# Object Detection using YOLOv2

Pushpendra Singh Chauhan

Department of Computer Science

Illinois Institute of Technology

November 29, 2020

## Abstract

This is a report for the final project. It will contain the description of the problem I am trying to solve, the algorithms I employed to solve the problem, implementation details like program design issues and the results obtained. The results obtained will be analyzed for correctness. The performance of the algorithm will be evaluated and discussed.

## 1 Problem Statement:

- Write a program in Python using TensorFlow and Keras libraries to perform object detection (i.e. predicting the bounding box and label of each object present in the image).
- To perform the task of object detection, implement the network architecture of YOLOv2 or YOLO9000 from scratch using TensorFlow and Keras as deep learning libraries.

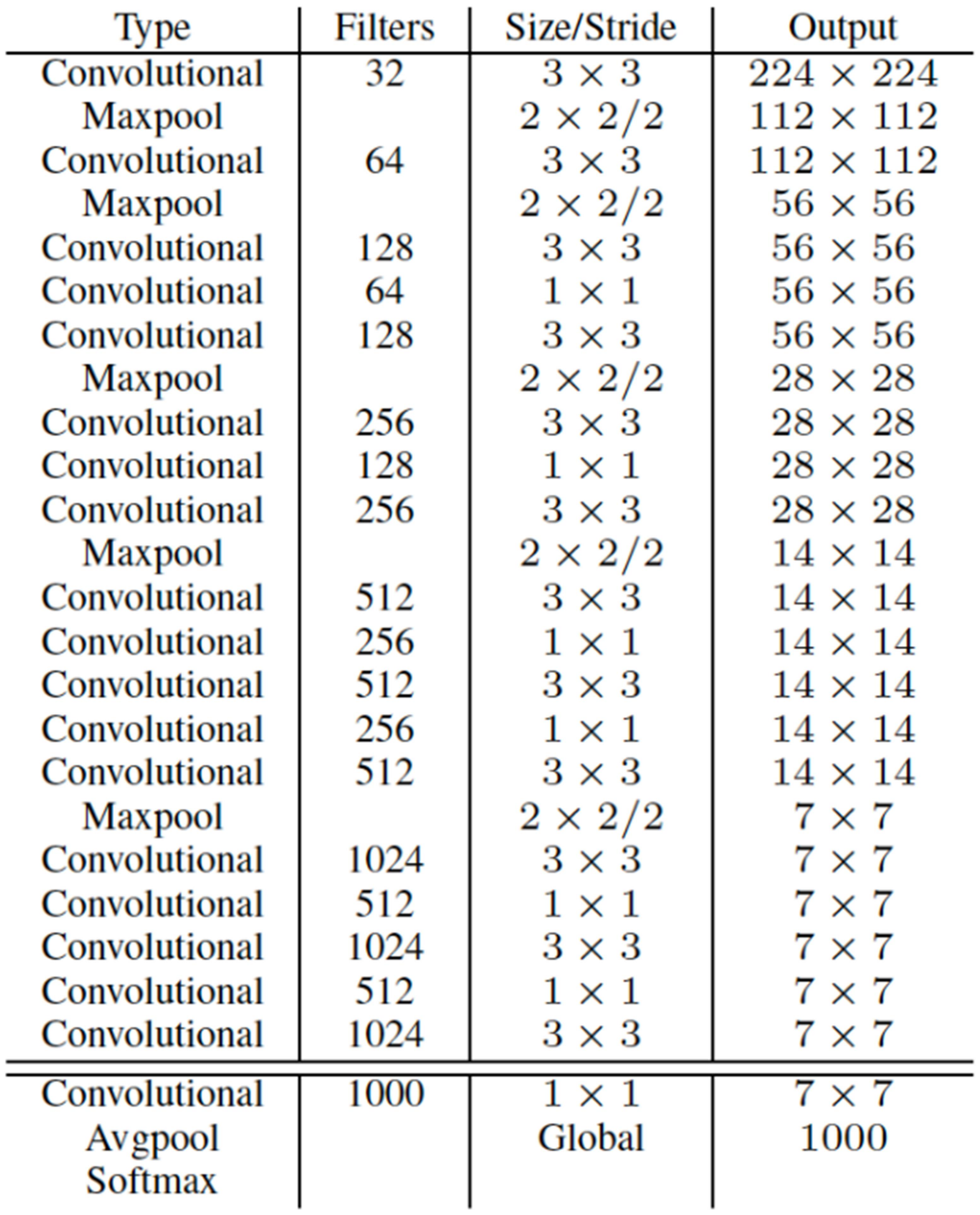
## 2 Proposed Solution:

- For this task I have used PASCAL VOC 2012 dataset. It consists of 20 unique classes as the class labels/target variable. The names of these classes are: aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable, dog, horse, motorbike, person, pottedplant, sheep, sofa, train and tvmonitor. The training data provided consists of a set of images; each image has an annotation file giving a bounding box and object class label for each object in one of the twenty classes present in the image.
- Download the PASCAL VOC 2012 dataset from this link http://host.robots.ox.ac.uk/pascal/VOC/voc2012/VOCtrainval_11-May-2012.tar in Google colab and extract the data.
- Used TensorFlow, Keras as GPU frameworks and OpenCV for the implementation of the project.
- Used xml to parse the data from annotation file.
- Used OpenCV for performing some image processing task.
- Used Convolutional Neural Network for training, classifying and detecting objects in the images.
- **Convolutional Neural Network:**
  (a) Convolutional Neural Network (CNN) is a special architecture of neural networks.
  (b) Here, the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.
  (c) Convolution layer takes in an image as input signal and applies a filter over it, essentially multiplies the input image with the kernel to get the modified image. Mathematically, a convolution of two functions f and g is nothing but dot product of the input function and a kernel function.
  (d) Pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

(e) The common types of pooling are max, min and average pooling. Max pooling is based on picking up the maximum value from the selected region. Min pooling is based on picking up the minimum value from the selected region. Average pooling is based on averaging all the values from the selected region.

- YOLOv2 uses k-means clustering to find the best anchor box sizes for the given dataset. The anchors defined in the project are taken from the following blog: https://fairyonice.github.io/Part_1_Object_Detection_with_Yolo_for_VOC_2014_data_anchor_box_clustering.html
- We defined a ImageReader class to process an image. It takes in an image and returns the resized image and all the objects in the image.
- Next, we used "BestAnchorBoxFinder" function which finds the best anchor box for a particular object. This is done by finding the anchor box with the highest IOU (Intersection over union) with the bounding box of the object.
- Next, we define a custom Batch generator to get a batch of 16 images and its corresponding bounding boxes.
- Now, I am adding a function to prepare the input and the output. The input is a (448, 448, 3) image and the output is a (7, 7, 30) tensor.
- Next, we defined the model architecture of YOLOv2/YOLO9000 from scratch as described in the original paper. (Note:- I have implemented the architecture from scratch on my own and for the rest of the part I followed the Yumi's blog as I am beginner to computer vision and took the code from his blog. Link to Yumi's blog: https://fairyonice.github.io/Part_1_Object_Detection_with_Yolo_for_VOC_2014_data_anchor_box_clustering.html )

