International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Object Detection Using Python

Khusbhu Bharti

Master of Technology in Computer Science & Engineering, Lingaya's Vidyapeeth, Faridabad, India

Abstract: ***Computer Vision is the branch of the science of computers and software systems which can recognize as well as understand images and scenes. Computer Vision is consists of various aspects such as image recognition, object detection, image generation, image super-resolution and many more. Object detection is widely used for face detection, vehicle detection, pedestrian counting, web images, security systems and self-driving cars. In this project, we are using highly accurate object detection-algorithms and methods such as R-CNN, Fast-RCNN, Faster-RCNN, RetinaNet and fast yet highly accurate ones like SSD and YOLO. Using these methods and algorithms, based on deep learning which is also based on machine learning require lots of mathematical and deep learning frameworks understanding by using dependencies such as Tensor Flow, OpenCV, image AI etc, we can detect each and every object in image by the area object in an highlighted rectangular boxes and identify each and every object and assign its tag to the object. This also includes the accuracy of each method for identifying objects.***

Keywords: Object detection, Computer Vision, Web images, R-CNN, Tensor Flow, OpenCV, Convolutional network, Bayes decision rule, Gaussian distribution function

1. Introduction

A few years ago, the creation of the software and hardware image processing systems was mainly limited to the development of the user interface, which most of the programmers of each firm were engaged in. The situation has been significantly changed with the advent of the Windows operating system when the majority of the developers switched to solving the problems of image processing itself. However, this has not yet led to the cardinal progress in solving typical tasks of recognizing faces, car numbers, road signs, analyzing remote and medical images, etc. Each of these "eternal" problems is solved by trial and error by the efforts of numerous groups of the engineers and scientists. As modern technical solutions are turn out to be excessively expensive, the task of automating the creation of the software tools for solving intellectual problems is formulated and intensively solved abroad. In the field of image processing, the required tool kit should be supporting the analysis and recognition of images of previously unknown content and ensure the effective development of applications by ordinary programmers. Just as the Windows toolkit supports the creation of interfaces for solving various applied problems.

Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization is refers to identifying the location of one or more objects in an image and drawing an abounding box around their extent. Object detection does the work of combines these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term "Object Recognition", they often mean "Object detection". It may be challenging for beginners to distinguish between different related computer vision tasks.

2. Overview

The aim of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image. Generally, only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored. Each detection of the image is reported with some form of pose information. This is as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. In some other situations, the pose information is more detailed and contains the parameters of a linear or non-linear transformation. For example for face detection in a face detector may compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face. An example of a bicycle detection in an image that specifies the locations of certain parts is shown in Figure 1. The pose can also be defined by a three-dimensional transformation specifying the location of the object relative to the camera. Object detection systems always construct a model for an object class from a set of training examples. In the case of a fixed rigid object in an image, only one example may be needed, but more generally multiple training examples are necessary to capture certain aspects of class variability

