ECN-343 Embedded Systems Report

Design and Implementation of an Embedded System for Object Detection

Under the guidance of

Dr. Brajesh Kumar Kaushik

Associate Professor, Dept. of Electronics & Comm.
Indian Institute of Technology Roorkee

Submitted by:

Jaskirat Singh (16116025)
Kaustubh Nayyar (16116030)
Rishabh Dhakarwal (16116052)
Shivanshu Raj Shrivastava (16116063)

(Students of B.Tech. in Electronics and Communication, IIT-Roorkee)

Dated: November 29, 2018

ACKNOWLEDGEMENT

In the accomplishment of this project successfully, many people have bestowed upon us their blessings and heart pledged support, this time we are utilizing to thank all the people who have been concerned with this project.

Primarily, we would like to thank God for providing us the strength to complete this project with success. We would like to thank **Dr. Brajesh Kumar Kaushik,** who have given us this opportunity and whose valuable guidance has helped us to complete this project and make it full proof success. His suggestions and instructions have served as the major contributor towards the completion of the project.

We would like to thank our team members and friends who have helped us with their valuable suggestions and guidance that have helped us to successfully finish this project in time.

CERTIFICATE

This is to certify that the Project entitled **"Design and Implementation of Embedded System for Object Detection"** is the bonafide work of **Mr. Jaskirat Singh, Mr. Kaustubh Nayyar, Mr. Rishabh Dhakarwal and Mr. Shivanshu Raj Shrivastava** done in the autumn semester of academic year **2018-2019**. They have duly completed their project and have fulfilled all the requirements of the course **ECN-343, Embedded Systems**, to my satisfaction.

Dr. Brajesh Kumar Kaushik

Associate Professor

Dept. of Electronics & Communication

Indian Institute of Technology Roorkee

Date: November 29, 2018

Place: IIT Roorkee

CONTENTS

I. Visual Perception

The ability to interpret the surrounding environment using light in the visible spectrum reflected by the objects in the Environment is known as **Visual Perception.** It is divided into 4 major categories (as shown in Fig. 1):

- **Classification**

In classification, we assign a label to an entire image, for example in Fig. 1, we know the image consists of a cat, hence, classify the entire image as a "CAT".

- **Localization**

In Localization, we assign a bounding box to a particular label in an image. In Fig. 1, we draw a bounding box around the particular object (cat in this case) and then classify that box as "CAT". It can be seen that **Classification and Localization** are for **single objects only**.

- **Object Detection**

In Object Detection, we draw multiple bounding boxes in an image and categorize these boxes as objects and assign them a label. It can be thought of as Localization + Classification for multiple objects.

- **Image Segmentation**

In Image Segmentation, we create precise segments of where objects lie in an image. In Fig. 1, the difference between Object Detection and Image Segmentation can clearly be seen.

In this project, we will only concentrate on Object Detection.