- Darknet-19 architecture given in the research paper is as follows:



| Type | Filters | Size/Stride | Output |
|---|---|---|---|
| Convolutional | 32 | 3 × 3 | 224 × 224 |
| Maxpool | | 2 × 2/2 | 112 × 112 |
| Convolutional | 64 | 3 × 3 | 112 × 112 |
| Maxpool | | 2 × 2/2 | 56 × 56 |
| Convolutional | 128 | 3 × 3 | 56 × 56 |
| Convolutional | 64 | 1 × 1 | 56 × 56 |
| Convolutional | 128 | 3 × 3 | 56 × 56 |
| Maxpool | | 2 × 2/2 | 28 × 28 |
| Convolutional | 256 | 3 × 3 | 28 × 28 |
| Convolutional | 128 | 1 × 1 | 28 × 28 |
| Convolutional | 256 | 3 × 3 | 28 × 28 |
| Maxpool | | 2 × 2/2 | 14 × 14 |
| Convolutional | 512 | 3 × 3 | 14 × 14 |
| Convolutional | 256 | 1 × 1 | 14 × 14 |
| Convolutional | 512 | 3 × 3 | 14 × 14 |
| Convolutional | 256 | 1 × 1 | 14 × 14 |
| Convolutional | 512 | 3 × 3 | 14 × 14 |
| Maxpool | | 2 × 2/2 | 7 × 7 |
| Convolutional | 1024 | 3 × 3 | 7 × 7 |
| Convolutional | 512 | 1 × 1 | 7 × 7 |
| Convolutional | 1024 | 3 × 3 | 7 × 7 |
| Convolutional | 512 | 1 × 1 | 7 × 7 |
| Convolutional | 1024 | 3 × 3 | 7 × 7 |
| Convolutional | 1000 | 1 × 1 | 7 × 7 |
| Avgpool | | Global | 1000 |
| Softmax | | | |

- YOLOv2 model architecture repeatedly stacks Convolution + Batch Normalization + Leaky Relu layers until the image shape reduces to the grid cell size.
- We used pre-trained weights for first 22 layers and trained only the last convolution layer. I have trained the model on Google colab and it took approximately 8.5 hours for training. I have downloaded the pre-trained

weights from here: https://pjreddie.com/media/files/yolov2.weights inside the Google colab notebook.

- For calculating the loss I have taken the code from Yumi's blog.
- I have only trained the final 23rd layer and freeze the other weights as it will take a long time to train the full network.
- After the model is trained I saved it so that it can be used for inference part later on.

## 3 Implementation Details:

- I have implemented the model architecture of YOLOv2/YOLO9000 from scratch on my own in the filename "model_architecture.py" which is in the 'src' folder.
- I have trained the model on Google colab in the filename "model_training.ipynb" which is also in the folder 'src'.
- Last part of the project i.e. objects detection in the images using previously trained model is performed in the filename "Inference.ipynb" which is also in the 'src' folder.
- The name of the trained model file is 'weights_yolo_on_voc2012.h5'. Since the size of the model is 579 MB approximately. So, I have uploaded it on the Google drive. Please click on this link to download the file of the trained model: https://drive.google.com/file/d/1vdP69qEVdRM3NmB1LvFdus0_Sy-R8eBN/view?usp=sharing .
- After downloading the file of the model, put it in the folder named 'yolo_v2' which is inside the 'src' folder. After that you can execute 'Inference.ipynb' file on various test images given in 'data' folder.
- 'Inference.ipynb' file can be executed in jupyter notebook.
- Folder structure: project (main folder) →

  (1). src (sub-folder) – It contains four source code files and their names are: "backend.py", "Inference.ipynb", "model_architecture.py" and "model_training.ipynb". It also contains one sub-folder named 'yolo_v2'. "weights_yolo_on_voc2012.h5" file should be put in 'yolo_v2' folder.

