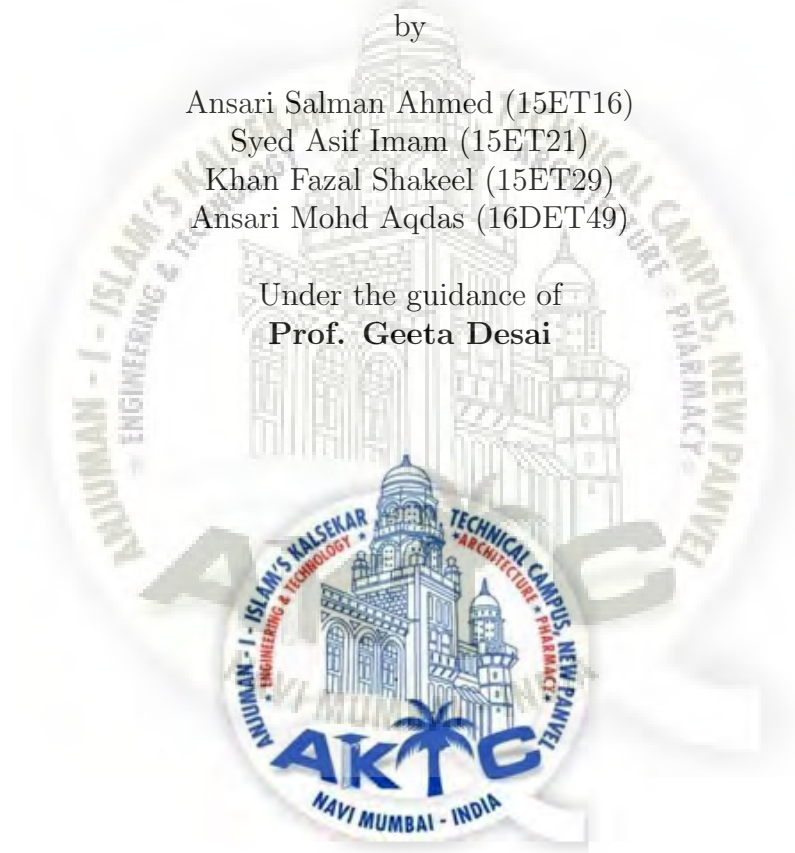


OBJECT AND DEFECT DETECTION USING IMAGE PROCESSING

Submitted in partial fulfillment of the requirements
for the degree of
Bachelor of Engineering
in
Electronics and Telecommunications
by

Ansari Salman Ahmed (15ET16)
Syed Asif Imam (15ET21)
Khan Fazal Shakeel (15ET29)
Ansari Mohd Aqdas (16DET49)

Under the guidance of
Prof. Geeta Desai



Department of Electronics and Telecommunications

Anjuman-I-Islam's Kalsekar Technical Campus
Sector -16, New Panvel, Navi Mumbai
University of Mumbai

Academic Year : 2018-2019

CERTIFICATE



Department of Electronics and Telecommunication Engineering
Anjuman-I-Islam's Kalsekar Technical Campus
Sector 16, New Panvel , Navi Mumbai
University of Mumbai

This is to certify that the project entitled “**Object And Defect Detection Using Image Processing**” is a bonafide work of **Ansari Salman Ahmed (15ET16)**, **Syed Asif Imam (15ET21)**, **Khan Fazal Shakeel (15ET29)**, **Ansari Mohd Aqdas (16DET49)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of Bachelor of Engineering in Department of Electronics and Telecommunication Engineering.

Supervisor

Examiner

Head of Department

Director

Project Report Approval for Bachelor of Engineering

This project entitled **Object And Defect Detection Using Image Processing** by **Ansari Salman Ahmed (Roll No:15ET16)**, **Syed Asif Imam (Roll No:15ET21)**, **Khan Fazal Shakeel (Roll No:15ET29)** and **Ansari Mohd Aqdas (Roll No:16DET49)** is approved for the degree of **Bachelor of Engineering in Department of Electronics and Telecommunications**.



Date:

Place:

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

.....
Ansari Salman Ahmed
Roll No:15ET16

.....
Syed Asif Imam
Roll No:15ET21

.....
Khan Fazal Shakeel
Roll No:15ET29

.....
Ansari Mohd Aqdas
Roll No:16DET49

Date:

Acknowledgements

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals . I would like to extend my sincere thanks to all of them.

We are highly indebted to Prof.Geeta Desai for her guidance and constant supervision as well as for providing necessary information regarding the project also for her support in completing the project.

We would like to express my gratitude towards my parents Staff of AnjumanI-Islam's Kalsekar Technical Campus for their kind co-operation and encouragement which help me in completion of this project.

My thanks and appreciations also go to my colleague in developing the project and people who have willingly helped me out with their abilities.

Ansari Salman Ahmed (15ET16)

Syed Asif Imam (15ET21)

Khan Fazal Shakeel (15ET29)

Ansari Mohd Aqdas (16DET49)

Abstract

Object And Defect Detection Using Image Processing

In Industrial development and production, quality imposition and maintenance are growing rapidly for the production of high quality final product and accurate specifications. Testing team in the industry strive to catch faults before the product is released but they always and they often reappear, even with the best manual testing process. Automated testing method is the best way to increase efficiency and analysis of our product testing. Defect in object can be found with Quality Control of object using Image Processing. It also shows the divergence for a fast evaluation of fault detection. This means early detection of possible problems so that process can be corrected in time, resulting in efficient quality control. Industries that implement these automated testing techniques benefit for lower testing time for product inspection. Sometimes, the defects in the components are found after the delivery of the product to the respective customers, even after effective manual testing. This leads to wastage of the product and manufacturing cost or requires rechecking. This project will extract the defective object or different types of object using tensorflow,open-cv on raspberry pi 3. It will help in industries to be free from human error and thus provide fault free product.Our object detection system, called Single Shot MultiBox Detector. 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.