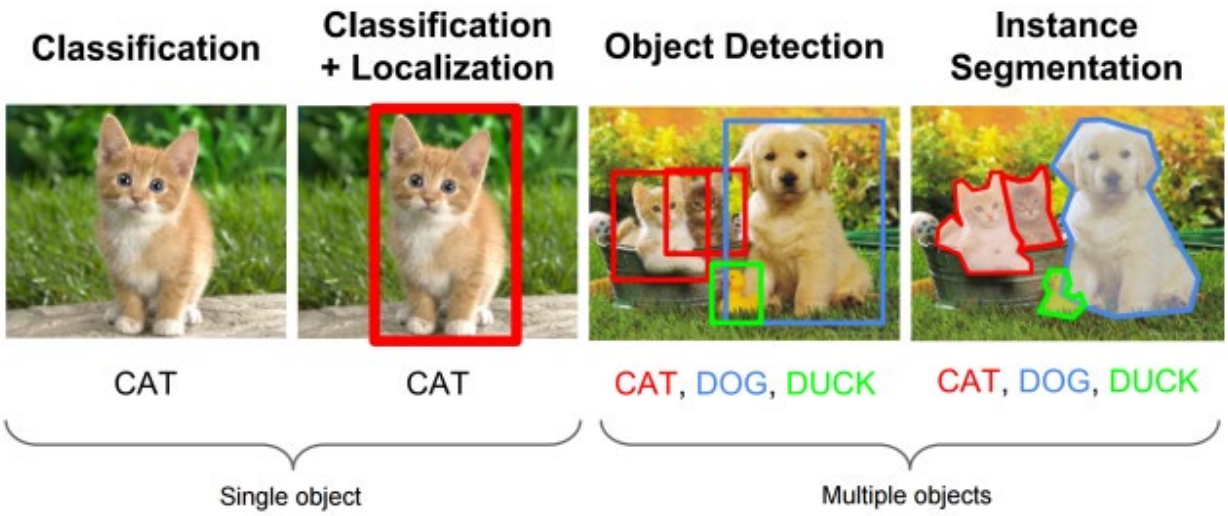


Fig. 1: Visual Perception and it's categories [1]

II. Object Detection

As explained in the previous section, Object Detection refers to detecting instances of objects from a particular class in an image. The goal of object detection is to detect all instances of objects from a known class, such as people, cars or faces in an image.

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found. [2]

There are a lot of Applications of Object Detection, few of which are discussed in the next section.

a. Applications of Object Detection

❖ **Autonomous Ground Vehicles** [3]

Modern Autonomous Car Giants like Tesla, Google, Mercedes-Benz, BMW, Audi, etc. have one or another algorithm for vehicle and obstructions detection. These autonomous vehicles are replacing a human's perception with an embedded system that takes live stream feed from a camera, processes it with object detection software and then gives an output to the driving software. It is obvious that such systems have to be lightning fast in decision making as well as processing. With speed, accuracy is of utmost importance because human lives are involved.

Fig. 2: Tesla Vehicle Detection System [4]

❖ Manufacturing Industry and Anomaly Detection

When the manufacturing industry is expanded and production is scaled to thousands and millions per day, humans are not enough to detect faults and errors in every single product. Hence, systems with object and anomaly detection were installed to replace humans. Such systems have incredibly fast response times and highly accurate.

❖ Online Image Classification

Today in the era of internet, there are several search engine giants like Google, Bing, Yahoo etc. who offer image search. There are an estimated trillions of images on the internet [5]. But how does the Google know what images to show us in results when we search "Chip Diagram" in Google images? A simple answer is Object detection and classification on all these images.

❖ Security Applications

There are a few security companies that provide security solutions with embedded systems that use object detection and tracking to detect presence of any intruder in the building/room.

❖ Face Detection

Everyone these days seem to use "filters" and "Face Locks" in various applications in their smart phones. All of them use Face Detection which is advanced version of object detection and tracking.

There are countless other applications of object detection such as Object Tracking, Counting Number of People, etc. used in everyday lives for commercial as well as research purposes.

III. Details of the Problem Statement

The exact Problem Statement is: **"Designing and implementing an embedded system for Real-Time Object Detection using a live video-feed from a camera interfaced with a raspberry pi"**

Hence, we developed an embedded system which works in the following way:

a. Feeding the live feed with camera

To feed in the live stream videos or images, we used a Logitech C310 Camera.

To save ourselves from the tiring task of holding the camera steady, we developed a separate automation system. This system consisted of a servo motor, raspberry pie and the camera itself. The camera was mounted on a servo-motor which will be controlled through an Arduino having a customized code for slow and steady movements of camera.

Earlier the video stream and images used for testing our embedded system was streamed to the laptop for storage. Further, the same setup was used for live-stream and real time object detection directly through pi itself.