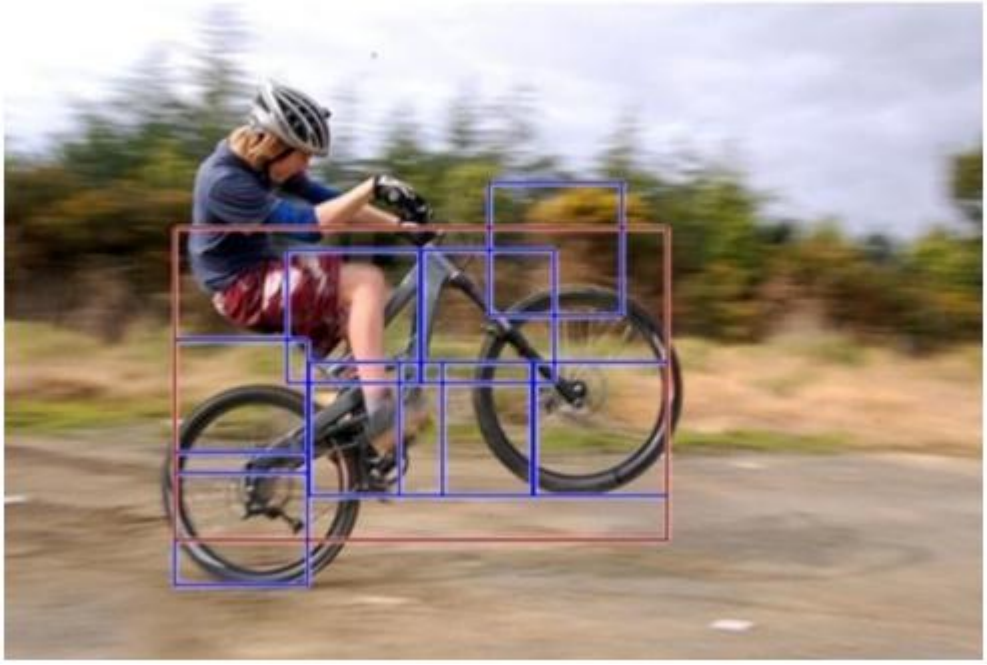

Figure 1

Convolutional implementation of the sliding windows Before we discuss the implementation of the sliding window using convents, let us analyze how we can convert the fully connected layers of the network into convolutional layers. Fig.2 shows a simple convolutional network with two fully connected layers each of shape.

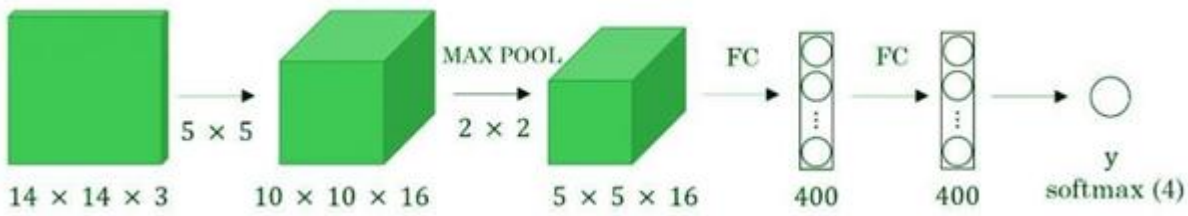

Figure 2: Simple convolution network

Object detection is an important task, yet challenging vision task. It is a critical part of many applications such as image search, image auto-annotation and scene understanding, object tracking. Moving object tracking of video image sequences was one of the most important subjects in computer vision. It had already been applied in many computer vision fields, such as smart video surveillance (Arun Hampapur 2005), artificial intelligence, military guidance, safety detection and robot navigation, medical and biological application. In recent years, a number of successful single-object tracking system appeared, but in the presence of several objects, object detection becomes difficult and when objects are fully or partially occluded, they are obtruded from the human vision which further increases the problem of detection. Decreasing illumination and acquisition angle. The proposed MLP based object tracking system is made robust by an optimum selection of unique features and also by implementing the Adaboost strong classification method.

a) Background Subtraction

The background subtraction method by Horprasert et al (1999), was able to cope with local illumination changes, such as shadows and highlights, even globe illumination changes. In this method, the background model was statistically modelled on each pixel. Computational colour mode, include the brightness distortion and the chromaticity distortion which was used to distinguish shading background from the ordinary background or moving foreground objects. The background and foreground subtraction method used the following approach. A pixel was modelled by a 4-tuple [Ei, si, ai, bi], where Ei-a vector with expected colour value, si-a vector with the standard deviation of colour value, ai-the variation of the brightness distortion and bi was the variation of the chromaticity distortion of the ith pixel. In the next step, the difference between the background image and the current image was evaluated. Each pixel was finally classified into four categories: original background, shaded background or shadow, highlighted background and moving foreground object. Liyuan Li et al (2003), contributed a method for detecting foreground objects in non-stationary complex environments containing moving background objects. A Bayes decision rule was used for classification of background and foreground changes based on inter-frame colour co-occurrence statistics. An approach to store and fast retrieve colour cooccurrence statistics was also established. In this method, foreground objects were detected in two steps. First, both the foreground and the background changes are extracted using background subtraction and temporal differencing. The frequent background changes were then recognized using the Bayes decision rule based on the learned colour co-occurrence statistics. Both short-term and

long term strategies to learn the frequent background changes were used. An algorithm focused on obtaining the stationary foreground regions as said by Álvaro Bayona et al (2010), which was useful for applications like the detection of abandoned/stolen objects and parked vehicles. This algorithm mainly used two steps. Firstly, asub-sampling scheme based on background subtraction techniques was implemented to obtain stationary foreground regions. This detects foreground changes at different time instants in the same pixel locations. This was done by using a Gaussian distribution function. Secondly, some modifications were introduced on this base algorithm such as thresh holding the previously computed subtraction. The main purpose of this algorithm was reducing the amount of stationary foreground detected.

b) Existing Methods

To circumvent the problem of selecting a huge number of regions, Ross Girshick et al. proposed a method where we use the selective search for extract just 2000 regions from the image and he called them region proposals. Therefore, instead of trying to classify the huge number of regions, you can just work with 2000 regions. These 2000 region proposals are generated by using the selective search algorithm which is written below.

