Group 16: Real-time Localization, Detection & Classification of Objects

Aman Verma
Department of Computer Science
Arizona State University
1211170659

Anchit Agarwal
Department of Computer Science
Arizona State University
1211213390

Siddhant Prakash
Department of Computer Science
Arizona State University
1211092724

Abstract

We propose to implement an algorithm to localize, detect and classify objects in real time using deep convolutional networks along with ROS compatible modules. Further we train our neural network model to drive a car in a simulated environment such as game. This can be the first step in training an artificial intelligence to play a computer game.

1 Introduction

Computer Vision has always dealt with the task of object localization, detection and classification. Each of the problem mentioned is a subclass of the previous one with localization being the most challenging amongst the three. We have seen many algorithms [1] [4] for performing individual tasks but methods which perform well in all three tasks are a scarcity. The first real advancement in simultaneously solving these problems came with the advent of deep neural networks in the form of convolutional networks. "OverFeat" [3] is a feature extractor built using convolutional networks which was shown to perform state-of-the-art performance for localization and detection tasks along with achieving competitive results in classification task. The challenge now remains to perform these or a subset of these tasks simultaneously in real time. "YOLO" [2] was a research work published in CVPR'16 which achieves the real time detection of objects. The complete system overview of our proposed system can be seen in Figure 1

2 Approach

In our project we propose a visual recognition system which utilizes a neural network framework for the localization and detection of objects. We first pass the scene captured via a camera sensor through an explicit neural network and we get bounding boxes, labels of object and the co-ordinates of the objects as an output of the localization, detection and classification of the video frame. We then use the obtained results to perform application specific tasks. We deployed our algorithm on an autonomous car driving simulator to detect the object in real time while the vehicle is driven, and make decision on acceleration or deceleration based on the detection of object in the field of view of the driver.

We start by explaining our neural network framework first. We explain its architecture and the network design decisions which makes it an efficient system enough for real time localization and

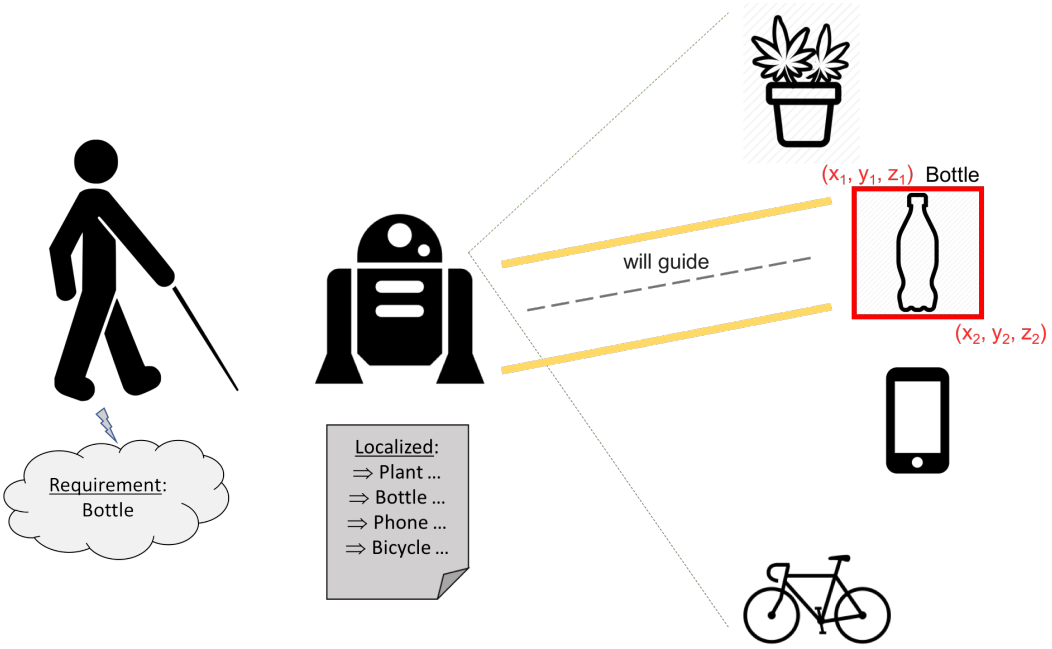

Figure 1: System overview

detection task. Then we look at the implementation we get using perception modules linked with the neural network architecture. We faced lots of latency issues due to the external nature of the neural network modules. And finally we explain the deployment of the system on the autonomous driving system.