Fig. 3: Camera Setup for feeding live stream to the Pi

b. SSD algorithm for object detection

Now, comes the tough part of processing our live stream video through a Neural Network modelled by **SSD Algorithm.**

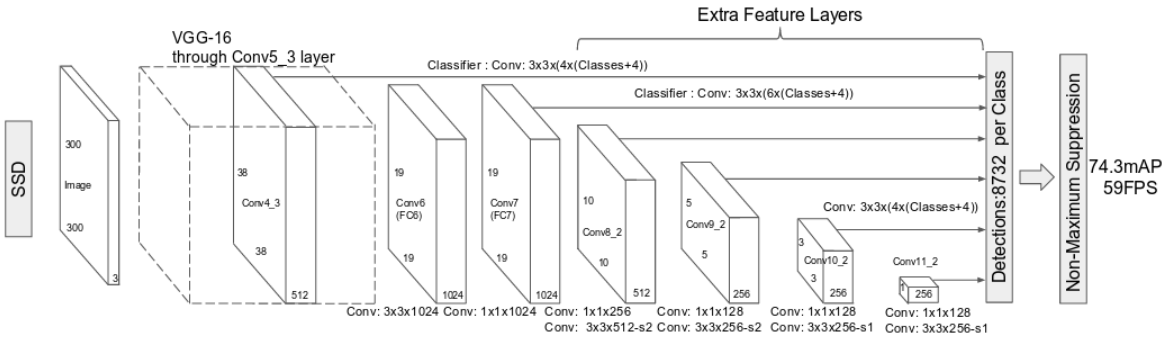


Fig. 4: Flowchart of a Neural Network implementing SSD Algorithm [6]

The key idea here is a single network (to account for improvements in speed) and there is no need for region proposals ALA faster R-CNN. Instead the idea is to use different bounding boxes and then adjust the bounding boxes as a part of its prediction. Different bounding box predictions is achieved by each of the last few layers of the network responsible for predictions for progressively smaller bounding box and final prediction is a union of all these predictions.

Model:

The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression (NMS) step to produce the final output detections (for bounding boxes with most overlap keep the one with highest score).

The key difference between training an SSD and training a typical detector that uses region proposals, is that ground truth information needs to be assigned to specific outputs in the fixed set of detector outputs. Training also involves choosing the set of default boxes and scales for detection as well as the hard-negative mining and data augmentation strategies which is similar to the strategy implemented in YOLO.

Hard negative mining:

After the matching step, most of the default boxes are negatives, especially when the number of possible default boxes is large. This introduces a significant imbalance between the positive and negative training examples which might lead to superfluous results that give out wrong results and in essence lead to wrong inferences. Instead of using all the negative examples, we sort them using the highest confidence loss for each default box and pick the top ones so that the ratio between the negatives and positives is at most 3:1. We found that this leads to faster optimization and a more stable training, since this arrangement gives effect to a much stable training regime.

Working of the Raspberry Pi using SSD:

When we get input video from the camera, we separate each frame and send it to an image resizer that resizes it to 300x300 pixels. This image is then fed to the SSD Algorithm which predicts the probability of detecting any object. Further, it sends it into another module which finally draws a bounding box around the detected object (the object with probability more than 0.23). Each of the processed frame is then fed to output stream module.

c. Output Stream

Now that we have bounding boxes with corresponding object classification probability result in each frame, all these frames are collected and played as our output video stream with objects detected and classified.

