Contents

0.1 Introduction

Frozen products are available and consumed in almost every corner of the world. There are several advantages of frozen products like increased shelf life, cheaper than the conventional food etc. All these products are stored in inventory operating at temperatures beyond -18C and sorting this inventory might be a tough task when humans are involved. Conventional freezers can't be kept open for long time as it runs on CO2 and counting the boxes for daily inventorying can be considered inhumane. So, we have to find ways to automate the process of counting the boxes where human reach is usually not considerable.

0.2 Problem Statement

In this project, a smart warehouse counting system has been implemented using Artificial intelligence, IoT and cloud where the counting of boxes can be done remotely and slightest human intervention.

0.3 Object Detection

Object detection is a subbranch of Computer vision applications in Machine learning. In object detection, similar or semantic objects are detected in a digital image or video. Along with the bounding boxes, class labels and the likelihood of that class is also sometimes mentioned in object detection. In today's world, real-time object detection has various applications such as driver-less cars etc.

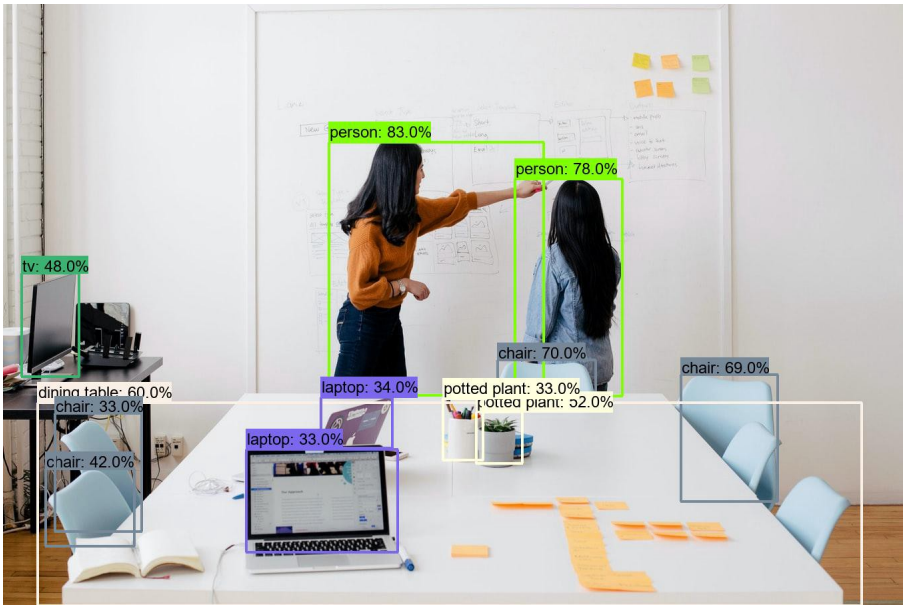


Figure 1: Object detection example.

0.3.1 Intersection over Union

The metric used for object detection is called Intersection over Union or IoU. IoU evaluates the degree of overlap of the bouding box region between the ground truth and prediction. In addition to IoU, we will analyse the average mAP (mean Average Precision) metric. Average mAP at a threshold like 0.50 or 0.70 determines the confidence level threshold we wish to work with. a good mAP indicates that the model is relatively stable and consistent.

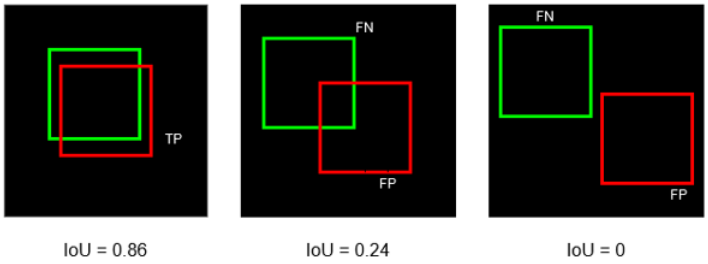


Figure 2: Different types of IoU

0.4 Dataset

0.4.1 Collection :

We have gathered our raw images from a small freezer stock of famous superstore from Calgary. The images were collected using mobile phone under various lighting conditions to create a robust dataset. Moreover, different angles were used while capturing the images. Images contains single and multiple objects.

0.4.2 Preparation :

Labelling of boxes in the image was done using LabelImg software. The .xml file has been exported for each image in the PASCAL VOC format for the bounding boxes to be determined by the model.