Selective Search:

1) Generate the initial sub-segmentation, we generate many candidate regions
2) Use the greedy algorithm to recursively combine similar regions into larger ones
3) Use generated regions to produce the final candidate region proposals

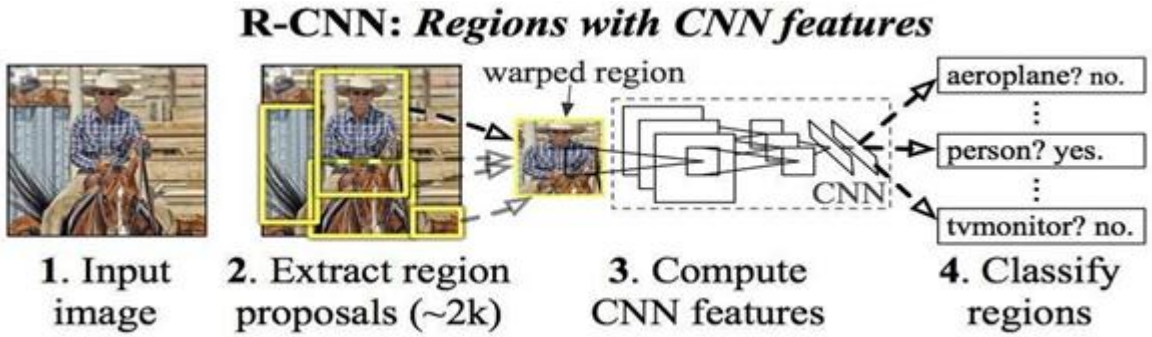

Figure 3

These 2000 candid and fed into a convolutional neural output. The CNN plays a role of feature extracted from the image and the extract object within that candidate region within the region proposals, the algorithm the precision of the bounding box. Have predicted the presence of a person been cut in half. Therefore, the offset region proposal.

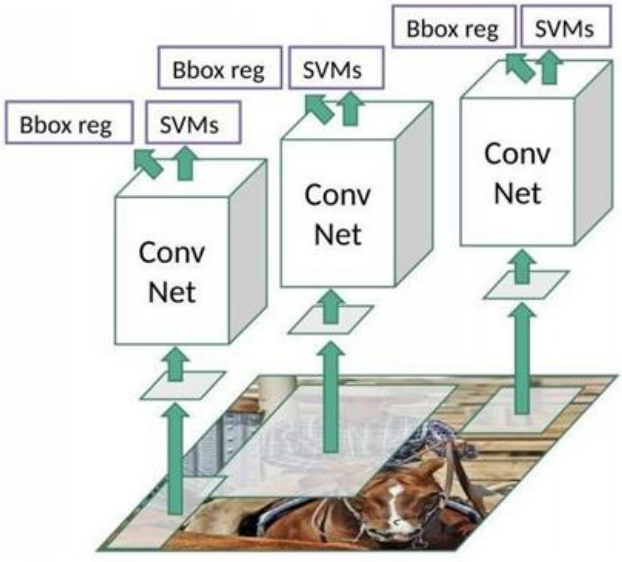

Figure 4: R-CNN

Problems with R-CNN

1) It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
2) It cannot be implemented real time as it takes around 47 seconds for each test image.

