A MINI PROJECT REPORT

On

"Implement visual surveillance applications and detect moving objects using object detection and tracking algorithm."

By

NAME OF STUDENTS	ROLL NO
Mayur Badgujar	405A010

Under the Guidance of **Prof. S. V. Patil**

BE. (COMPUTER ENGINEERING) 2022-23 (Semester II)

DEPARTMENT OF COMPUTER ENGINEERING,
SINHGAD COLLEGE OF ENGINEERING, VADGAON BK, PUNE

2022-23

ACKNOWLEDGEMENT

I would like to express my deepest appreciation to all those who provided me the possibility to complete this report. A special gratitude I give to **Prof. S. V. Patil** whose contribution in stimulating suggestions and encouragement, helped me to coordinate my project especially in writing this report.

Furthermore I would also like to acknowledge with much appreciation the crucial role **Prof. S. D. Lokhande** Principal of Sinhgad College of Engineering, Vadgaon and **Prof. M. P. Wankhade** Head of Department, who support to complete the project “Kid’s interactive Learning Platform Using Artificial Intelligence”.

Last but not least, many thanks go to all the teachers who have invested their full effort inguiding the team in achieving the goal. I have to appreciate the guidance given by other supervisor as well as the panels especially in our project presentation that has improved our presentation skills thanks to their comment and advices.

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.

So, we can distinguish between these three computer vision tasks with this example:
Image Classification: This is done by Predict the type or class of an object in an image. Input: An image which consists of a single object, such as a photograph.
Output: A class label (e.g. one or more integers that are mapped to class labels).

Object Localization: This is done through, Locate the presence of objects in an image and indicate their location with a bounding box.
Input: An image which consists of one or more objects, such as a photograph.
Output: One or more bounding boxes (e.g. defined by a point, width, and height).
Object Detection: This is done through, Locate the presence of objects with a bounding box and types or classes of the located objects in an image.
Image classification also involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is always more challenging and combines these two tasks and draws a bounding box around each object of interest in the image and assigns them a class label. Together, all these problems are referred to as object recognition.
Object recognition refers to a collection of related tasks for identifying objects in digital photographs. Region-based Convolutional Neural Networks, or R-CNNs, is a family of techniques for addressing object localization and recognition tasks, designed for model performance. You Only Look Once, or YOLO is known as the second family of techniques for object recognition designed for speed and real-time use.

2. Background :

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

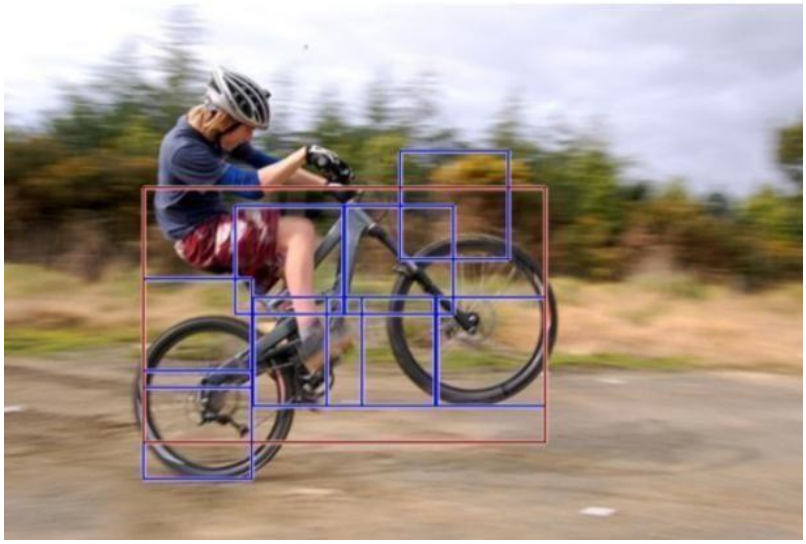


Figure 1

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 .