2.1 Neural Network Framework

In our implementation, the neural network is posed as a regression problem. It means, that running the network on a test set directly gives us the predictions of the detections. This contributes to the speed of our neural network for object detection and thus leading to real time object localization. Secondly, the predictions are made considering the global image, during both training and test time, rather than just parts of the image, for example in sliding window based approaches. This helps gives a better predictions of the objects since it captures the contextual information about the objects instead of just the local features of the object like the color and the shape, which are captured along side the contextual information in our model. Another great advantage of our system is that it is able to generalize well on natural images. We have used COCO data set [5] to train and test our system and we find really good accuracy on our test applications.

Network Design

We take the inspiration for our neural network model from the YOLO architecture [2]. The network is modeled as a convolutional neural network. The features of the image is extracted by the convolutional layer, while the class probabilities are drawn by the fully connected layers. The bounding box co-ordinates are also given by the fully connected layers.

The full network has 24 convolutional layer followed by 2 fully connected layers. We can see the final output of the convoltional layer is a 7X7X30 tensor, which represents the direct bounding boxes in an image. These bounding boxes are then fed into the fully-connected layers to predict the class probabilities corresponding to each of the boxes. The full network architecture is shown in Figure 2.

There is also a tiny-YOLO network which was designed for very fast object detection. The tiny-YOLO framework has 9 convolutional layer instead of 24 and is considerably fast with respect to

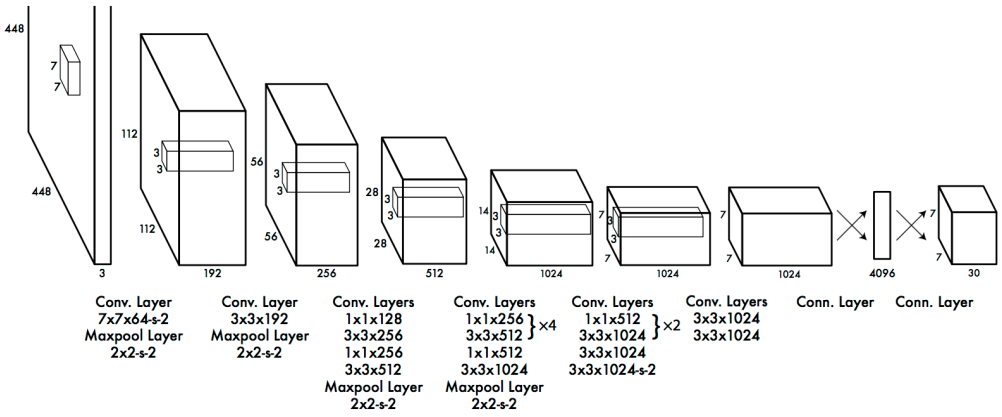

Figure 2: The YOLO Neural Network Architecture

the full network model. There is no change in parameters except the number of layers and a reduced number of filters for each layer in the tiny YOLO with respect to the full YOLO architecture.

Detection as Regression

The neural network system combines different aspect of object detection and localization together in one network. The network extracts features from the entire image to extract bounding boxes for different class labels. Initially it divides the image in SxS grids. If an object lies in the grid, that grid is responsible for detecting that object.