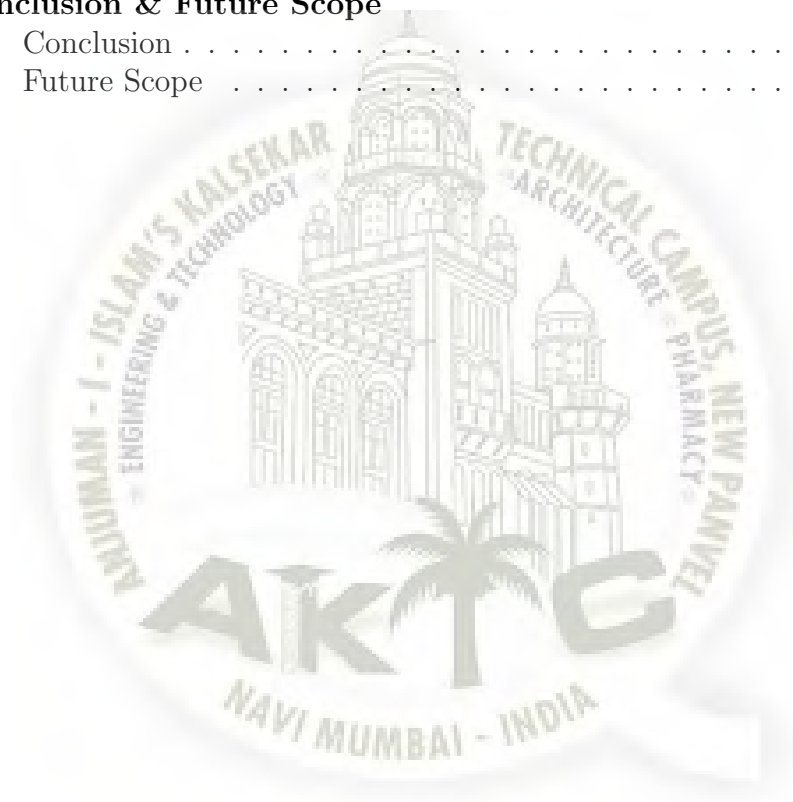
Keyword:

Computer vision, machine vision, single shot multibox detector, feed forward convolution network, convolutional neural networks, tensorflow, feature map, convolutional layers, bounding box, raspberry pi.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 1 |
| 1.1 | Objective | 3 |
| 1.2 | Problem Statement | 3 |
| 1.3 | Motivation | 3 |
| 1.4 | Thesis Organization | 4 |
| 2 | Literature Review | 5 |
| 2.1 | Research on Daily Objects Detection Based on Deep Neural Network | 5 |
| 2.2 | Deep Learning for Computer Vision: A Brief Review | 5 |
| 2.3 | Review on Moving Object Detection in Video | 6 |
| 2.4 | Machine parts recognition and defect detection in automated assembly systems using computer vision techniques | 6 |
| 2.5 | Quality checking and inspection based on machine vision technique to determine tolerancevalue using single ceramic cup | 7 |
| 2.6 | Quality Control of PCB using Image Processing | 8 |
| 2.7 | SSD:Single Shot MultiBox Detector | 8 |
| 2.8 | Single Shot Multi-Box Detector with Multi Task Convolutional Network for Carabao Mango Detection and Classification using Tensorflow | 9 |
| 3 | Technical Details | 10 |
| 3.1 | Object Detection | 10 |
| 3.1.1 | R-CNN | 11 |
| 3.1.2 | Fast R-CNN | 11 |
| 3.1.3 | Faster R-CNN | 12 |
| 3.1.4 | SSD and R-FCN | 13 |
| 3.2 | Single Shot Detector | 14 |
| 3.2.1 | VGG16 | 18 |
| 3.2.2 | MultiBox | 19 |

| | | |
|----------|--------------------------------------|-----------|
| 4 | Proposed Methodology | 24 |
| 4.1 | Block Diagram | 24 |
| 4.2 | System Flow Diagram | 25 |
| 4.3 | Project Requirements | 26 |
| 4.3.1 | Software Requirements | 26 |
| 4.3.2 | Hardware Requirements | 29 |
| 5 | Results & Discussion | 33 |
| 5.1 | Implementation | 33 |
| 5.2 | Result | 42 |
| 6 | Conclusion & Future Scope | 46 |
| 6.1 | Conclusion | 46 |
| 6.2 | Future Scope | 46 |



Glossary

RCNN- Regional Convolutional Neural Network
RST- Rotation Scaling Translation
SSD-Single Shot MultiBox Detection
CMOS- complementary metal-oxide-semiconductor
PCB- Printed Circuit Board
RPN- Region Proposal Network
COCO- Common Objects in Context
ILSVRC- ImageNet Large Scale Visual Recognition Challenge
CNN- Convolutional neural network
ROI- Region of interest
YOLO- You only look once
SVMs- support vector machines
R-FCN- Fully Convolutional Network
VGG- Visual Geometry Group
IOU- Intersection Over Union
NMS- Non-Maximum Suppression
CPU- central processing unit
GPU- Graphics Processing Unit
TPU- Tensor Processing Unit
AI- Artificial Intelligence
XML- eXtensible Markup Language
DSP- Digital signal processor
FPS- Frame per second
OpenCV- Open Source Computer Vision

Chapter 1

Introduction

There are numbers of method used in the modern industry depends on the requirement of inspection. Different manufacturing process typically needs a specially designed machine vision for high performance in the inspection process with least expenses on their inspection system. Internal checking usually involves in food industry where it is required to see if the food is packed correctly in place or position, the amount of food or contains to fulfil the production requirement. Whereas, external inspection typically involves the packaging and printing on a product if is there any damages or printing error occur. Thus, many different criteria of an inspection process have been discussed as in the follows.

High technology inspection systems are implemented in modern industry and manufacturing process to replace with labour inspection that may cause technical issues or error due to mankind physical constraints. Some inspection typically need visual system which is similar to how human's eye observe. Computer vision refers in broad term to the capture and automation of image analysis, whereas machine vision refers to the use of computer vision to factory automation. In the early century, manufacturing process and plantations often use human eye or labours for inspection of every product until lately both the computer and light sensory device has been introduced to the public.

People then started to use computer and sensory device for image analysis and processing to try to imitate human inspection and replace labour with computer vision system. This can avoid human errors and improve the productivity of certain product, as it can be operate in 24 hours per day with the source of electricity.