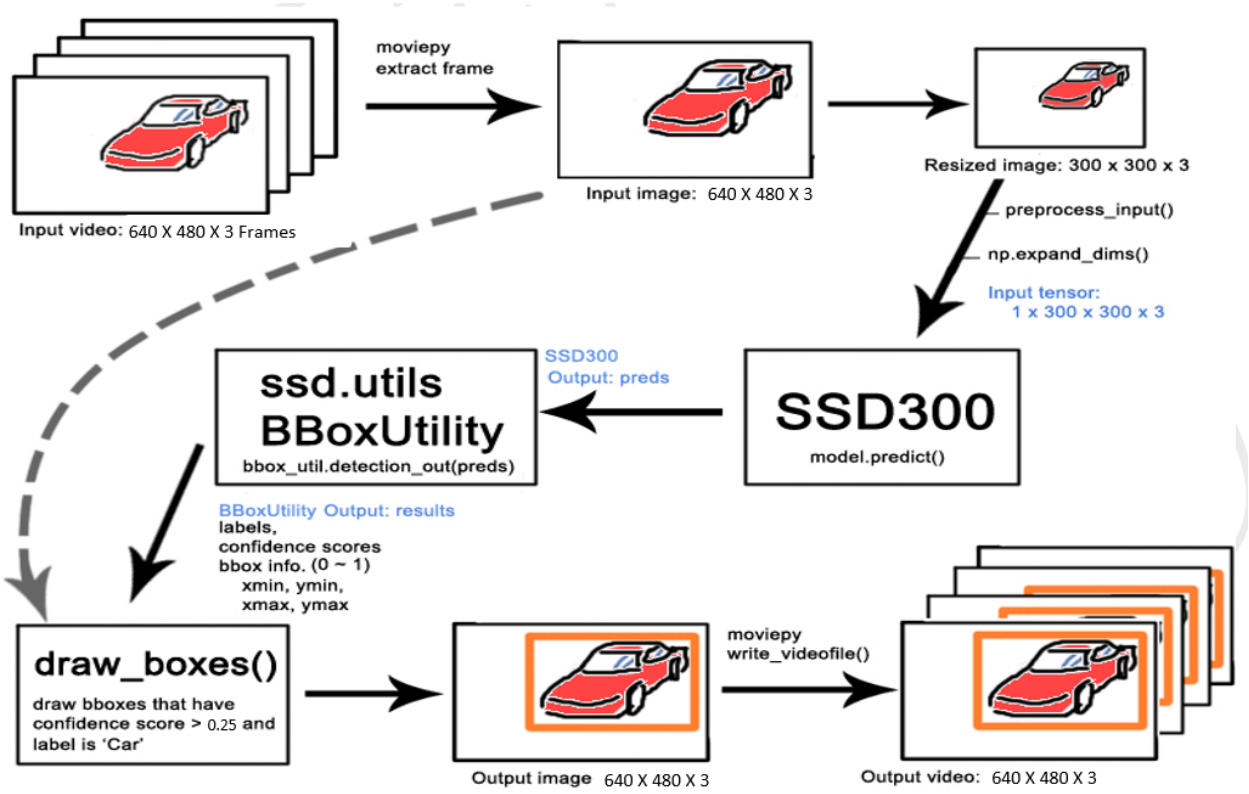


Fig. 5: Flowchart of a working of the code in Raspberry pi [7]

IV. Why did we choose SSD?

There are three well known model architectures for the fulfillment of our objective, which is object detection in live video stream using a raspberry pi:

a. Faster R-CNN

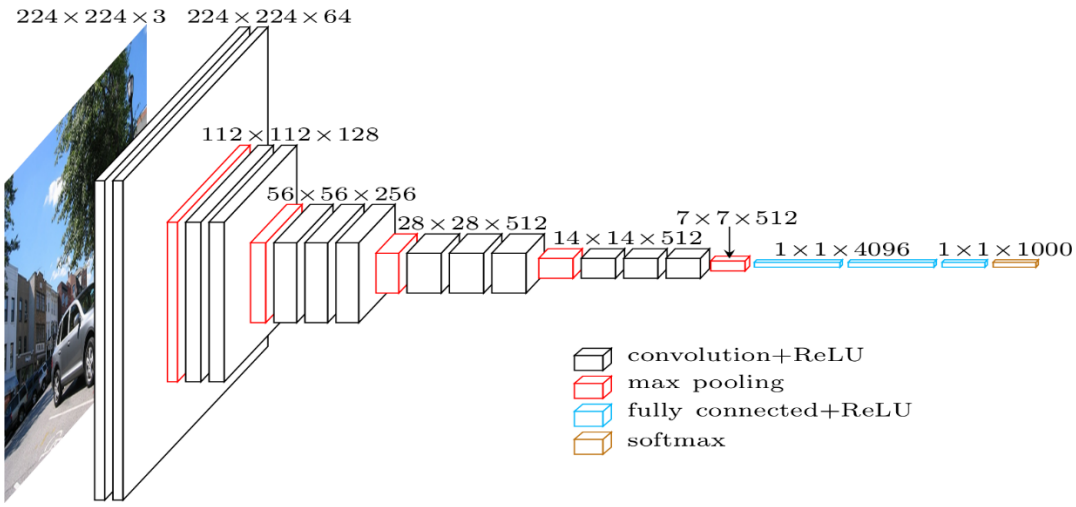


Fig. 6: Flowchart of Faster R-CNN Neural Network [8]

This is an improved version of the conventional R-CNN architecture. The main difference between the conventional R-CNN architecture and the faster R-CNN architecture is that the latter network proposed an RPN (Regional Proposal Network) [8] to detect those locations of the input image which have a high probability for the presence of an object associated with the desired classes. This was done to solve the limitations imposed by the Selective search algorithm in conventional R-CNN architecture.

Despite of the very high accuracy offered, we didn't choose this architecture due to the fact that this technique is quite slow and doesn't operate in real time.

b. You Only Look Once (YOLO)

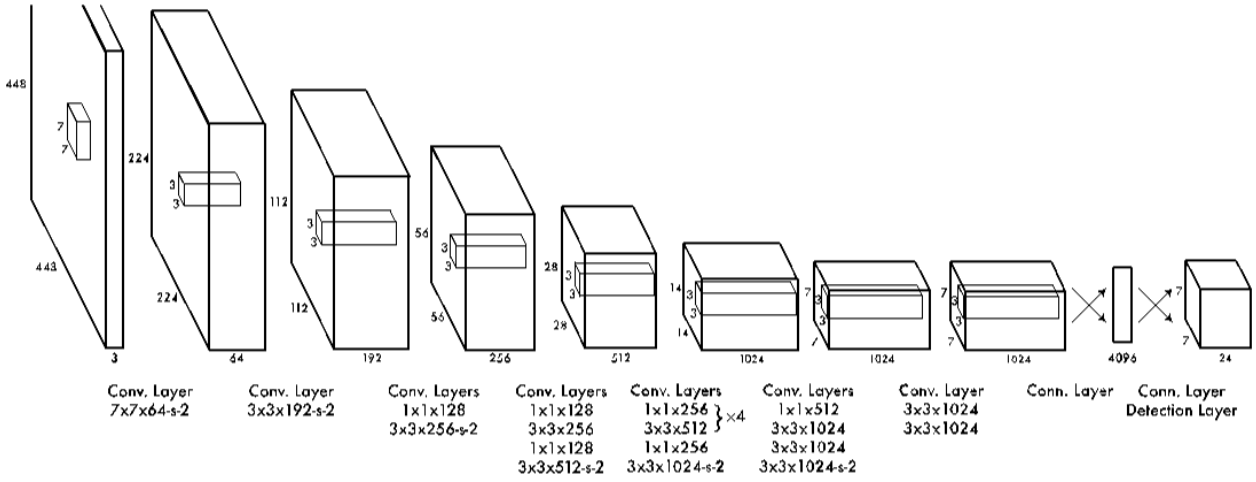


Fig. 7: Flowchart of YOLO Neural Network [9]

As quoted directly from paper [9], "A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. First, YOLO is extremely fast. Since we frame detection as a regression problem, we don't need a complex pipeline. We simply run our neural network on a new image at test time to predict detections. Our base network runs at 45 frames per second."