0.4.3 Augmentation :

In order to build robust dataset, augmentation was done using ready tools like Roboflow.com. Using Roboflow was very easy. Augmentation included Horizontal and vertical flips, +10 and -10 rotation and some greyscale images.

0.5 Machine Learning Model : YOLO

YOLO stands for You Look Only Once. It's a state-of-the-art neural network. In this architecture, CNN (Convolution Neural Networks) are used in a single forward propagation to simultaneously detect the objects or class probabilities and bounding boxes as well.

Its specially made for object detection. In this algorithm, first images is divided into various grids and each grid has a specific dimension. Each grid, which is of dimension SxS will detect the object appearing in them. So, the YOLO uses the bounding box regression to predict the dimensions of the object in an image and the probability of object appearing in it.

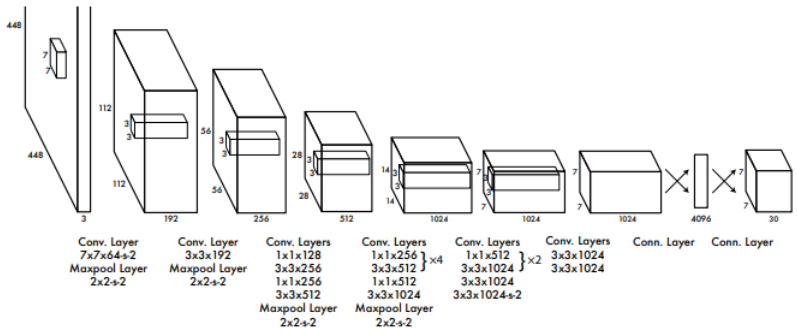


Figure 3: YOLO architecture

0.5.1 Impletmentation of the model

Implementation of YOLOv4 is done on the Darknet framework. It's a custom framework written by Joseph Redmon. We cloned a repository from AlexyAB of Darknet and made changes to it. As our goal was to remotely monitor the number of boxes in the images, we must choose a model which can be implemented on Edge device like Raspberry Pi. Thats the reason, YOLOv4 tiny has been selected with fewer parameters.

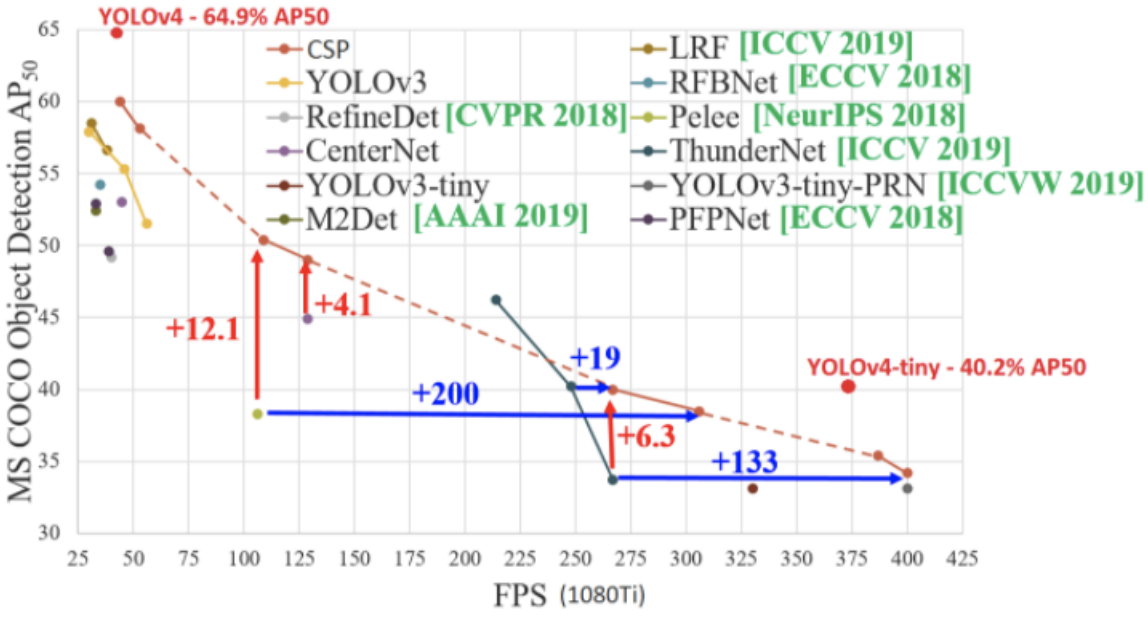


Figure 4: Comparison of different Architectures

0.6 Model Training

- As the model was trained with Darknet architecture, training was relatively fast. We have trained the model for 4000 iterations with a batch size of 2 as smaller batch sizes would lower the generalization error and it offers a regularization effect. Moreover, it helps with memory optimization.
- The training parameters is printed after every iteration and mean Average Precision (mAP @0.50) is printed after every 100 iteration The best weights are kept up to date.
- 3) Once the training is completed, the configuration data is saved and the weights at different iterations as well as the best weights are saved.