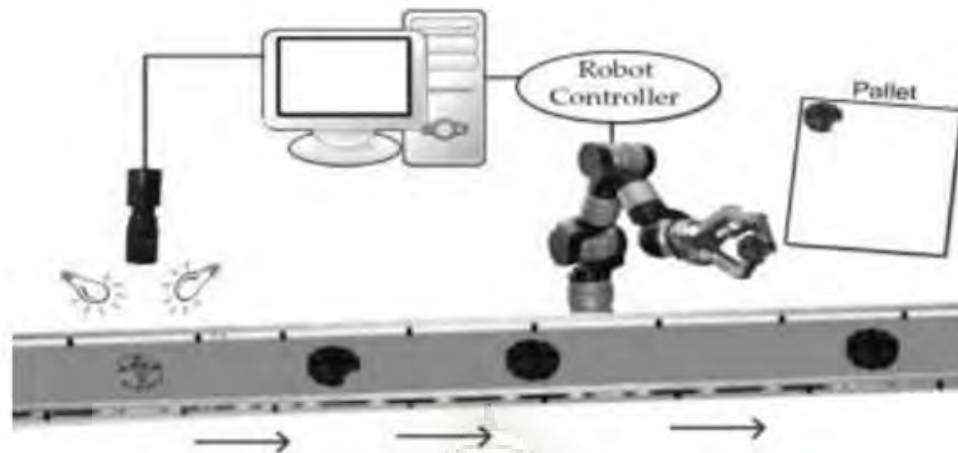


Fig 1.1 A schematic diagram for machine parts defect detection

The object recognition in automated assembly industries is entirely different from general object recognition; moreover the ability of human to distinguish between healthy and unhealthy object are good but it is a complicated task for a machine. In general manual defect detection by human inspectors are impractical with fast moving machine parts on conveyor in addition it is expensive, subjective, inaccurate, eye straining and other health issues to quality control inspectors. By considering these issues, a computer vision based non-contact inspection technique is developed for defect detection in industrial machine parts by image processing techniques. The present study will help the industrial robot used in assembly process and industrial inspection systems. The defect detection is performed on machine parts at early stages of the assembly line to ensure product quality. The schematic diagram for defect detection from machine parts is shown in Figure 1.1. The 2D vision of machine parts which are moved along the conveyor are captured and its boundary characteristics are analyzed to match with its model shape. If there is any deviation in matching result leads to notification of defect which in turn instructs the robot controller to pick up the defective piece from the conveyor.

To inspect a certain object, a suitable algorithm is required to determine whether the sample inspected is good unit or defective unit. Once the digital image is captured by the camera, the image will then undergo image processing which includes object recognition and detection via Python programming language. Besides, a computer vision library for image analysis and processing of an open-source computer vision library (OpenCV) is needed to develop the algorithm for inspection process.

Object detection has always been the focus and challenge in the field of computer vision, including the detection for small objects. To recognize objects at various scales, the majority of previous detectors are based on hand-crafted features. Those works are computationally expensive considering memory and inference time. With the arrival of deep convolutional networks, the performance of object detection has been improved significantly. However, small object detection is still a challenging issue due to relatively small area with less information in images. Images of objects from a particular class are highly variable. One source of variation is the actual imaging process. Changes in illumination, changes in camera position, all produce significant variations in image appearance, even in a static scene. The second source of variation is due to the intrinsic appearance variability of objects within a class, even assuming no variation in the imaging process. For example, people have different shapes and wear a variety of clothes, while the handwritten digit 7 can be written with or without a line through the middle, with different slants, stroke widths, etc.

1.1 Objective

In this project, we are working on the industrial inspection of the product. The objective of this project is to scan frames and extract the features of the object and score them. The predefined object is selected to remove from conveyor belt. The project can also score the object and if the score is below a set value than the object is removed. This will be implemented by using Single Shot MultiBox Detector in Python using Tensorflow and Open-CV.

1.2 Problem Statement

The core problem faced by machine recognizing is the frame per second and precise detection of object. To achieve high frame per second and precise value of object, system should have high computational system. Thus, the cost is increase for the system. This limitation provided us an impetus to build a cost effective, efficient and moderate speed processing to scan an image/video without constant inspection.

1.3 Motivation

The basic motivation behind this project is that machine will automatically operate the quality of the manufacturing product and eliminate the human error. Object and defect detection can reduce human effort and

provides efficiency. It is of interest as it may help humans to be aware of product information and its quality. Object recognition and extraction in the smart system can be used in industries.

1.4 Thesis Organization

The organization of this thesis is as follows:

Chapter 1: Introduction. This chapter describes the general introduction for object detection use in industrial inspection along with thesis objective, problem statement and motivation.

Chapter 2: Literature Review. This chapter gives a review of the different papers for object detection.

Chapter 3: Technical Details. This chapter gives the information on object detection and its types. This chapter also describes the single shot multibox detector explanation, architecture and its base model VGG16 in detail.

Chapter 4: Proposed Methodology. This chapter covers the block diagram, system flow diagram and requirements of the project which explain the methodology algorithm and describe steps that are followed in the implementation procedure.

Chapter 5: Results Discussion. This chapter discusses and explains the obtained experimental results for object detection.

Chapter 6: Conclusion Future Scope. The recommendation and need of future work for the improvement of efficiency and reliability of the developed method is suggested.

Chapter 2

Literature Review

2.1 Research on Daily Objects Detection Based on Deep Neural Network

Author: Sheng Ding, Kun Zhao

This article mainly focus on a small daily items detection data set. The data set is then trained on different object detection models and has achieved good results in daily object detection. These well trained models can be used in the mobile platform, Nao robot platform or other intelligent devices, the daily items to achieve accurate detection. In the future, we can get a better model by increasing the capacity of the data set, the optimization of the model structure and the fine tuning of the parameters.

2.2 Deep Learning for Computer Vision: A Brief Review

Author: Athanasios Voulodimos , Nikolaos Doulamis, Anastasios Doulamis and Eftychios Protopapadakis

This review paper provides a brief overview of some of the most significant deep learning schemes used in computer vision problems, that is, Convolutional Neural Networks, Deep Boltzmann Machines and Deep Belief Networks, and Stacked Denoising Autoencoders. A brief account of their history, structure, advantages, and limitations is given, followed by a description of their applications in various computer vision tasks, such as object detection, face recognition, action and activity recognition, and human pose estimation. Finally, a brief overview is given of future directions

in designing deep learning schemes for computer vision problems and the challenges involved therein.

2.3 Review on Moving Object Detection in Video

Author: Aqsa Khan , Mr. N. J. Janwe

This paper presents a review and systematic study on the moving object detection and surveillancing of the video as they are important and challenging task in many computer vision applications. Such as human detection algorithm, vehicles detection, threat, security. Video surveillancing in a dynamic environment ,especially for human and vehicles and for specific object in case of security is one of the current challenging research topic in computer vision it is a key technology to fight against terrorism, crime, public safety and for efficient management of accidents and crime seen going on now a days. The paper also presents the concept of real time implementation computing task in video surveillances system. In this review paper various methods are discussed were evaluation of order to access how well they can detect moving object in an outdoor/indoor section in real time situation.

2.4 Machine parts recognition and defect detection in automated assembly systems using computer vision techniques

Author: P.Arjun , T.T.Mirnalinee

In this paper, a computationally efficient 2D computer vision based approach to recognize the machine parts and detect damaged parts in automated assembly systems has been presented. The machine part defects in the form of damage, cracks are identified by scanning the shape of the object and a feature vector is generated from the shape. The shape descriptor discussed is simple, compact, and fast one dimensional feature vector which preserves the shape information using contour pixel coordinates of the shape. The shape descriptor was generated by partitioning the shape object into fixed number of equal part area segments with respect to centroid using sector area approach. The feature vector is a 1D array consists of distance between centroid to subsequent normalized contour points.