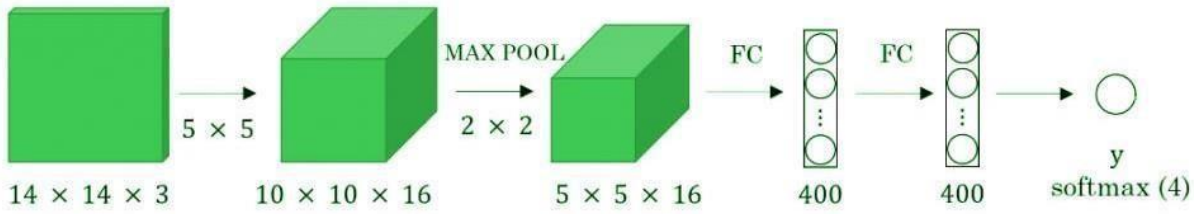


Figure 2: simple convolution network

3. LITERATURE SURVEY :

In various fields, there is a necessity to detect the target object and also track them effectively while handling occlusions and other included complexities. Many researchers (Almeida and Guting 2004, Hsiao-Ping Tsai 2011, Nicolas Papadakis and Aure lie Bugeau 2010) attempted for various approaches in object tracking. The nature of the techniques largely depends on the application domain. Some of the research works which made the evolution to proposed work in the field of object tracking are depicted as follows

3.1 OBJECT DETECTION

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 singleobject 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.

3.1.1 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, a subsampling 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.

3.1.2 Template Matching

Template Matching is the technique of finding small parts of an image which match a template image. It slides the template from the top left to the bottom right of the image and compares for the best match with the template. The template dimension should be equal to the reference image or smaller than the reference image. It recognizes the segment with the highest correlation as the target. Given an image S and an image T, where the dimension of S was both larger than T, output whether S contains a subset image I where I and T are suitably similar in pattern and if such I exists, output the location of I in S as in Hager and Bellhumear (1998). Schweitzer et al (2011), derived an algorithm which used both upper and lowers bound to detect 'k' best matches. Euclidean distance and Walsh transform kernels are used to calculate match measure. The positive things included the usage of priority queue improved quality of decision as to which bound-improved and when good matches exist inherent cost was dominant and it improved performance. But there were constraints like the absence of good matches that lead to queue cost and the arithmetic operation cost was higher. The proposed methods dint use queue thereby avoiding the queue cost rather used template matching. Visual tracking methods can be roughly categorized in two ways namely, the feature-based and region-based method as proposed by Ken Ito and Shigeyuki Sakane (2001). The feature-based approach estimates the 3D pose of a target object to fit the image features the edges, given a 3D geometrical model of an object. This method requires much computational cost. Region-based can be classified into two categories namely, parametric method and view-based method. The parametric method assumes a parametric model of the images in the target image and calculates optimal fitting of the model to pixel data in a region. The view-based method was used to find the best match of a region in a search area given the reference template. This has the advantage that it does not require much computational complexity as in the feature-based approach

4. Existing Methods:

4.1 ResNet

To train the network model in a more effective manner, we herein adopt the same strategy as that used for DSSD (the performance of the residual network is better than that of the VGG network). The goal is to improve accuracy. However, the first implemented for the modification was the replacement of the VGG network which is used in the original SSD with ResNet. We will also add a series of

convolution feature layers at the end of the underlying network. These feature layers will gradually be reduced in size that allowed prediction of the detection results on multiple scales. When the input size is given as 300 and 320, although the ResNet–101 layer is deeper than the VGG–16 layer, it is experimentally known that it replaces the SSD's underlying convolution network with a residual network, and it does not improve its accuracy but rather decreases it.

4.2 R-CNN

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

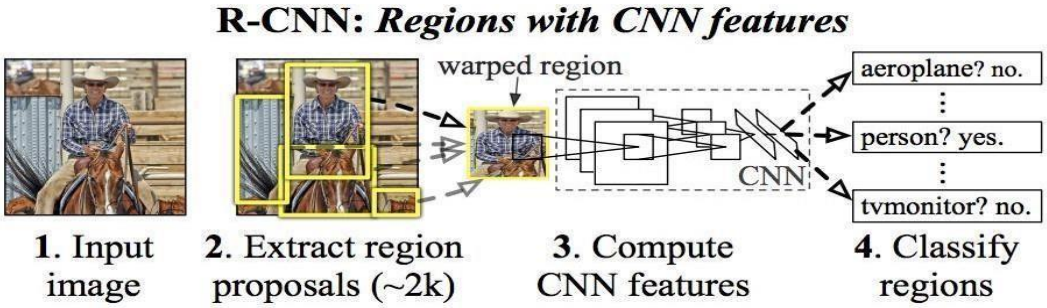


Figure 3 R-CNN

These 2000 candidate regions which are proposals are warped into a square and fed into a convolutional neural network that produces a 4096-dimensional feature vector as output. The CNN plays a role of feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into an SVM for the classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values for increasing the precision of the bounding box. For example, given the region proposal, the algorithm might have predicted the presence of a person but the face of that person within that region proposal could have been cut in half. Therefore, the offset values which is given help in adjusting the bounding box of the region proposal.