0.6.1 Results

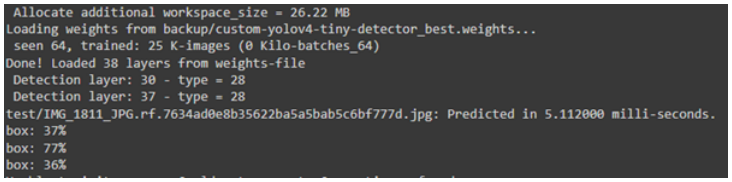


Figure 5: Inference data in Colab

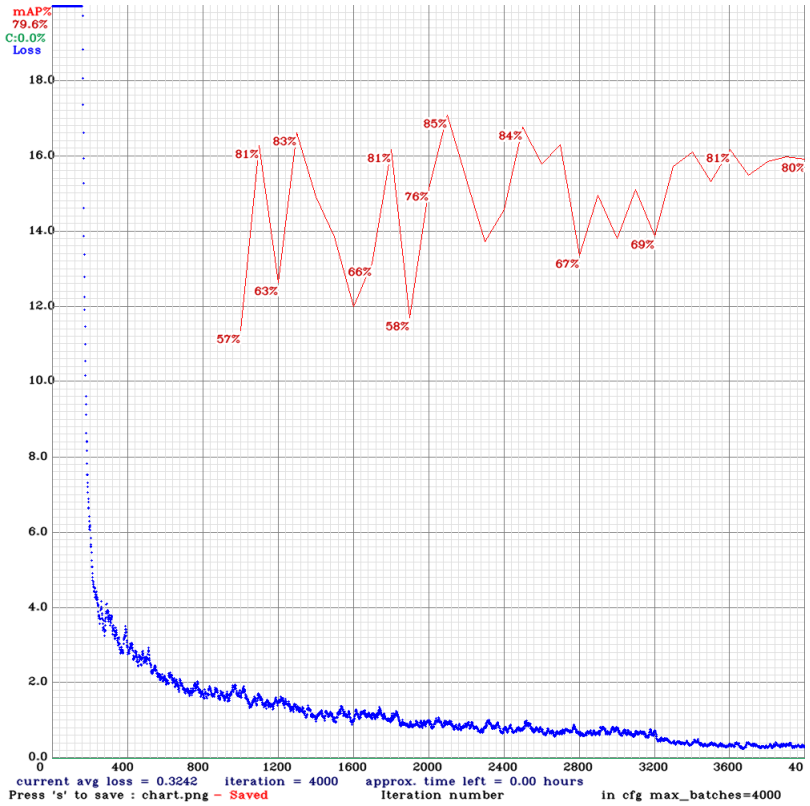


Figure 6: Result analysis

- In the Above figure, we can see the performance of our YOLOv4 tiny model as the number of iterations passes.
- From the graph, we can conclude that in between the 3700 to 4000 iterations, loss is almost the same and increasing the number of iterations may cause the model to overfit and this may decrease the generalisation factor of our model.
- From the mAP (mean Precision Average) point of view, we can see that as the iterations increases, the stability of the mAP also takes place, with the maximum being around 85% after 2200 iterations. Once the peak has passed, there is some consistency in the mAP.

0.7 Integration with Raspberry PI

- Once our model was trained, the configured data as well as best weights have to be transferred to the edge device. We have used Raspberry PI 4 Model B as our edge device.
- Due to the limited scope of this project, instead of taking photos from the attached camera module of the Raspberry PI, we have beforehand uploaded the test images in our Edge devices.
- Now, to automate the workflow of detecting the number of boxes and sending it to the the cloud (azure IoT Central), we have created a shell script and we can change the test image in the shell script itself.

```
Attempting to Send Messages to Azure IoT
Sending message: {'box': 8}
Message Sent
```

Figure 7: Parsing data.