Two kinds of tests were performed on the machine part images to confirm the validity of the descriptor; namely RST invariance test and defect detection test. The correlation coefficient metric was used for comparing the images for recognition and detection of defects in machine part. From the experiments, the centroid to normalized contour points distance shape descriptor performs good for machine parts recognition and defect detection. Future work will be focused on development and integration of multiple features from equal area normalization descriptor to further strengthen it in terms of accuracy and efficiency.

2.5 Quality checking and inspection based on machine vision technique to determine tolerance value using single ceramic cup

Author: Nursabillilah Mohd Alie , Mohd Safirin Karis , Gao-Jie Wong , Mohd Bazli Bahar , Marizan Sulaiman , Masrullizam Mat Ibrahim and Amar Faiz Zainal Abidin

The development of an algorithm for inspection and quality checking using machine vision was discussed in this paper. The design of the algorithm is to detect the sign of defect when a sample of the product is used for inspection purposes. It is also designed to track specific colour of product and conduct the inspection process. Programming language of python and open source computer vision library were used to design the inspection algorithm based on the algorithm required to achieve the inspection task. Illumination and surrounding environment were considered during the design as it may affect the quality of image acquisition by image sensor. Experiment and set-up by using CMOS image sensor were conducted to test the designed algorithm for effectiveness evaluation. The experimental results were obtained and are represented in graphical form for further analysis purposes. Besides, analysis and discussion were made based on the obtained results through the experiments. The designed algorithm is able to perform the inspection by sample object detection and differentiate between good and defect unit.

2.6 Quality Control of PCB using Image Processing

Author: Rasika R. Chavan, Swati A. Chavan, Gautami D. Dokhe, Mayuri B. Wagh, Archana S. Vaidya

In this paper an automated testing system for Printed Circuit Board (PCB) is used to get the technological advances in PCBs design and manufacturing, eliminates particular aspects and then provides fast, quantitative, and dimensional impositions. It reduces the testing time and manufacturing cost as human inspectors decisions are ineffective, slow and costly. Thus in this area, digital image processing can be used mainly for the detection of faulty parts or missing components. This system mainly deals with analysis to detect faulty PCB. Digital camera is used in automated visual inspection system that captures image of each sample PCB product. The captured image is then provided to computer for further processing which includes conversion in various forms such as Gray scale image and binarized image. XOR operation is performed on these converted images to obtain the required results. Contour Analysis is performed on these results for classification. Missing components, polarities, circuit breaks, missing tracks these types of faults are detected and classified accordingly. This concept increases the speed and accuracy, eliminates human errors which are frequent in quality testing and also overcomes the weakness in the existing system. Hence the productivity can be increased by replacing manual testing with the proposed concept.

2.7 SSD: Single Shot MultiBox Detector

Author: Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg

The paper present a method for detecting objects in images using a single deep neural network. The approach of paper, named 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. SSD is simple relative to methods that require object proposals because it completely eliminates proposal generation and subsequent pixel or feature re-sampling stages and encapsulates all computation in a single network.

This makes SSD easy to train and straightforward to integrate into systems that require a detection component. Experimental results on the PASCAL VOC, COCO, and ILSVRC datasets confirm that SSD has competitive accuracy to methods that utilize an additional object proposal step and is much faster, while providing a unified framework for both training and inference. For 300300 input, SSDa achieves 74.3% mAP1 on VOC 2007 test at 59 FPS on a Nvidia Titan X and for 512512 input, SSD achieves 76.9% mAP, outperforming a comparable state of the art Faster R-CNN model. Compared to other single stage methods, SSD has much better accuracy even with a smaller input image size.

2.8 Single Shot Multi-Box Detector with Multi Task Convolutional Network for Carabao Mango Detection and Classification using Tensorflow

Author: Ryan Joshua H. Liwag, Kevin Jeff T. Cepria, Anfernee S. Rapio, Karlos Leo F. Castillo, Melvin K. Cabatuan, Edwin J. Calilung

The project on this paper scan the quality of the mangoes. As the input of the system would be images of mangoes, convolutional neural networks are the most appropriate deep learning system for this application. This study would be using Visual Geometry Group 16 deep convolutional model due to its lower overhead delay compared to other models. The algorithm was then retrained and refined using transfer learning so that it could be used to classify images of mangoes into three ripeness categories, unripe, partially ripe, and ripe, on a Raspberry Pi 3-controlled hardware system for portability and mobility. After obtaining an accuracy that is less than ideal for the industry, the neural network was adjusted by decreasing its learning rate and adding a drop out layer within the network. The new accuracy achieved by the system was at 98.32%. The neural network was also installed on a laptop for increased computing power and to show the versatility of the system with regards to hardware. This research therefore shows the potential of using technology in supporting the advancement of the agriculture industry.

Chapter 3

Technical Details

3.1 Object Detection

Object detection is one of the areas of computer vision that is maturing very rapidly because of deep learning. Every year, new algorithms/models keep on outperforming the previous ones. There is a lot of pre-trained models for object detection (YOLO, RCNN, Fast RCNN, Mask RCNN, Single Shot Multibox Detection etc.). CNNs are the basic building blocks for most of the computer vision tasks in deep learning era. Localization and Object detection are two of the core tasks in Computer Vision, as they are applied in many real-world applications such as Autonomous vehicles and Robotics.

- Classification/Recognition: Given an image with an object, find out what that object is. In other words, classify it in a class from a set of predefined categories.
- Localization: Find where the object is and draw a bounding box around it.
- Object detection: Classify and detect all objects in the image. Assign a class to each object and draw a bounding box around it.
- Semantic Segmentation: Classify every pixel in the image to a class according to its context, so that each pixel is assigned to an object.
- Instance Segmentation: Classify every pixel in the image to a class so that each pixel is assigned to a different instance of an object.

Object detection is the problem of finding and classifying a variable number of objects on an image. The important difference is the “variable” part. In contrast with problems like classification, the output of object detection is variable in length, since the number of objects detected may change from image to image. One of the first advances in using deep

learning for object detection was OverFeat from NYU published in 2013. They proposed a multi-scale sliding window algorithm using Convolutional Neural Networks (CNNs).

3.1.1 R-CNN

Quickly after OverFeat, Regions with CNN features or R-CNN from Ross Girshick, et al. at the UC Berkeley was published which boasted an almost 50% improvement on the object detection challenge. What they proposed was a three stage approach:

- Extract possible objects using a region proposal method (the most popular one being Selective Search).
- Extract features from each region using a CNN.
- Classify each region with SVMs.

