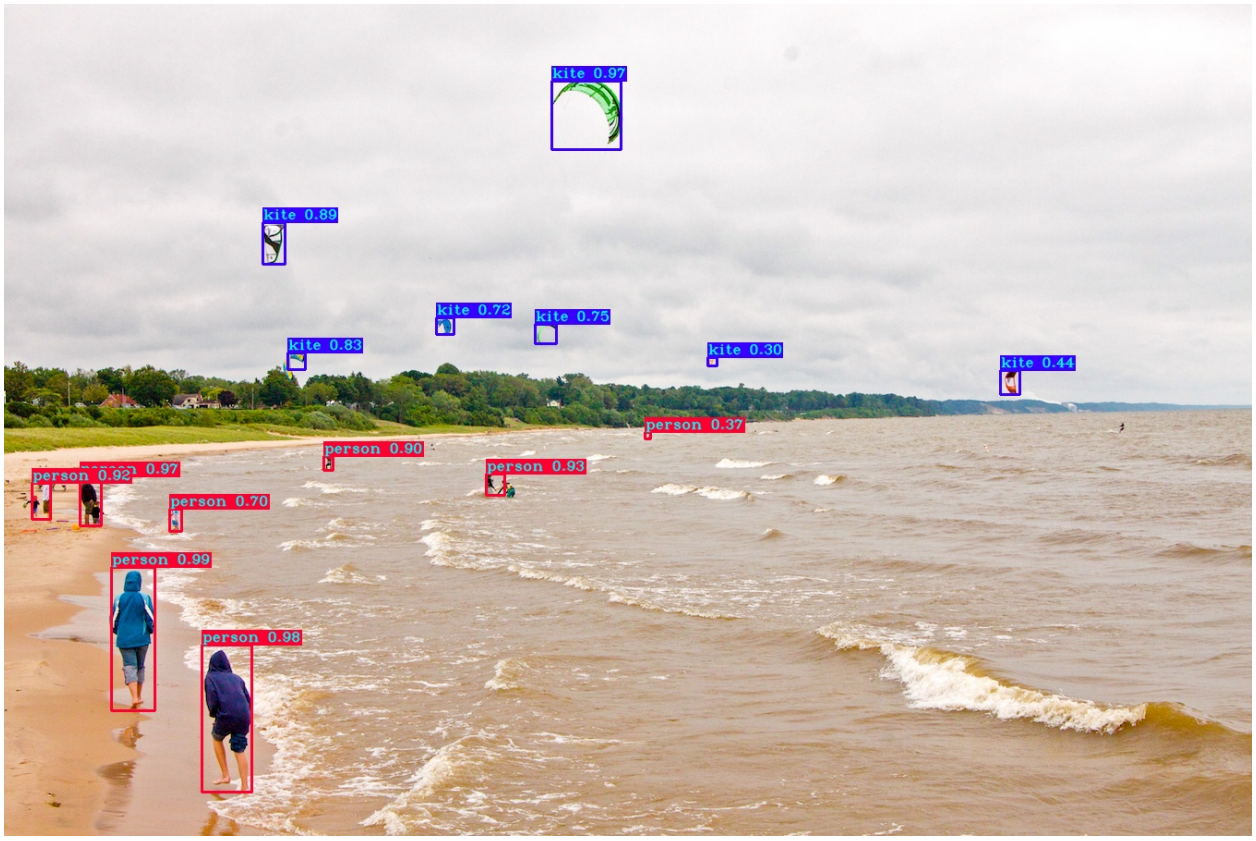
The output of the three branches of the YOLOv3 network will be sent to the decode function to decode the channel information of the Feature Map. The dimensions of the bounding box are predicted by applying a log-space transformation to the output and then multiplying with an anchor: in the following picture: the black dotted box represents the a priori box (anchor), and the blue box represents the prediction box.



- **b** denotes the length and width of the prediction frame, respectively, and **P** represents the length and width of the a priori frame.
- **t** represents the offset of the object's center from the upper left corner of the grid, and **C** represents the coordinates of the upper left corner of the grid.

Results

- Hence applying the above knowledge, I constructed and compiled the YOLO v3 model in Tensorflow and keras.
- It also required transferring weights from original Darknet weights to constructed model.
- Object Detection was carried out and satisfactory results were obtained.



The above is the result obtained after running the algorithm over the image. The objects are detected with their class and percentage probability.

Future Scope and Improvements

- The YOLO v3, can be further extended to detect tiny objects, which can be easily done with using YOLOv3-tiny.
- The model can be tweaked according to the needs, as there is a tradeoff between speed and accuracy, as YOLOv3-tiny is much faster yet less accurate.
- Further, this object detection model can be extended to real time object tracking. Where approaches such as Deep Regression Networks, ROLO (Recurrent - YOLO).
- Also, using DEEP SORT (Simple Real-time Tracker), helps us to track objects in real time and detect them.

Conclusion

- Exploring object detection was an excellent learning experience. I was able to realize the wide range of applications that this technology can have.
- This technology can be used in autonomous vehicles, face detectors, image filters, satellite mapping, robot vision, and so much more.
- Computer vision is still in a growing age and after my experience with it in the last few months, I am sure that it has the potential to completely change the way we live.

Bibliography

1. https://pjreddie.com/media/files/papers/YOLOv3.pdf
2. https://viso.ai/deep-learning/yolov3-overview/
3. https://towardsdatascience.com/object-detection-using-yolov3-and-opencv-19ee0792a420
4. https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/
5. https://pylessons.com/YOLOv3-TF2-introduction
6. https://www.analyticsvidhya.com/blog/2021/06/implementation-of-yolov3-simplified/