4.1.1Problems with R-CNN

- It still takes a huge amount of time to train the network as you would have to classify 2000 region proposals per image.
- It cannot be implemented real time as it takes around 47 seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

4.3 Fast R-CNN

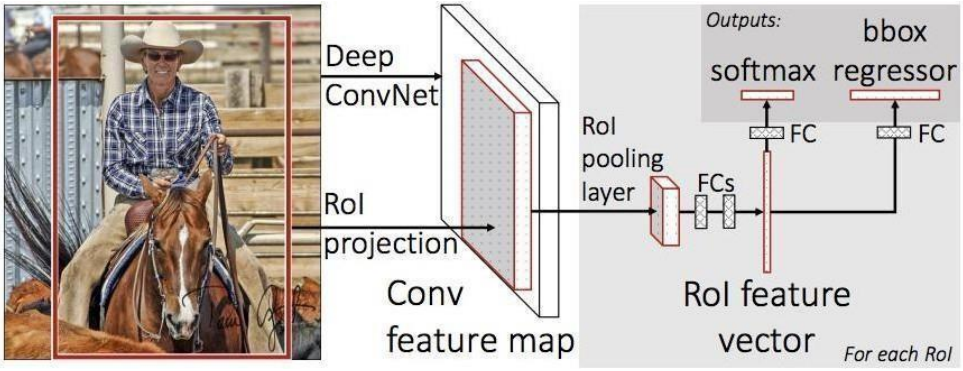


Figure 4: Fast R-CNN

The same author of the previous paper(R-CNN) solved some of the drawbacks of R-CNN to build a faster object detection algorithm and it was called Fast R-CNN. The approach is similar to the RCNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map, we can identify the region of the proposals and warp them into the squares and by using an RoI pooling layer we reshape them into the fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, we can use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.

The reason "Fast R-CNN" is faster than R-CNN is because you don't have to feed 2000 region proposals to the convolutional neural network every time. Instead, the convolution operation is always done only once per image and a feature map is generated from it.

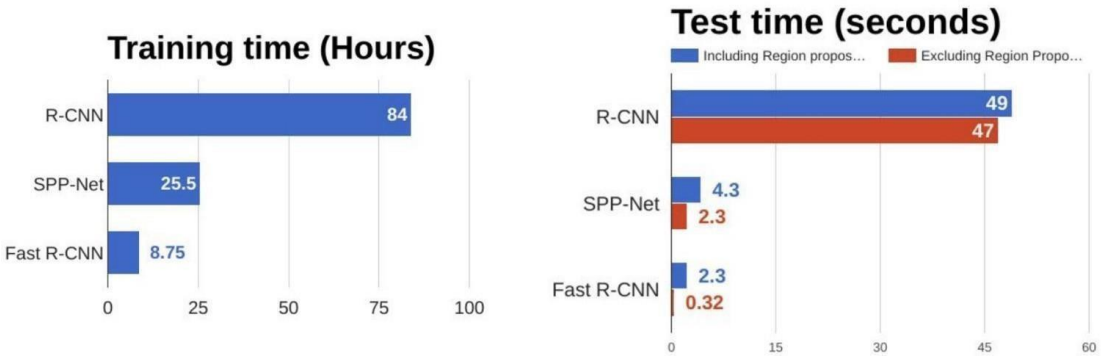


Figure 5: Comparison of object detection algorithms

From the above graphs, you can infer that Fast R-CNN is significantly faster in training and testing sessions over R-CNN. When you look at the performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when compared to not using region proposals. Therefore, the region which is proposals become bottlenecks in Fast R-CNN algorithm affecting its performance.