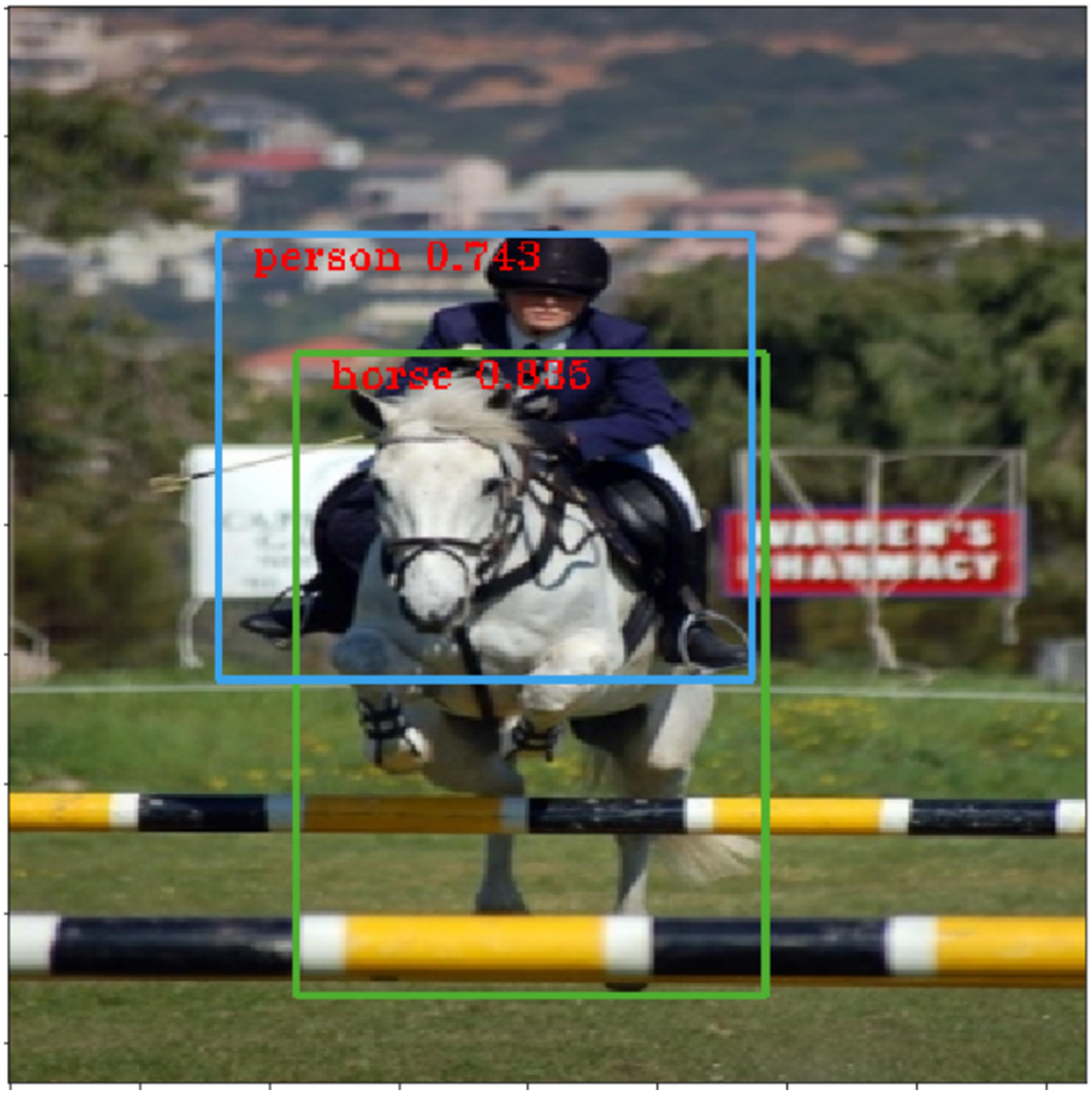
(2). data (sub-folder) – It contains all test files (images).