a) Faster R-CNN

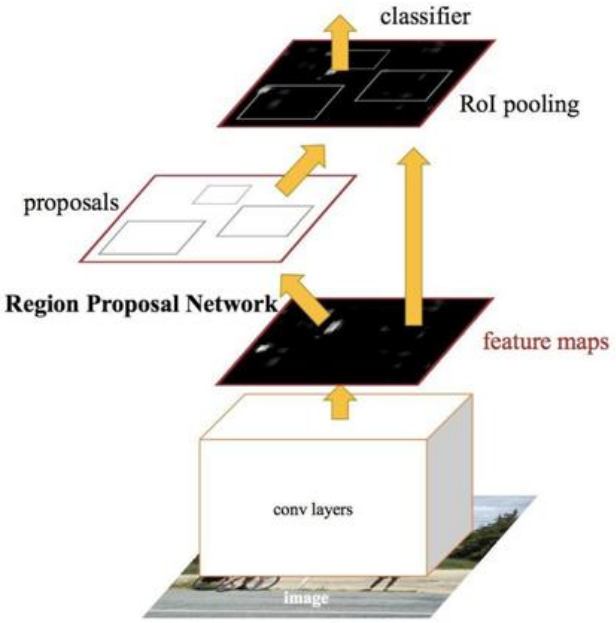

Figure 5: Faster R-CNN

Both of the above algorithms (R-CNN & Fast Rxi-CNN) uses selective search to find out the region proposals. Selective search is the slow and time-consuming process which affects the performance of the network.

Similar to Fast R-CNN, the image is provided as an input to a convolutional network which provides a convolutional feature map. Instead of using the selective search algorithm for the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted the region which is proposals are then reshaped using an RoI pooling layer which is used to classify the image within the proposed region and predict the offset values for the bounding boxes.

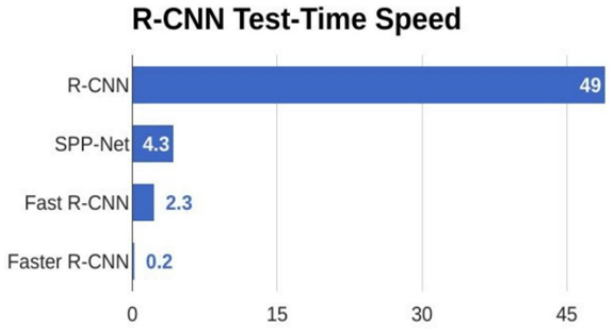

Figure 6: Comparison of test-time speed of object detection algorithms

From the above graph, you can see that Faster R-CNN is much faster than it's predecessors. Therefore, it can even be used for real-time object detection.

b) YOLO — You Only Look Once

All the previous object detection algorithms have used regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which has high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much is different from the region based algorithms which seen above. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