This network offers lesser accuracy compared to faster R-CNN but can be applied in applications involving real time detection due to its higher speed. But, this technique involves some fully connected layers of pre-fixed sizes, this architecture is limited to using only fixed scaled images which severely limited the extensibility of this network.

c. Single Shot Detection (SSD)

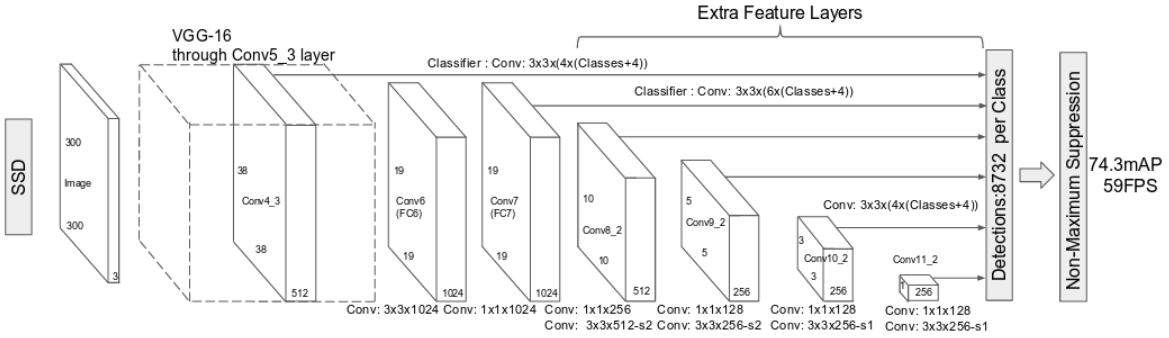


Fig. 8: Flowchart of a Neural Network implementing SSD Algorithm [6]

As quoted directly from the paper [6], "SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape. Additionally, the network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes."

This architecture can thus be extended to the cases where we have images of different scales. Also, this method doesn't involve the use of fully connected layers and is quite fast and suitable for real time detection. The accuracy offered by this method is not as high as offered by faster R-CNN, but better as well as faster than YOLO.