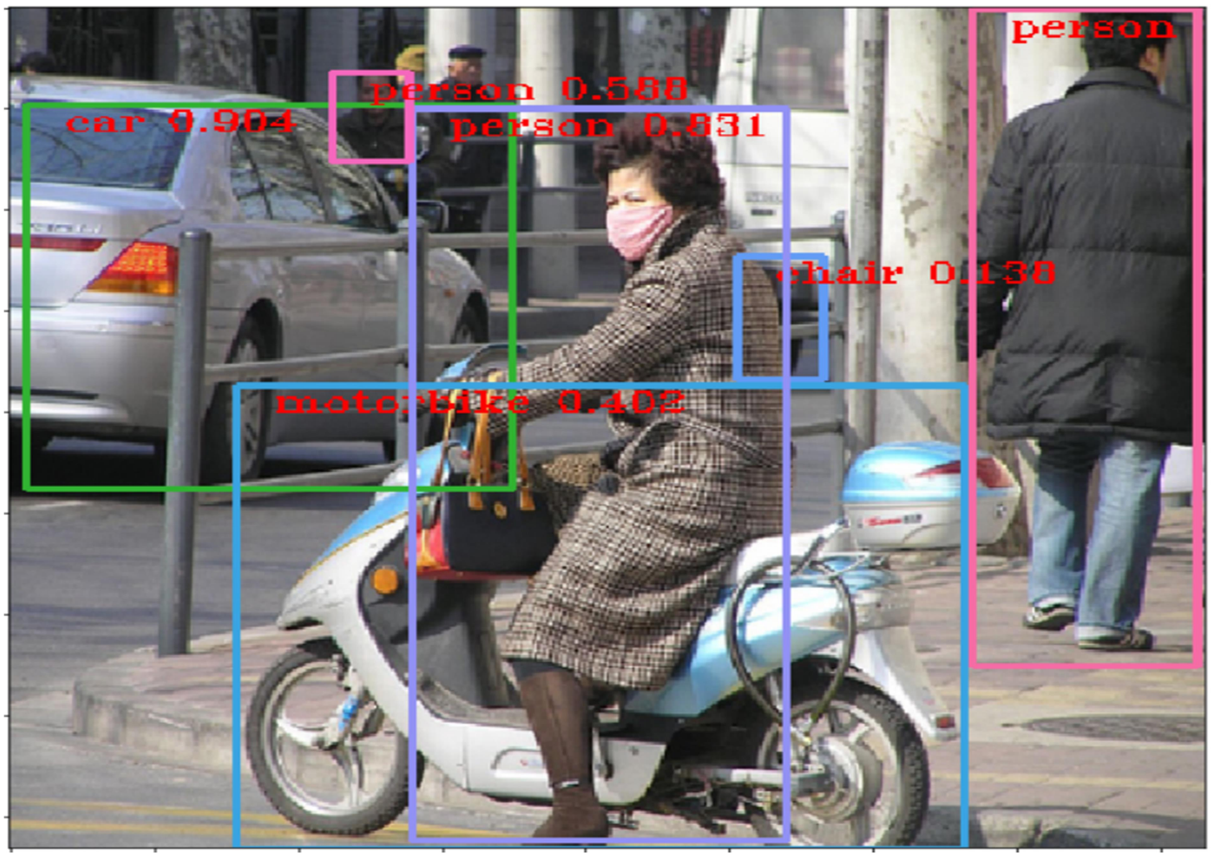
(3). doc (sub-folder) – It contains one pdf file and its name is “project report.pdf”.