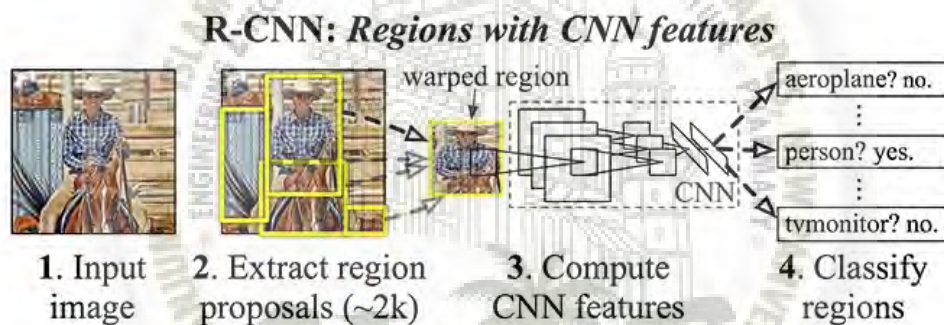


Fig 3.1.1 R-CNN Architecture

While it achieved great results, the training had lots of problems. To train it you first had to generate proposals for the training dataset, apply the CNN feature extraction to every single one (which usually takes over 200GB for the Pascal 2012 train dataset) and then finally train the SVM classifiers.

3.1.2 Fast R-CNN

This approach quickly evolved into a purer deep learning one, when a year later Ross Girshick (now at Microsoft Research) published Fast R-CNN. Similar to R-CNN, it used Selective Search to generate object proposals, but instead of extracting all of them independently and using SVM classifiers, it applied the CNN on the complete image and then used both

Region of Interest (RoI) Pooling on the feature map with a final feed forward network for classification and regression. Not only was this approach faster, but having the RoI Pooling layer and the fully connected layers allowed the model to be end-to-end differentiable and easier to train. The biggest downside was that the model still relied on Selective Search (or any other region proposal algorithm), which became the bottleneck when using it for inference.

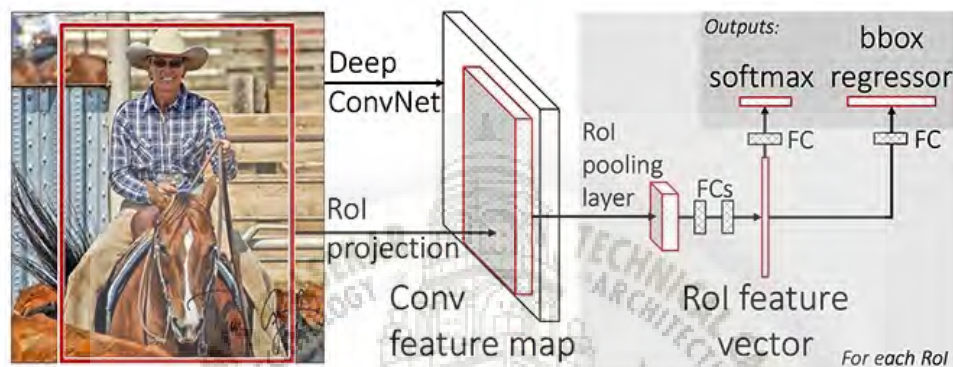


Fig 3.1.2.1 Fast R-CNN architecture

Shortly after that, You Only Look Once: Unified, Real-Time Object Detection (YOLO) paper published by Joseph Redmon (with Girshick appearing as one of the co-authors). YOLO proposed a simple convolutional neural network approach which has both great results and high speed, allowing for the first time real time object detection.

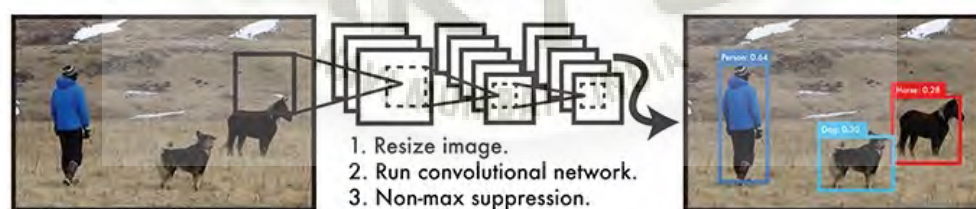


Fig 3.1.2.2 Fast R-CNN Detection Example

3.1.3 Faster R-CNN

Subsequently, Faster R-CNN authored by Shaoqing Ren (also co-authored by Girshick, now at Facebook Research), the third iteration of the R-CNN series. Faster R-CNN added what they called a Region Proposal Network (RPN), in an attempt to get rid of the Selective Search algorithm and

make the model completely trainable end-to-end. We won't go into details on what the RPNs does, but in abstract it has the task to output objects based on an "objectness" score. These objects are used by the RoI Pooling and fully connected layers for classification.