4.4 Faster R-CNN

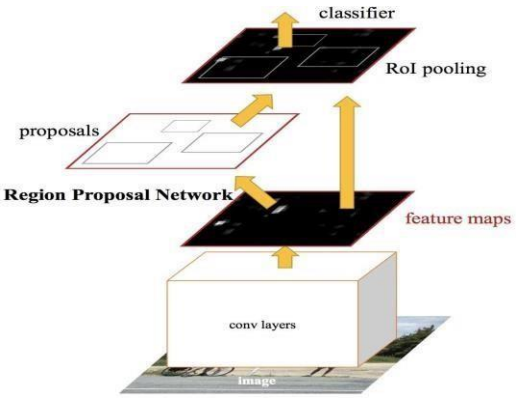
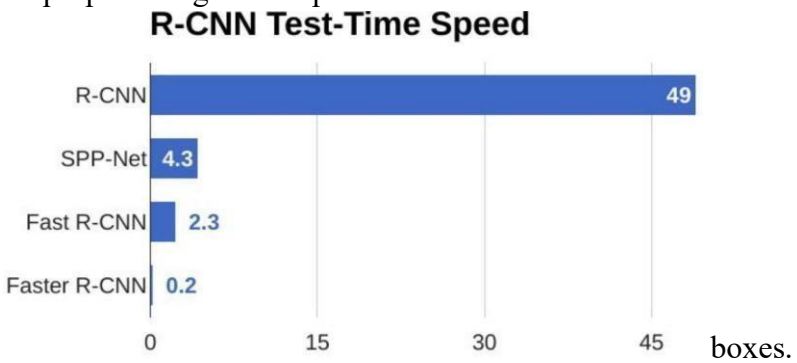


Figure 6: Faster R-CNN

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

Figure7: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.

4.5 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.

5.
SYSTEM REQUIREMENT:

Install Python on your computer system

1. Install ImageAI and its dependencies like tensorflow, Numpy,OpenCV, etc.

2. Download the Object Detection model file(Retinanet)

5.1 Steps to be followed :-

1)Download and install Python version 3 from official Python Language website *https://python.org*

2)Install the following dependencies via pip: **i. Tensorflow:**

Tensorflow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is an symbolic math library, and is also used for machine learning application such as neural networks,etc.. It is used for both research and production by Google.

Tensorflow is developed by the Google Brain team for internal Google use. It is released under the Apache License 2.0 on November 9,2015.Tensorflow computations are expressed as stateful dataflow graphs. The name Tensorflow derives from operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

pip install tensorflow -command **ii. Numpy:**

NumPy is library of Python programming language, adding support for large, multi-dimensional array and matrice, along with large collection of high-level mathematical function to operate over these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several developers. In 2005 Travis Olphant created NumPy by incorporating features of computing Numarray into Numeric, with extension modifications. NumPy is open-source software and has many contributors.

iii. OpenCV:

OpenCV is an library of programming functions mainly aimed on real time computer vision. originally developed by Intel, it is later supported by Willow Garage then Itseez. The library is a cross-platform and free to use under the open-source BSD license.

pip install h5py

i. Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. **ii. ImageAI:**

ImageAI provides API to recognize 1000 different objects in a picture using pre-trained models that were trained on the ImageNet-1000 dataset. The model implementations provided are SqueezeNet, ResNet, InceptionV3 and DenseNet.

6.

METHODOLOGY:

6.1 SqueezeNet:

SqueezeNet is name of a DNN for computer vision. SqueezNet is developed by researchers at DeepScale, University of California, Berkeley, and Stanford University together. In SqueezeNet design, the authors goal is to create a smaller neural network with few parameters that can more easily fit into memory of computer and can more easily be transmitted over a computer network.

SqueezeNet is originally released in 2016. This original version of SqueezeNet was implemented on top of the Caffe deep learning software framework.The open-source research community ported SqueezeNet to a number of other deep learning frameworks. And is released in additions,in 2016, Eddie Bell released a part of SqueezeNet for the Chainer deep learning framework. in 2016, Guo Haria released a part of SqueezeNet for the Apache MXNet framework. 2016, Tammy Yang released a port of SqueezeNet for the Keras framework. In 2017, companies including Baidu, Xilnx, Imagination Technologies, and Synopsys demostrated SqueezedNet running on low-power processing platforms such as smartphones, FPGAs, and customprocessors.

SqueezeNet ships as part of the source code of a number of deep learning frameworks such as PyTorch, Apache MXNet, and Apple CoreML.In addition, 3rd party developers have created implementation of SqueezeNet that are compatible with frameworks such as TensorFlow. Below is summary of frameworks that support SqueezeNet.

DNN Model	Application	Original Implementation	Other Implementations
SqueezeDet	Object Detection on Images	TensorFlow	Caffe, Keras
SqueezeSeg	Semantic Segmentation of	TensorFlow	
SqueezeNext	Image Classificat ion	Caffe	TensorFlow, Keras, PyTorch
SqueezeNAS	Neural Architect ure Search for Semantic Segmentation	PyTorch	

7.