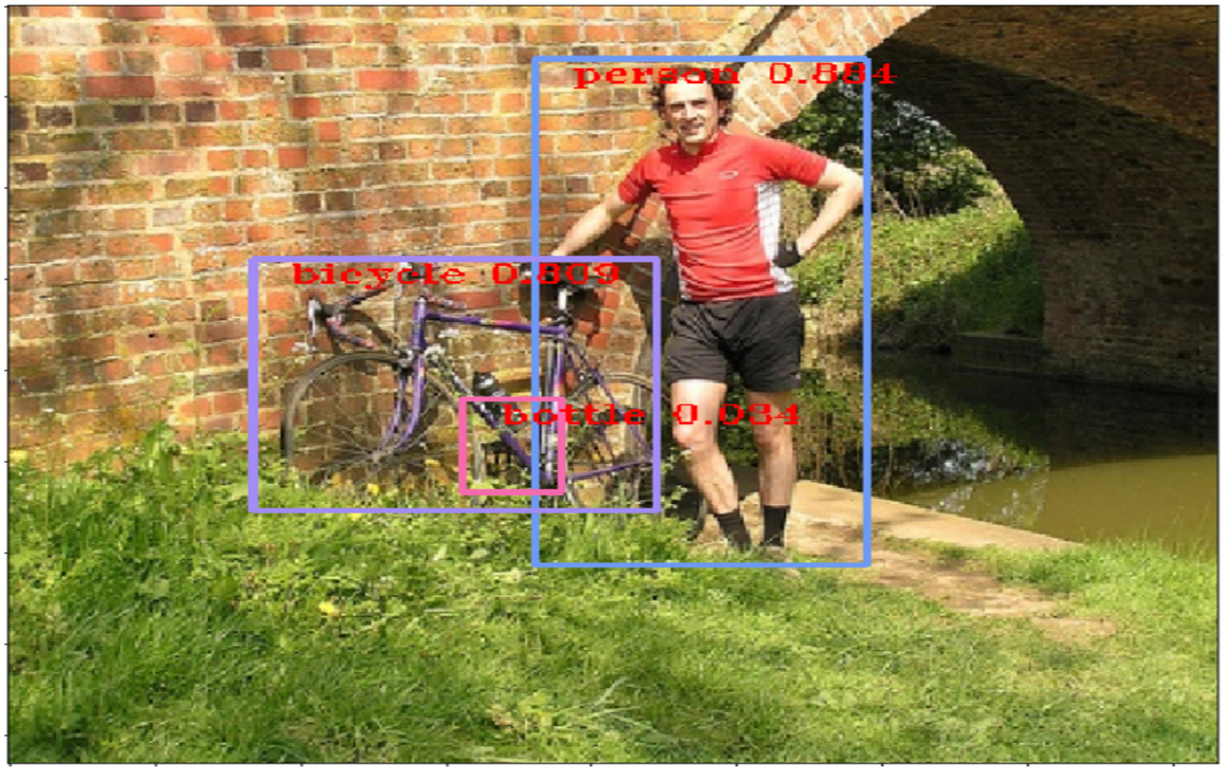
(4). presentation (sub-folder) – It contains one pdf file and its name is “project presentation.pdf”.