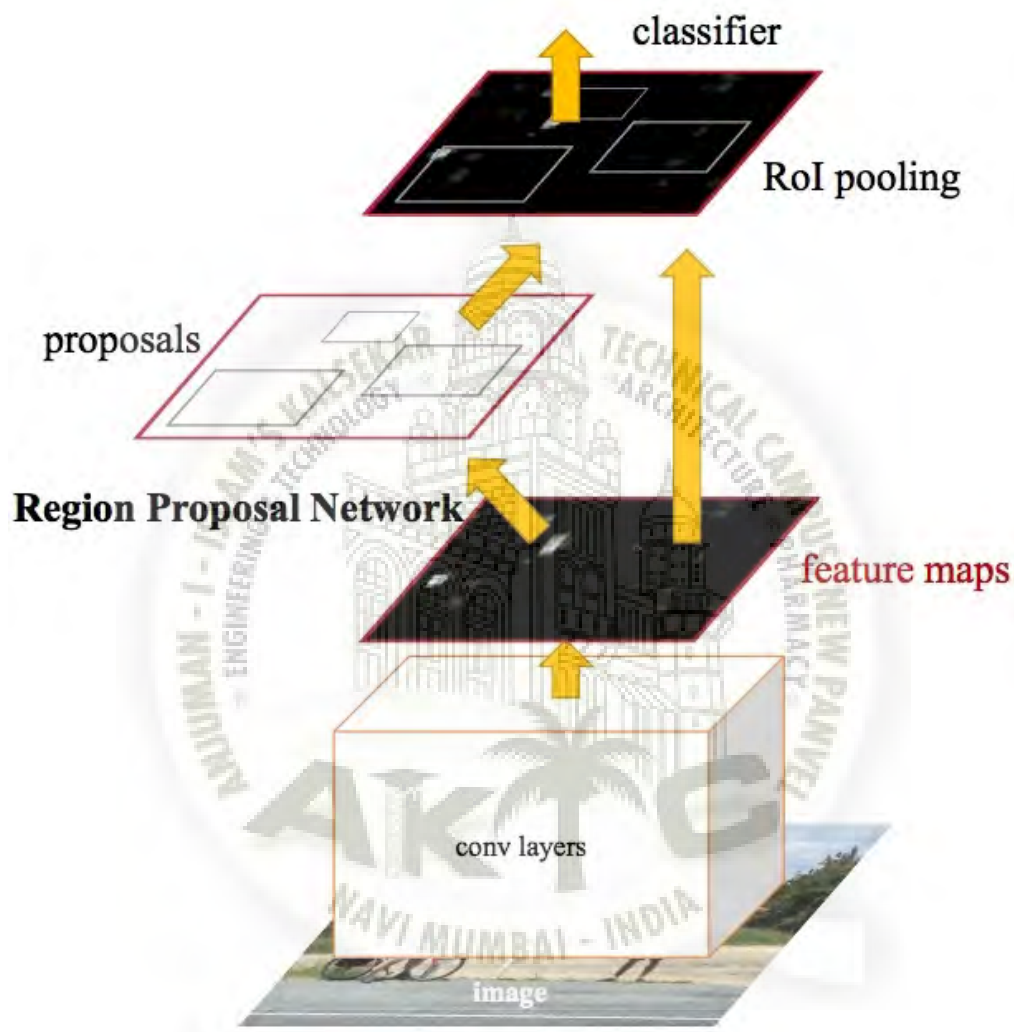


Fig 3.1.3 Faster R-CNN architecture

3.1.4 SSD and R-FCN

Finally, there are two notable papers, Single Shot Detector (SSD) which takes on YOLO by using multiple sized convolutional feature maps achieving better results and speed, and Region-based Fully Convolutional Networks (R-FCN) which takes the architecture of Faster R-CNN but with only convolutional networks. In our project we are using SSD model.

3.2 Single Shot Detector

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 step to produce the final detections. The early network layers are based on a standard architecture used for high quality image classification (truncated before any classification layers), which we will call the base network. We then add auxiliary structure to the network to produce detections with the following key features: Multi-scale feature maps for detection We add convolutional feature layers to the end of the truncated base network. These layers decrease in size progressively and allow predictions of detections at multiple scales. The convolutional model for predicting detections is different for each feature layer (cf Overfeat and YOLO that operate on a single scale feature map). Convolutional predictors for detection Each added feature layer (or optionally an existing feature layer from the base network) can produce a fixed set of detection predictions using a set of convolutional filters. These are indicated on top of the SSD network architecture in Fig. 3.2.1. For a feature layer of size $m \times n$ with p channels, the basic element for predicting parameters of a potential detection is a $3 \times 3 \times p$ small kernel that produces either a score for a category, or a shape offset relative to the default box coordinates. At each of the $m \times n$ locations where the kernel is applied, it produces an output value. The bounding box offset output values are measured relative to a default box position relative to each feature map location (cf the architecture of YOLO that uses an intermediate fully connected layer instead of a convolutional filter for this step). Default boxes and aspect ratios We associate a set of default bounding boxes with each feature map cell, for multiple feature maps at the top of the network. The default boxes tile the feature map in a convolutional manner, so that the position of each box relative to its corresponding cell is fixed. At each feature map cell, we predict the offsets relative to the default box shapes in the cell, as well as the per-class scores that indicate the presence of a class instance in each of those boxes. Specifically, for each box out of k at a given location, we compute c class scores and the 4 offsets relative to the original default box shape. This results in a total of $(c + 4)k$ filters that are applied around each location in the feature map, yielding $(c + 4)k \times m \times n$ outputs for a $m \times n$ feature map. For an illustration of default boxes, please refer to Fig. 1. Our default boxes are similar to the anchor boxes used in Faster R-CNN, however we apply them to several feature maps of different resolutions. Allowing different default box shapes in several feature maps let us efficiently discretize the space of possible output box shapes.

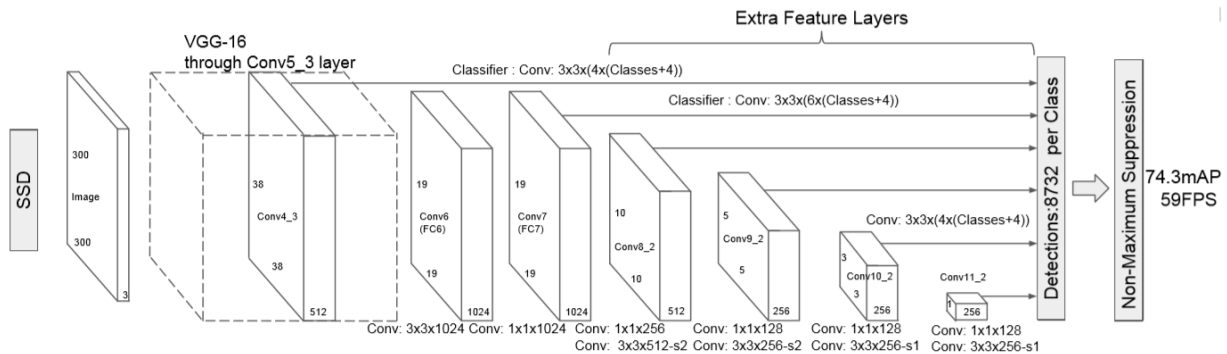


Fig 3.2.1 Single Shot MultiBox Detector architecture

A comparison between two single shot detection models: SSD and YOLO . Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300300 input size significantly outperforms its 448448 YOLO counterpart in accuracy on VOC2007 test while also improving the speed. By using SSD, we only need to take one single shot to detect multiple objects within the image, while regional proposal network (RPN) based approaches such as R-CNN series that need two shots, one for generating region proposals, one for detecting the object of each proposal. Thus, SSD is much faster compared with two-shot RPN-based approaches.

The key difference between training 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. Some version of this is also required for training in YOLO and for the region proposal stage of Faster R-CNN and MultiBox. Once this assignment is determined, the loss function and back propagation are applied end to-end. Training also involves choosing the set of default boxes and scales for detection as well as the hard negative mining and data augmentation strategies.

Matching strategy During training we need to determine which default boxes correspond to a ground truth detection and train the network accordingly. For each ground truth box we are selecting from default boxes that vary over location, aspect ratio, and scale. We begin by matching each ground truth box to the default box with the best jaccard overlap (as in MultiBox). Unlike MultiBox, we then match default boxes to any ground truth with jaccard overlap higher than a threshold (0.5). This simplifies the

learning problem, allowing the network to predict high scores for multiple overlapping default boxes rather than requiring it to pick only the one with maximum overlap.