Each grid cells then predict B bounding boxes and the corresponding confidence score for each class. The score is an two-fold indication of the probability that object lies within the box and second, whether the bounding box is correctly predicted. Thus, each bounding box consists of 5 predictions, viz. x, y, w, h and confidence. While x, y is the center of the bounding box, w, h represents the width and height of the bounding box with confidence score.

Each bounding box also predicts the conditional class probabilities for each of the C class. These class probabilities are conditional on presence of object in the grid cell. Although there are B bounding boxes in each grid cell, the conditional class probabilities are calculated for only one set of class probabilities per cell. Thus at test time, the final class specific confidence score is given by Equation 1.

$$Pr(Class_i|Object) * Pr(Object) * IOU^{truth}_{prediction} = Pr(Class_i) * IOU^{truth}_{prediction} \quad (1)$$

Figure 3 shows the detection problem as a regression. using subsequent frame information of a video, we localize the objects in the scene by utilizing the x, y, w and h information of the predicted bounding box. The model we built was evaluated for $S = 7$ with $B = 2$ and $C = 20$ class labels obtaining the network output as $SXSX(B * 5 + C)$ which comes out as $7X7X30$ for our model.

2.2 Perception Module

The camera node of our module captures frames from real-time video feed of camera. The camera node then publishes the image to an image channel which is simultaneously read by a Processing node. The processing node is the central processing unit of our architecture. This is where all the deep neural nets reside. The processing node is subscribed to the image channel and then feed it to the neural nets to detect objects. At a time, only one frame is fed into the network for detection. Depending on the processing speed of the hardware, once the evaluation of an image is complete, the processing node invokes a service to the camera node. The service asks the camera node to capture a frame and publish the image to the image channel.

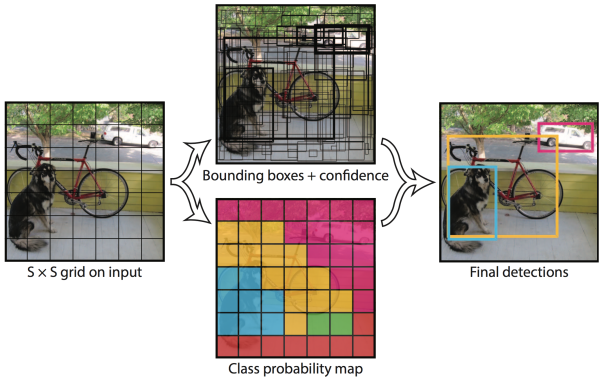

Figure 3: Detection as a regression problem

The processing node outputs the bounding boxes and classification probabilities of the objects in the frame and pass this information to the Movement node. Movement node is an abstract node and can be implemented on a bot, or use it to drive a car - as we did in our project. The movement node makes the decision that whether the bot should move, if yes then what's the direction. Figure 4 describes the architecture of the system.