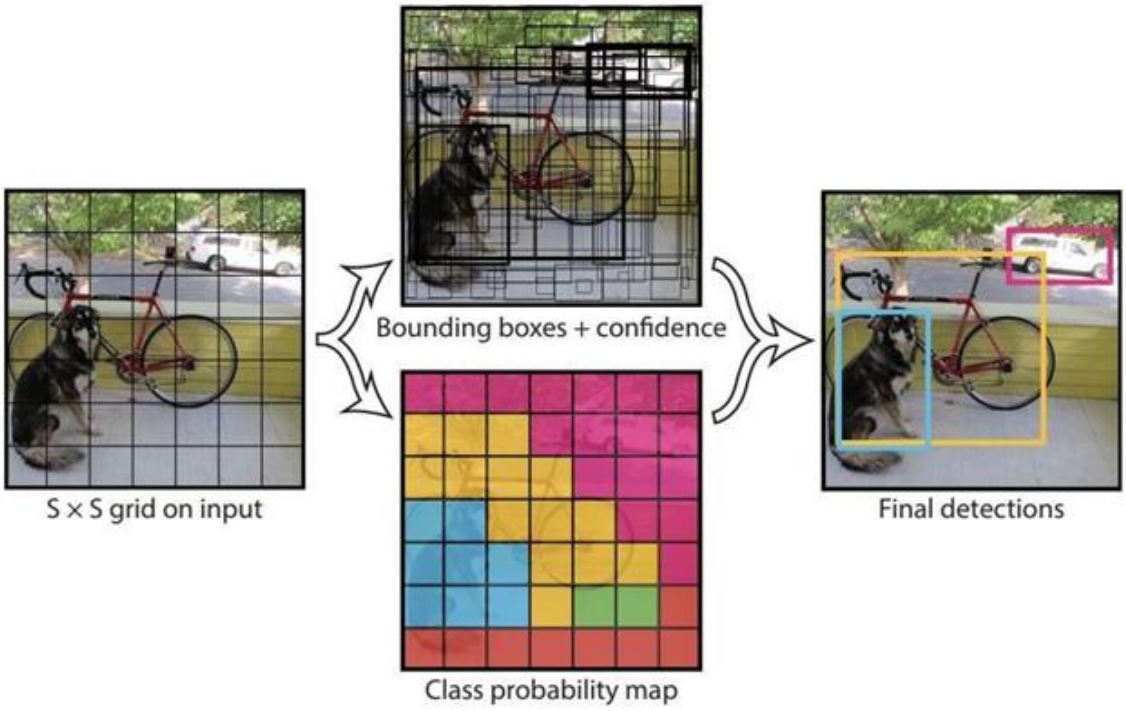

Figure 7

YOLO works by taking an image and split it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding box, the network gives an output a class probability and offset values for the bounding box. The bounding boxes have the class probability above a threshold value is selected and used to locate the object within the

image.

YOLO is orders of magnitude faster (45 frames per second) than any other object detection algorithms. The limitation of YOLO algorithm is that it struggles with the small objects within the image, for example, it might have difficulties in identifying a flock of birds. This is due to the spatial constraints of the algorithm.

System Requirement

Install Python on your computer system
Install ImageAI and its dependencies like tensorflow, Numpy, OpenCV, etc.

1) Download the Object Detection model file (Retinanet)
2) Steps to be followed:-
3) Download and install Python version 3 from official Python Language website
   https://python. org

4) Install the following dependencies via pip:
5) Tensorflow:

pip install tensorflow-command

6) Numpy:

pip install numpy-command

7) SciPy:

pip install scipy-command

8) OpenCV:

pip install opencv-python-command

9) Pillow:

pip install pillow -command

10) Matplotlib:

pip install matplotlib-command

11) H5py:

pip install h5py

12) Keras

pip install keras

13) ImageAI:

pip3 install imageai--upgrade

14) Download the RetinaNet model file that will be used for object detection using following link

https://github. com/OlafenwaMoses/ImageAI/releases/download/1.0/resnet 50_coco_best_v2.0.1. h5

Copy the RetinaNet model file and the image you want to detect to the folder that contains the python file.

InceptionV3:
Inception v3 is widely used as image recognition model that has showed to obtain accuracy of greater than 78.1% on the ImageNet dataset. The model is the culmination of many ideas developed by researchers over years. It is based on "Rethinking the Inception Architecture Computer Vision" by Szegedy.

The model is made of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concats, dropouts, and fully connected layers. Batchnorm is used more throughout the model and applied to activation inputs. Loss is computed via Softmax.

A high-level diagram of the model is shown below:

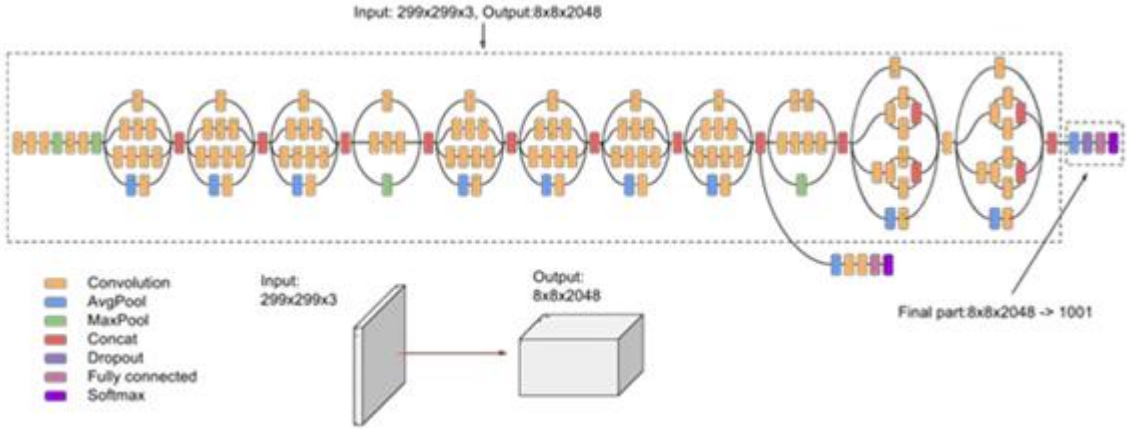

Figure 8

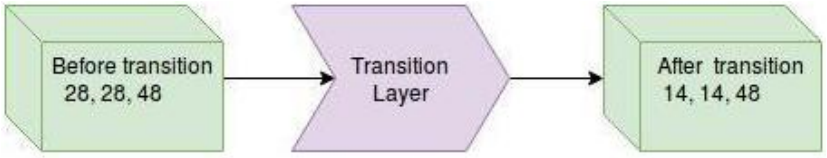

Figure 9: Transition layer

3. Results and Discussion

Figure 10: Before Detection

This is a sample image we feed to the algorithm and expect our algorithm to detect and identify objects in the image and label them according to the class assigned to it.