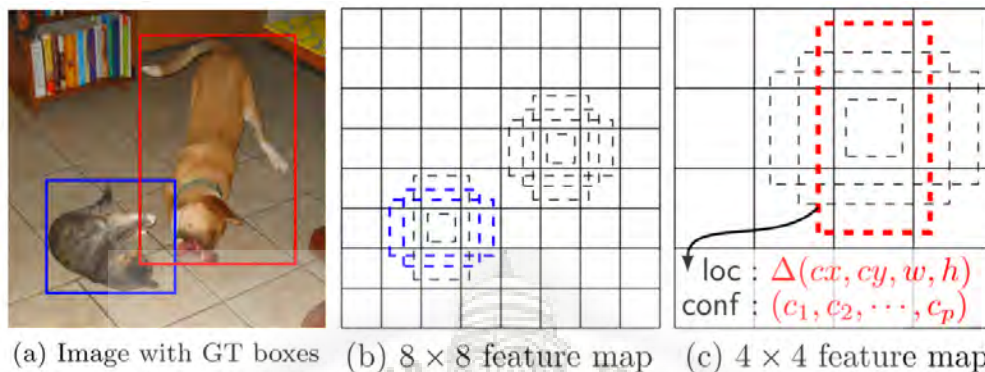


Fig 3.2.2 Generating of Feature map

SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 88 and 44 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories $((c_1, c_2, \dots, c_p))$. At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss

Choosing scales and aspect ratios for default boxes to handle different object scales, some methods suggest processing the image at different sizes and combining the results afterwards. However, by utilizing feature maps from several different layers in a single network for prediction we can mimic the same effect, while also sharing parameters across all object scales. Previous works have shown that using feature maps from the lower layers can improve semantic segmentation quality because the lower layers capture more fine details of the input objects. Similarly, showed that adding global context pooled from a feature map can help smooth the segmentation results.

Motivated by these methods, we use both the lower and upper feature maps for detection. Figure shows two exemplar feature maps (88 and 44) which are used in the framework. In practice, we can use many more

with small computational overhead. Feature maps from different levels within a network are known to have different (empirical) receptive field sizes. Fortunately, within the SSD framework, the default boxes do not necessary need to correspond to the actual receptive fields of each layer. We design the tiling of default boxes so that specific feature maps learn to be responsive to particular scales of the objects.

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. 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.

Data augmentation To make the model more robust to various input object sizes and shapes, each training image is randomly sampled by one of the following options:

- Use the entire original input image.
- Sample a patch so that the minimum jaccard overlap with the objects is 0.1, 0.3, 0.5, 0.7, or 0.9.
- Randomly sample a patch.

The size of each sampled patch is $[0.1, 1]$ of the original image size, and the aspect ratio is between 1 and 2. We keep the overlapped part of the ground truth box if the center of it is in the sampled patch. After the aforementioned sampling step, each sampled patch is resized to fixed size and is horizontally flipped with probability of 0.5, in addition to applying some photo-metric distortions.

Multiple output layers at different resolutions is better. A major contribution of SSD is using default boxes of different scales on different output layers. To measure the advantage gained, we progressively remove layers and compare results. For a fair comparison, every time we remove a layer, we adjust the default box tiling to keep the total number of boxes similar to the original (8732). This is done by stacking more scales of boxes on remaining layers and adjusting scales of boxes if needed. We do not exhaustively optimize the tiling for each setting. Table 3 shows a decrease in accuracy with fewer layers, dropping monotonically from 74.3 to 62.4. When we stack boxes of multiple scales on a layer, many are on the image boundary and need to be handled carefully. We tried the strategy used

in Faster R-CNN, ignoring boxes which are on the boundary. We observe some interesting trends. For example, it hurts the performance by a large margin if we use very coarse feature maps (e.g. conv11 2 (1 1) or conv10 2 (33)). The reason might be that we do not have enough large boxes to cover large objects after the pruning. When we use primarily finer resolution maps, the performance starts increasing again because even after pruning a sufficient number of large boxes remains. If we only use conv7 for prediction, the performance is the worst, reinforcing the message that it is critical to spread boxes of different scales over different layers. Besides, since our predictions do not rely on ROI pooling as in , we do not have the collapsing bins problem in low-resolution feature maps . The SSD architecture combines predictions from feature maps of various resolutions to achieve comparable accuracy to Faster R-CNN, while using lower resolution input images.

3.2.1 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of

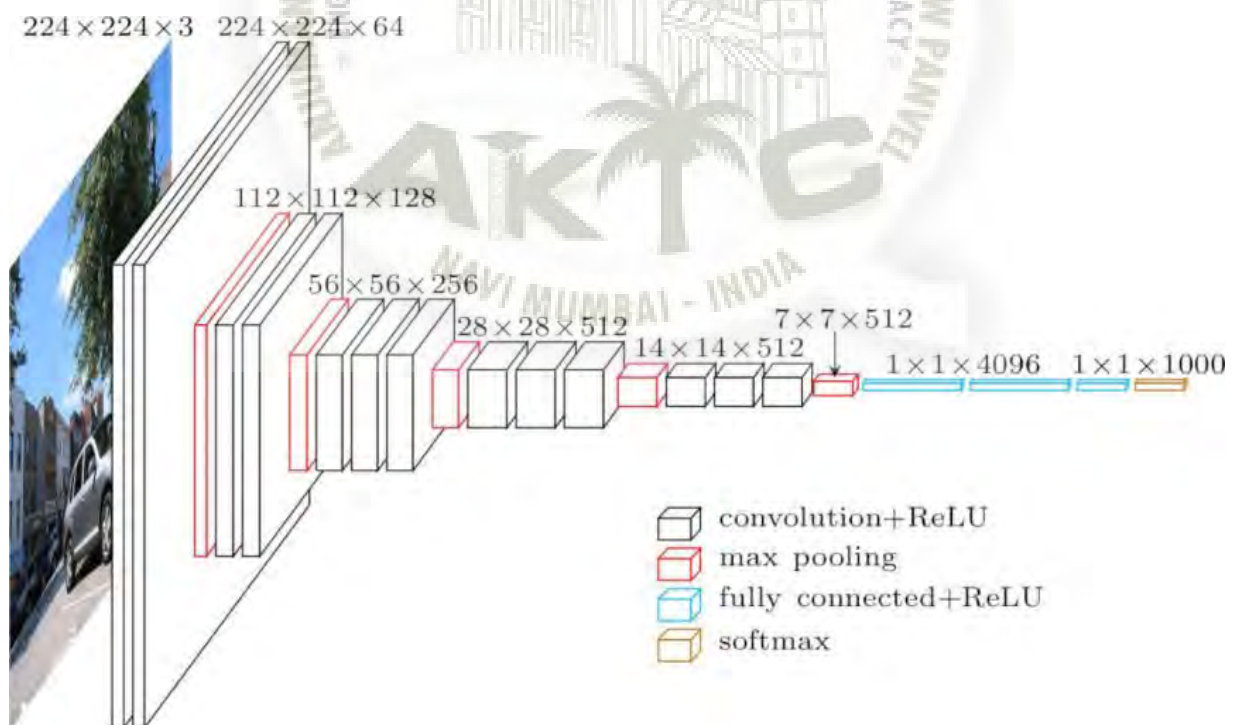


Fig 3.2.1.1 VGG16 architecture

the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3x3 kernel-sized filters one after another. The architecture depicted below is VGG16.