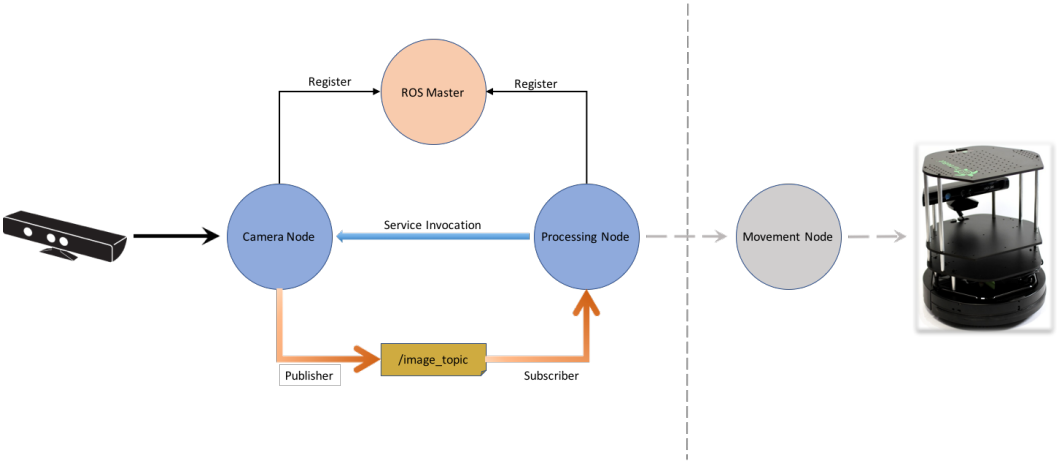

Figure 4: The ROS Perception Module

3 Experiments

3.1 Data Collection

For our initial task of object localization, detection and classification of objects, our model was capturing the images real time from a camera feed. We recorded the output of the video feed generated by our model. We tested our system in various scenarios such as street biking, night drive, and indoors. The output from our model is the video captured from the camera, along with the bounding boxes and their labels.

3.2 Experimental Results

Figure 5 shows the output of video from bike ride during day time. We can see for various scenarios our model constructs the bounding boxes with decent accuracy for both localization and classification.

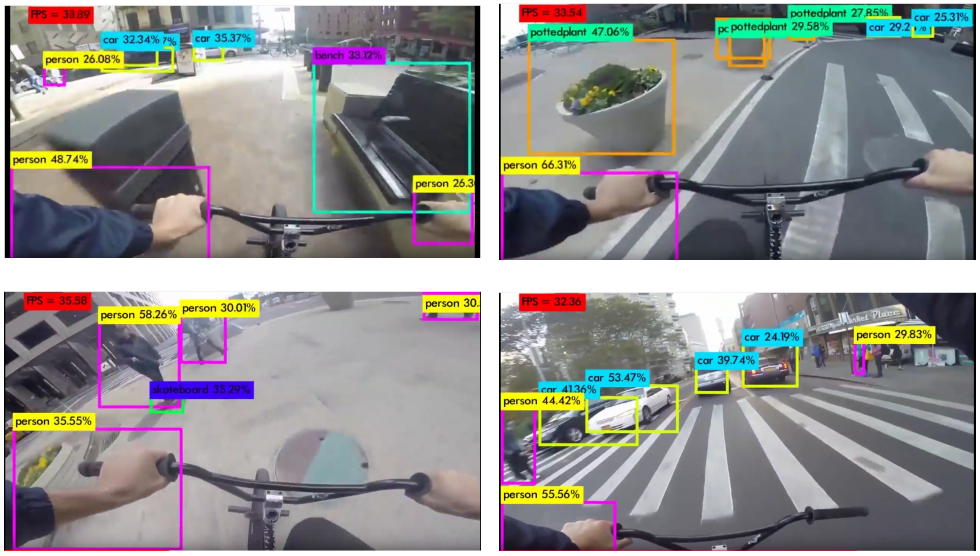

Figure 5: Day time bike ride

Figure 6 shows the results of a similar environment - bike ride, but during night time. We want to show that our model is able to localize, detect and classify objects in real time even when the light is not sufficient. We can see that it recognizes traffic lights, bicycle, cars and various objects.