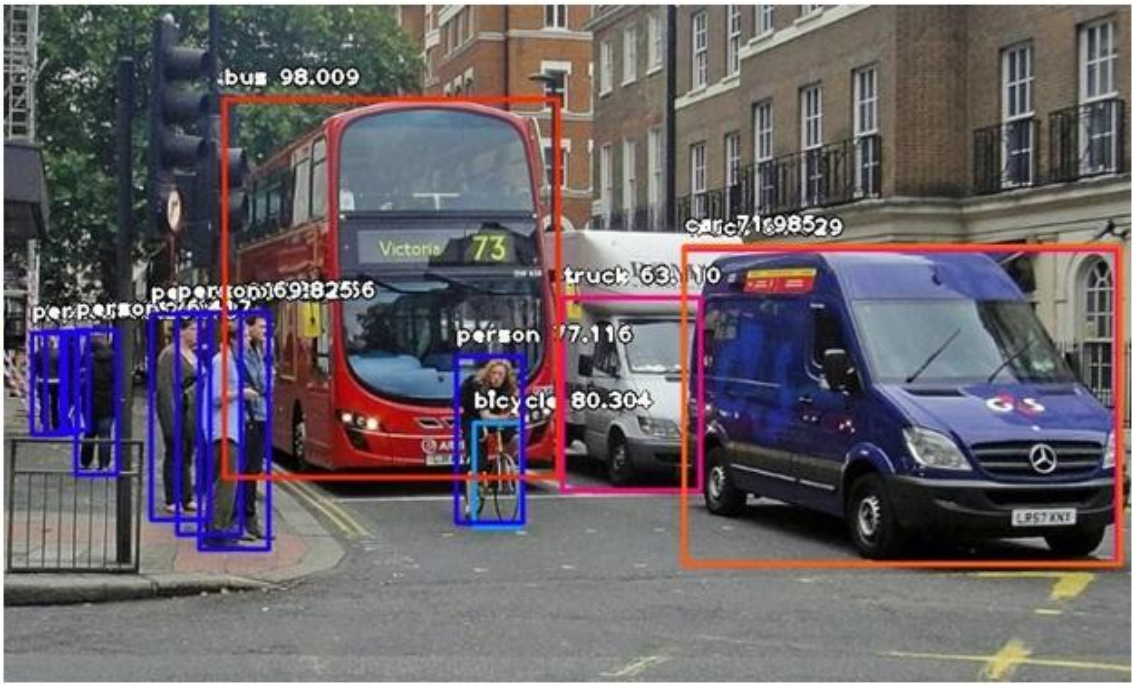

Figure 11: After Detection

ImageAI provides many more features useful for customization and production capable deployments for object detection tasks. Some of the features supported are:

1) Adjusting Minimum Probability: By default, objects detected with a probability percentage of less than 50 will not be shown or reported. You can increase this value for high certainty cases or reduce the value for cases where all possible objects are needed to be detected.
2) Custom Objects Detection: Using a provided CustomObject class, you can tell the detection class to report detections on one or a few number of unique objects.
3) Detection Speeds: You can reduce the time it takes to detect an image by setting the speed of detection speed to “fast”, “faster” and “fastest”.
4) Input Types: You can specify and parse in file path to an image, Numpy array or file stream of an image as the

input image

5) Output Types: You can specify that the detectObjectsFromImage function should return the image in the form of a file or Numpy array

Detection Speed:

Image AI now provides detection speeds for all object detection tasks. The detection speeds allow you to reduce the time of detection at a rate between 20%-80%, and yet having just slight changes but accurate detection results. Coupled with lowering the minimum-percentage-probability parameter, detections can match the normal speed and yet reduce detection time drastically. The available detection speeds are **"normal"** (default), **"fast"**, **"faster", "fastest"** and **"flash"**. All you need to do is to state the speed mode you desire when loading the modelin the code.

detector. loadModel (detection_speed="fast")

Hiding/Showing Object Name and Probability:

Image AI provides options to hide the name of objects detected and / or the percentage probability from being shown on the saved/returned detected image. Using the detect Objects From Image () and detect Custom Objects From Image () functions, the parameters display-object-name and display-percentage-probability can be set to True of False individually.

detections=detector.detectObjectsFromImage(input_image= os.path.join(execution_path,"image3.jpg", output_image_path=os.path.join(execution_path,"image3new_nodetails.jpg"), minimum_percentage_probability=30, display_percentage_probability=False, display_object_name=False)

4. Conclusion

By using this thesis and based on experimental results we are able to detect object more precisely and identify the objects individually with exact location of an object in the picture in x, y axis. This paper also provides experimental results on different methods for object detection and identification and compares each method for their efficiencies.

5. Future Enhancements

For Night time visual tracking, night vision mode should be available as an inbuilt feature in the CCTV camera.

To make the system fully automatic and also to overcome the above limitations, in future, multi-view tracking can be implemented using multiple cameras. Multi view tracking has the obvious advantage over single view tracking because of wide coverage range with different viewing angles for the objects to be tracked.

In this thesis, an effort has been made to develop an algorithm to provide the base for future applications such as listed below.

Acknowledgement

incere efforts have been taken by me. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I wish to express my deep sense of gratitude to my mentor, Dr. Nand Kumar for his able guidance and useful suggestions, which helped me in completing the project work, in time. He has been an immense source of inspiration. Thanks a lot for his encouragement in carrying out the project.

My thanks and appreciations also go to my colleagues in developing the project and people who have willingly helped me out with their abilities: Khushbu Bharti, 20PGCS02

References

[1] Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. IEEE Trans. Pattern Anal. Mach. Intell.26, 1475-1490. doi: 10.1109/TPAMI.2004.108

[2] Alexe, B., Deselaers, T., and Ferrari, V. (2010). "What is an object?, " in ComputerVision and Pattern Recognition (CVPR), 2010 IEEE Conference on (San Francisco, CA: IEEE), 73-80. doi: 10.1109/CVPR.2010.5540226

[3] Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. Int. J. Comput. Vis.1, 333-356. doi: 10.1007/BF00133571

[4] Andreopoulos, A., and Tsotsos, J. K. (2013).50 years of object recognition: direc-tions forward. Comput. Vis. Image Underst.117, 827-891. doi: 10.1016/j. cviu.2013.04.005

[5] Azizpour, H., and Laptev, I. (2012). "Object detection using strongly-superviseddeformable part models, " in Computer Vision-ECCV 2012 (Florence: Springer), 836-849.

[6] Azzopardi, G., and Petkov, N. (2013). Trainable cosfire filters for keypoint detectionand pattern recognition. IEEE Trans. Pattern Anal. Mach. Intell.35, 490- 503. doi: 10.1109/TPAMI.2012.106

[7] Azzopardi, G., and Petkov, N. (2014). Ventral-stream-like shape representation: from pixel intensity values to trainable object-selective cosfire models. Front. Comput. Neurosci.8: 80. doi: 10.3389/fncom.2014.00080

[8] Benbouzid, D., Busa-Fekete, R., and Kegl, B. (2012). "Fast classification using sparsedecision dags, " in Proceedings of the 29th International Conference on MachineLearning (ICML-12), ICML '12, eds

[9] J. Langford and J. Pineau (New York, NY: Omnipress), 951-958.

[10] Bengio, Y. (2012). "Deep learning of representations for unsupervised and transferlearning, " in ICML Unsupervised and Transfer Learning, Volume 27 of JMLRProceedings, eds I. Guyon, G. Dror,

[11] V. Lemaire, G. W. Taylor, and D. L. Silver (Bellevue: JMLR. Org), 17-36.

[12] Bourdev, L. D., Maji, S., Brox, T., and Malik, J. (2010). "Detecting peopleusing mutually consistent poselet activations, " in Computer Vision - ECCV2010 - 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-

[13] 11, 2010, Proceedings, Part VI, Volume 6316 of Lecture Notes inComputer Science, eds K. Daniilidis,

P. Maragos, and N. Paragios (Heraklion: Springer), 168- 181.

[14] Bourdev, L. D., and Malik, J. (2009). “Poselets: body part detectors trained using 3dhuman pose annotations, ” in IEEE 12th International Conference on ComputerVision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009 (Kyoto: IEEE), 1365-1372