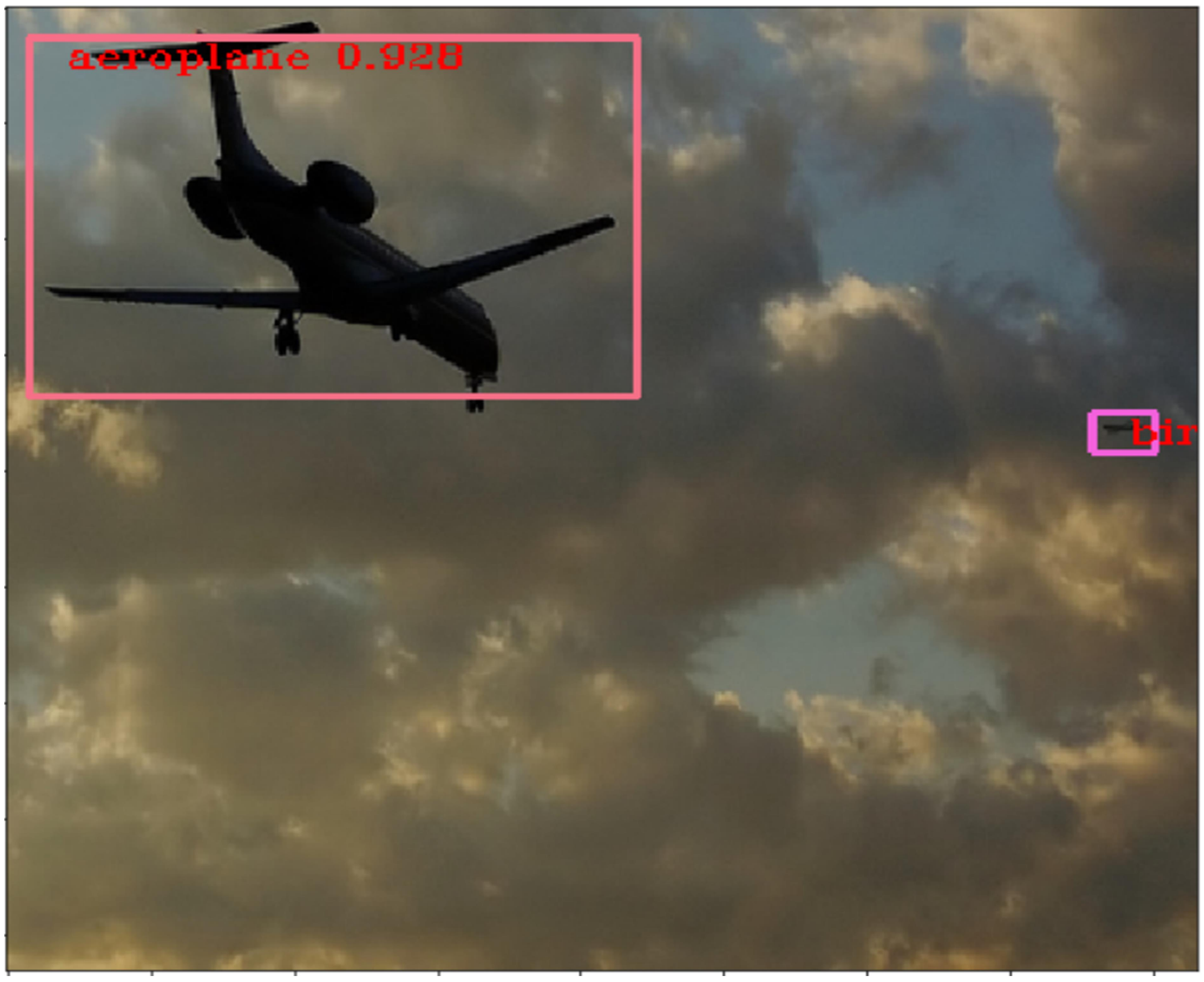
(5). sources (sub-folder) – It contains two files. The names of these files are: “YOLO9000 research paper.pdf” and “backend.py”.

## 4 Results and discussion:




person
person 0.588
car 0.904
person 0.831
chair 0.138
motorbike 0.402


person 0.884
bicycle 0.809
bottle 0.034



## 5 References:

- http://host.robots.ox.ac.uk/pascal/VOC/voc2012/
- https://ieeexplore.ieee.org/document/8100173
- https://arxiv.org/pdf/1612.08242.pdf
- https://pjreddie.com/darknet/yolo/
- https://www.geeksforgeeks.org/yolo-v2-object-detection/
- https://arxiv.org/pdf/1506.02640.pdf
- https://medium.com/@y1017c121y/how-does-yolov2-work-daaaa967c5f7
- https://fairyonice.github.io/Part_1_Object_Detection_with_Yolo_for_VOC_2014_data_anchor_box_clustering.html

- https://fairyonice.github.io/Part%202_Object_Detection_with_Yolo_using_VOC_2014_data_input_and_output_encoding.html
- https://fairyonice.github.io/Part_3_Object_Detection_with_Yolo_using_VOC_2012_data_model.html
- https://fairyonice.github.io/Part_4_Object_Detection_with_Yolo_using_VOC_2012_data_loss.html
- https://fairyonice.github.io/Part_5_Object_Detection_with_Yolo_using_VOC_2012_data_training.html
- https://fairyonice.github.io/Part_6_Object_Detection_with_Yolo_using_VOC_2012_data_inference_image.html