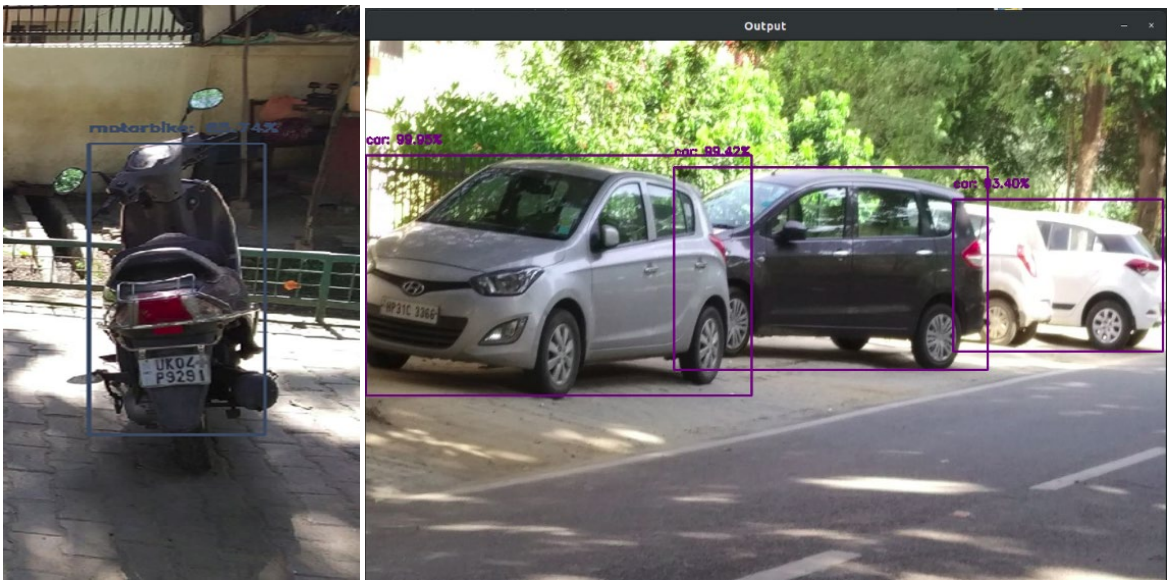
V. What we have Achieved

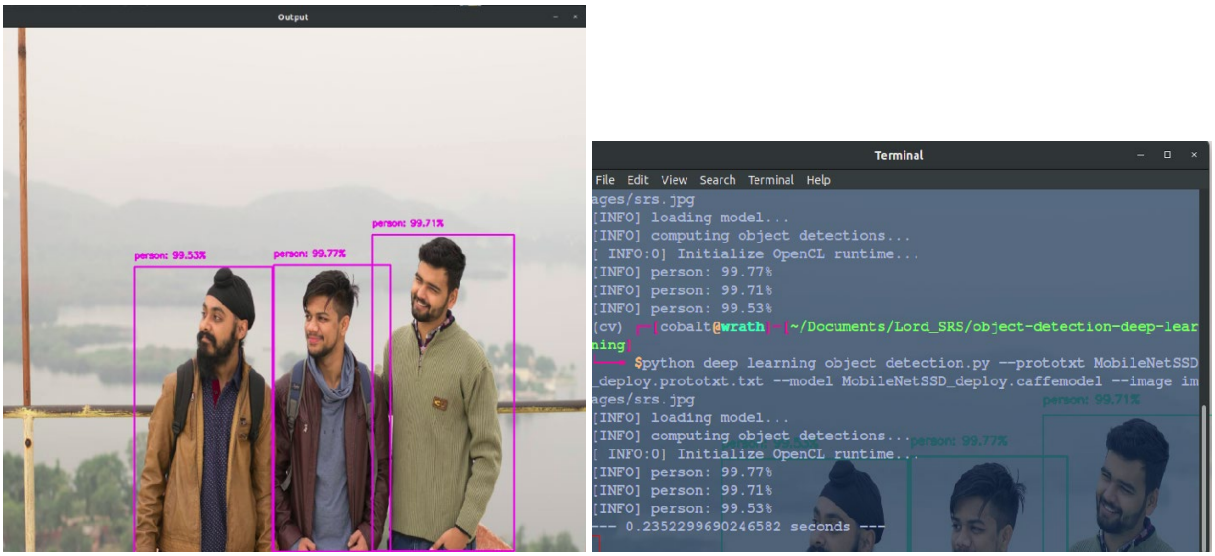
We used the setup as explained in Section III. We successfully implemented object detection system for live stream videos. We used Raspberry pi 3E for all kinds of processing, but all the setup was tested on a Laptop first.

We successfully integrated a pre-trained model with our hardware (Raspberry Pi 3E) which is able to detect the following objects:

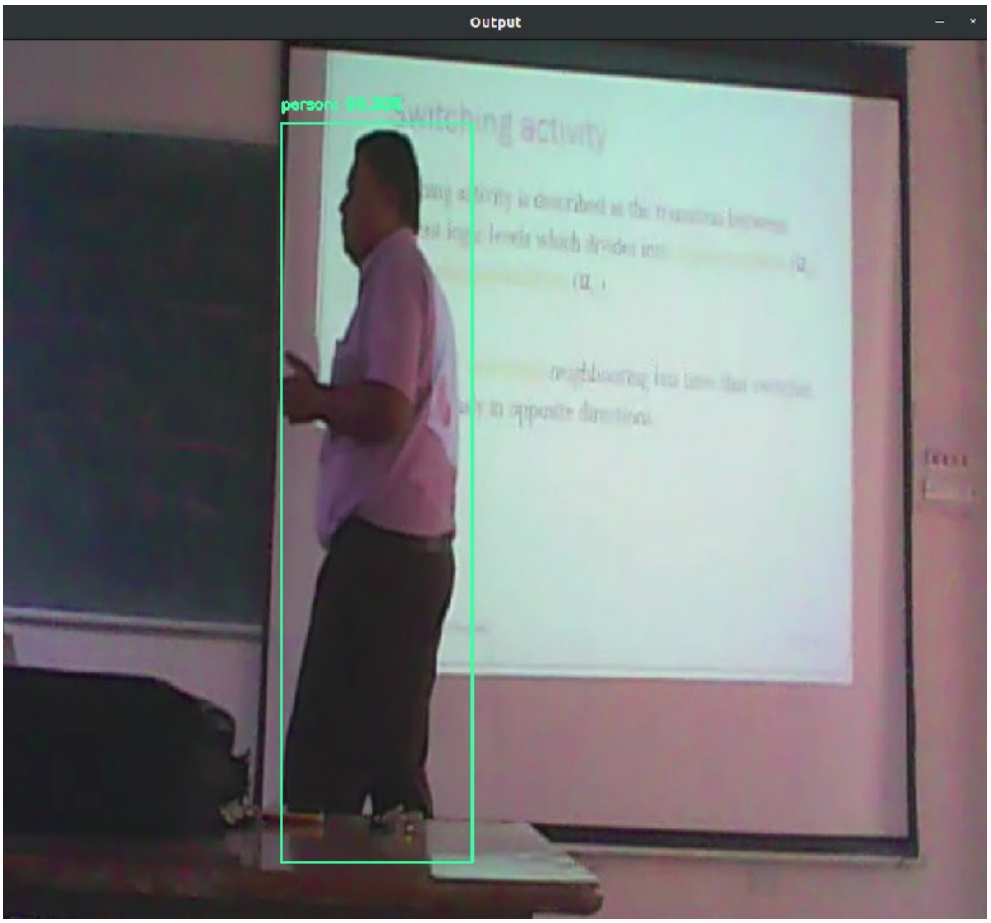
- ✓ Background
- ✓ Humans
- ✓ Cows
- ✓ Table
- ✓ Sofa
- ✓ Dog
- ✓ Horse
- ✓ Potted Plants
- ✓ Buses
- ✓ Television
- ✓ Sheep
- ✓ Boat
- ✓ Aero-planes
- ✓ Bird
- ✓ Vehicles (Cars, Motorbikes)
- ✓ Cats
- ✓ Bottle
- ✓ Trains

Here are a few images obtained from our testing on a raspberry pi using still image object detection as well as live stream video object detection:





Clearly, we can see that the total processing time of a **still image in Raspberry Pi** is about **0.235 seconds.** This timing is considered crucial in case of live stream video processing.



The above picture is a screenshot from live video streaming. We achieved a **frame rate** of about **30 fps in a laptop** and slightly less than **1 fps in the Raspberry Pi 3E.**

VI. Scope for Further Work

There are still a lot of things we can work on to improve our embedded system like:

- ❖ Expand our database to detect more classes of objects
- ❖ Enhance our algorithm of neural network to be more efficient and fast to reduce the input to output stream delay
- ❖ Discuss the possibility of making a custom system-on-chip performing the task using minimum power

VII. References

[1]https://medium.com/nanonets/how-to-easily-detect-objects-with-deep-learning-on-raspberrypi

[2] https://en.wikipedia.org/wiki/Object_detection

[3]www.researchgate.net/publication/283757446_Autonomous_Cars_Past_Present_and_Future_-_A_Review_of_the_Developments_in_the_Last_Century_the_Present_Scenario_and_the_Expected_Future_of_Autonomous_Vehicle_Technology

[4] Tesla Inc., https://www.tesla.com/

[5]https://www.theatlantic.com/technology/archive/2015/11/how-many-photographs-of-you-are-out-there-in-the-world/413389/

[6] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector." In *European conference on computer vision*, pp. 21-37. Springer, Cham, 2016.

[7] CSN-231 Convolutional Neural Networks for Visual Recognition, Stanford University

[8] Girshick, Ross. "Fast r-cnn." In *Proceedings of the IEEE international conference on computer vision*, pp. 1440-1448. 2015.

[9] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).