RESULTS AND DISCUSSION:

Figure 8: 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.

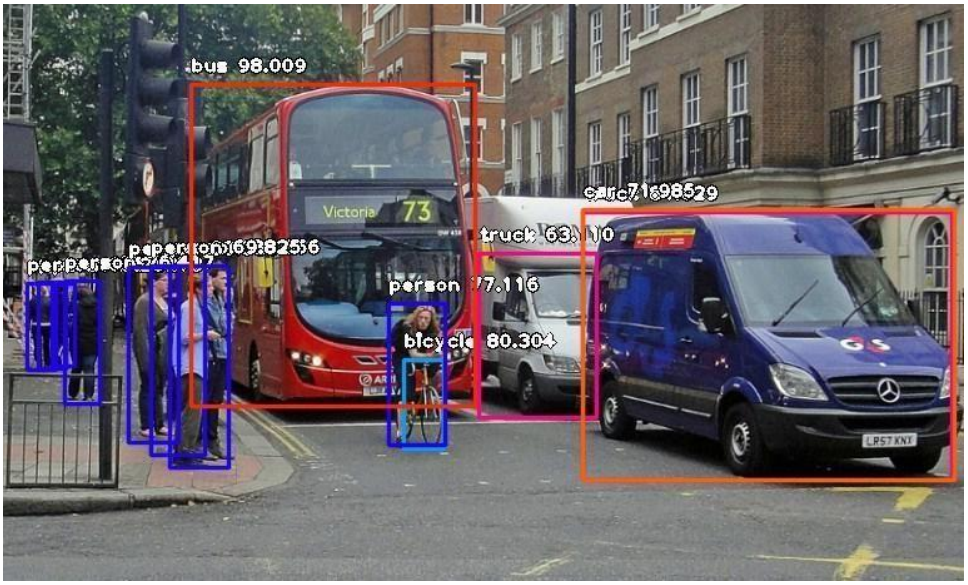


Figure 9: After Detection

As expected our algorithm identifies the objects by its classes ans assigns each object by its tag and has dimensions on detected image.

8.

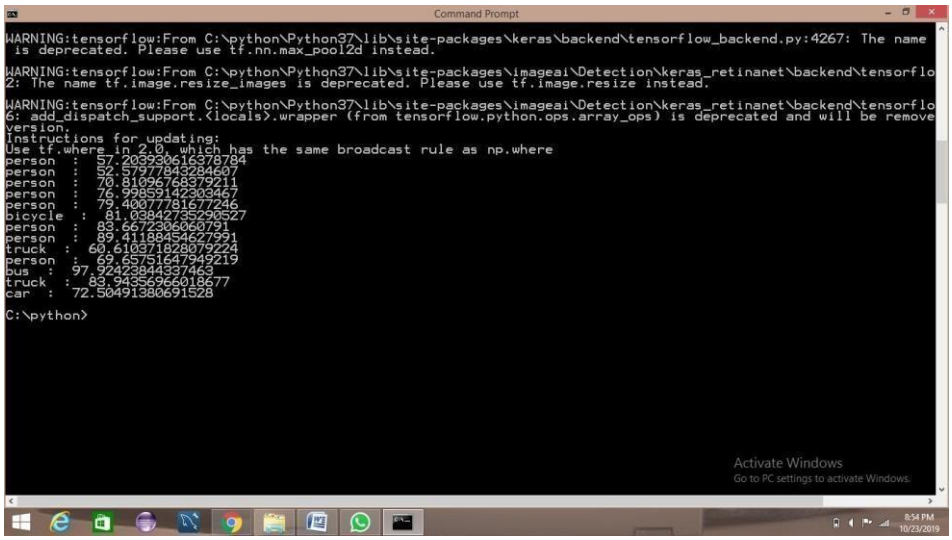


Figure 10: ***Console result for above image***

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

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

7.1 Detection Speed

ImageAI 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

ImageAI 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

detectObjectsFromImage() and detectCustomObjectsFromImage() 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)
)
```

8. CONCLUSION:

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

9. 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 J. Langford and J. Pineau (New York, NY:Omnipress), 951–958.
9. 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, V. Lemaire, G. W. Taylor, and D. L. Silver(Bellevue: JMLR.Org), 17–36.
10. 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 5-11, 2010, Proceedings, Part VI, Volume 6316 of Lecture

Notes inComputer Science, eds K. Daniilidis, P. Maragos, and N. Paragios (Heraklion:Springer), 168–181.