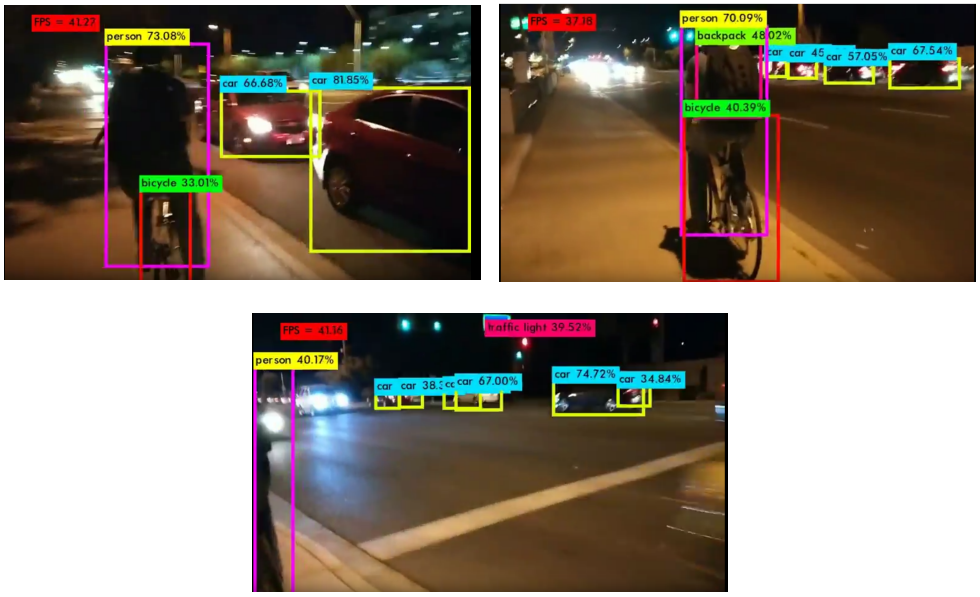

Figure 6: Night time bike ride

The following Figure 7 displays the result of object detection in a simulated environment.

Next we trained our module to drive a car in a game. In Figure 8, we can see that when there's an empty stretch of road the car accelerates, but when there's a truck present ahead of our car, the algorithm predicts to apply breaks and the car stops.

Figure 7: Object detection in a game

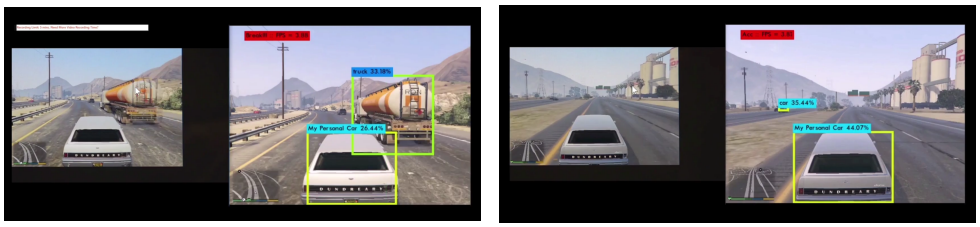

Figure 8: Automated driving in a game

4 Discussion and Conclusion

We developed a system that can localize, detect and classify objects in real time which was the main goal of our system. We also trained our application to drive cars in a simulated environment. We can accelerate and break the car looking at the road - if there's an obstacle present on the road in front.

Table 1 shows the statistics of our model on various hardwares. The best performance was obviously achieved when using GTX 1060, we got frame-per-second rate of around 40. Thinking of a modular bot, we can use NVIDIA TX2 on the bot and run our application there - we can get fps rate of around 10. 10 fps is decent if we want the bot to move around indoors to search for objects and guide the bot to the object location.

Table 1: Statistics

	TK1	GTX 1060	TX2
#Cores	192 Cores (Kepler)	1280 Cores (Pascal)	256 Cores (Pascal)
Memory	2 GB 64 bit	6 GB 192 bit 192 GB/s	8 GB 128 bit 58.3 GB/s
FPS 448X448	4 - 5	35 - 40	8 - 10* (Prediction)

We were not able to deploy our model on TX2 due to insufficient library support. In future, one of our main work will be to port the system on actual embedded system and mobile robot carts, which will give best performance utilizing a powerful GPU like the TX2.

Video links
Day time bike ride: `https://youtu.be/feTTaMCy__0`
Night time bike ride: `https://youtu.be/VLJTF_DuZSg`
GTA V driving: `https://youtu.be/5feduACpWeo`

References

[1] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

[2] 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).

[3] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks, CoRR abs/1312.6229. URL http://arxiv. org/abs/1312.6229.

[4] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[5] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." European Conference on Computer Vision. Springer International Publishing, 2014.