0.7.1 Edge Device Results :

(a) Raw Image - 1 (b) Prediction

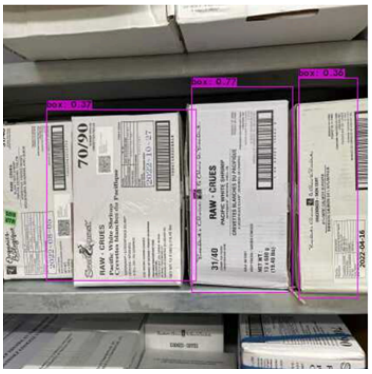
Figure 8: Image type -1(boxes at different angle)

(a) Raw Image -2

(b) Prediction

Figure 9: Image type -2(boxes with shadow in place)

(a) Raw Image -3



(b) Prediction

Figure 10: Image type -3(Different coloured box)

0.8 Remote access

As one of the goal of this project was to access our result from the cloud itself, I have used Microsoft Azure IoT central for it. Azure IoT central is an IoT application platform that is used to create the IoT solutions with an ready to use User interface and API which can be used to connect and manage the edge devices. For this project, a free trial was used. In the figure given below, we can see the screen shot of azure Iot central showing the number of boxes.

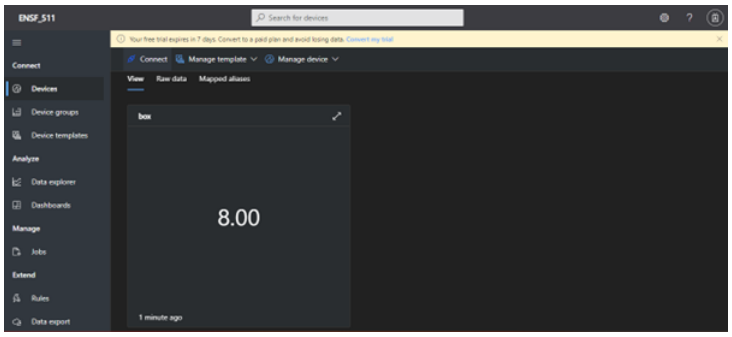


Figure 11: Azure IoT central result

0.9 Challenges

- The initial challenge was the collection of a proper dataset. Since, the freezer was relatively small, lack of organization can be seen. We did organize the boxes in a way that it can depict a small warehouse freezer. During preparation of dataset, it was made sure that theimages are collected with various lighting conditions and angles. Another challenge was lebelling the image. As some images were from different angles, adequate omission and bounding boxes has to be done.

- One of the major challenges was to increase the IoU score. At first, due to low IoU score, boxes were not able to detect with great accuracy and due to which model was not performing well. The tweak here was to increasing the iterations adequately so as not to overfit while decreasing the batch size. Since training was performed on Google Colab, the allocation of resources were limited and due to which it took more time to experiment.

- Another challenge came up when trying to set to set up the Raspberry Pi and Azure IoT central. As new Raspbian OS has been launched, it took some time to get used to it and make it work for the project.

0.10 Future Improvements

- Detection on Video : One of the future tasks involves the Machine Learning model implementation on a video gathered using Drones or camera modules attached to a edge device. The video will be parsed to the edge device when the inference will take place and the data will be further parsed to the cloud to remotely count the number of boxes.

- Training on Different Architectures : As the goal of this project was to deploy the model on a edge device, we have presumed the use of tiny version of YOLOv4 architecture. As a future task, I will try out different lite architectures and compare their IoU score in order to select the best model for this task.

0.11 References

- Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).
- Voutos, Yorghos Drakopoulos, Georgios & Mylonas, Phivos. (2019). Smart Agriculture: An Open Field For Smart Contracts.
- Train YOLOv4-tiny on Custom Data - Lightning Fast Object Detection. Roboflow. *https://blog.roboflow.com/train-yolov4-tiny-on-custom-data-lighting-fast-detection/*
- YOLOv4 vs YOLOv4-tiny, Medium.com. *https://medium.com/analytics-vidhya/yolov4-vs-yolov4-tiny-97932b6ec8ec*
- Google Colaboratory *https://colab.research.google.com*
- Machine Learning Tracking using Raspberry Pi. *https://www.hackster.io/*
- Microsoft Azure IoT central *https://azure.microsoft.com/en-us/services/iot-central/*
- Yolo v4, v3 and v2 for Windows and Linux, by AlexyAB. *https://github.com/AlexeyAB/darknet*
- Roboflow.com for splitting,preprocessing and augmentation of dataset. *https://app.roboflow.com*
- Raspbian Pi for home. *https://www.raspberrypi.com/for-home/*