The input to conv1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3x3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 11 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3x3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2x2 pixel window, with stride 2. Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

3.2.2 MultiBox

The bounding box regression technique of SSD is inspired by Szegedy's work on MultiBox, a method for fast class-agnostic bounding box coordinate proposals. Interestingly, in the work done on MultiBox an Inception-style convolutional network is used. The 1x1 convolutions that you see below help in dimensionality reduction since the number of dimensions will go down (but "width" and "height" will remain the same).

MultiBox's loss function also combined two critical components that made their way into SSD:

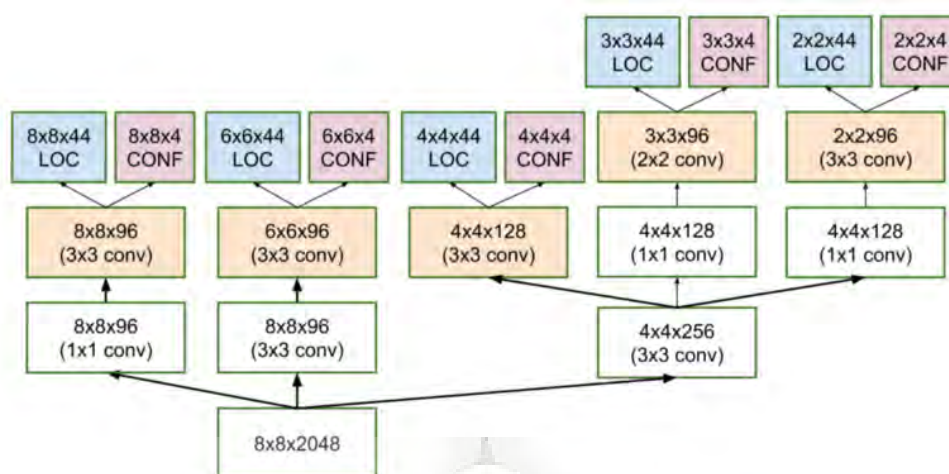


Fig 3.2.2.1 Architecture of multi-scale convolutional prediction of the location and confidences of multibox

Data augmentation To make the model more robust to various input object sizes and shapes, each training image is randomly sampled by one of the following options:

- Confidence Loss: this measures how confident the network is of the objectness of the computed bounding box. Categorical cross-entropy is used to compute this loss.
- Location Loss: this measures how far away the network's predicted bounding boxes are from the ground truth ones from the training set. L2-Norm is used here.

Without delving too deep into the math (read the paper if you are curious and want a more rigorous notation), the expression for the loss, which measures how far off our prediction “landed”, is thus: $\text{multiboxloss} = \text{confidenceloss} + \alpha * \text{locationloss}$. The alpha term helps us in balancing the contribution of the location loss. As usual in deep learning, the goal is to find the parameter values that most optimally reduce the loss function, thereby bringing our predictions closer to the ground truth.

1. MultiBox Priors And IoU

The logic revolving around the bounding box generation is actually more complex than what I earlier stated. But fear not: it is still within reach. In MultiBox, the researchers created what we call priors (or anchors in Faster-R-CNN terminology), which are pre-computed, fixed size bounding boxes that closely match the distribution of the original ground truth

boxes. In fact those priors are selected in such a way that their Intersection over Union ratio (aka IoU, and sometimes referred to as Jaccard index) is greater than 0.5. As you can infer from the image below, an IoU of 0.5 is

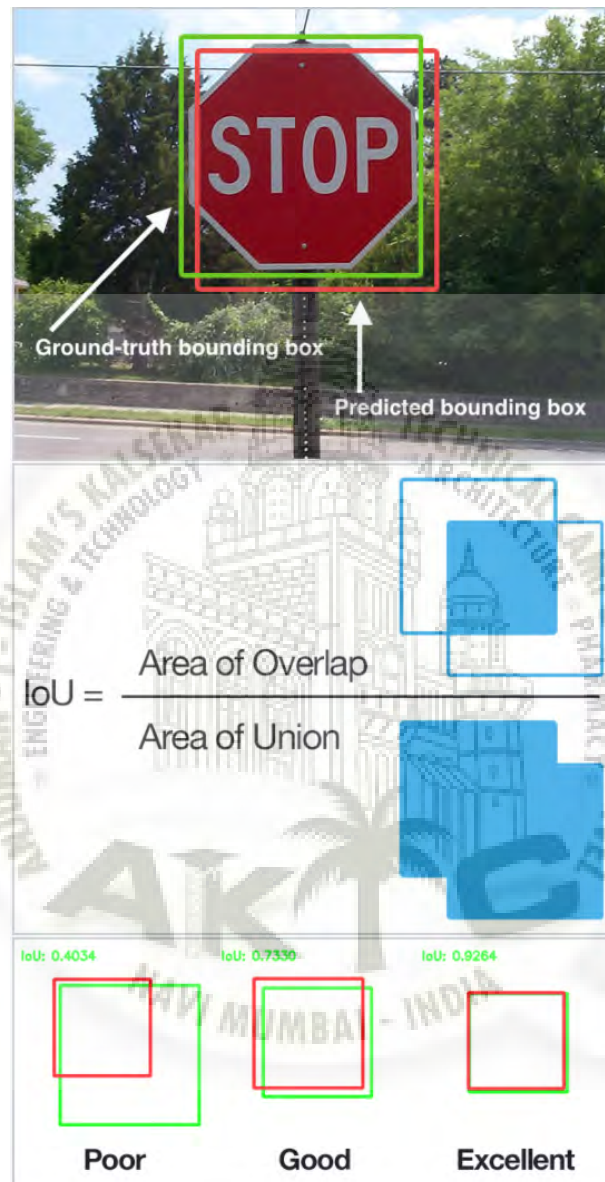


Fig 3.2.2.2 IOU

still not good enough but it does however provide a strong starting point for the bounding box regression algorithm, it is a much better strategy than starting the predictions with random coordinates. Therefore MultiBox starts with the priors as predictions and attempt to regress closer to the ground truth bounding boxes.

The resulting architecture (check MultiBox architecture diagram above again for reference) contains 11 priors per feature map cell (8x8, 6x6, 4x4, 3x3, 2x2) and only one on the 1x1 feature map, resulting in a total of 1420 priors per image, thus enabling robust coverage of input images at multiple scales, to detect objects of various sizes. At the end, MultiBox only retains the top K predictions that have minimised both location (LOC) and confidence (CONF) losses.

2. Feature Maps

Features maps (i.e. the results of the convolutional blocks) are a representation of the dominant features of the image at different scales, therefore running MultiBox on multiple feature maps increases the likelihood of any object (large and small) to be eventually detected, localized and appropriately classified. The image below shows how the network “sees” a given image across its feature maps:

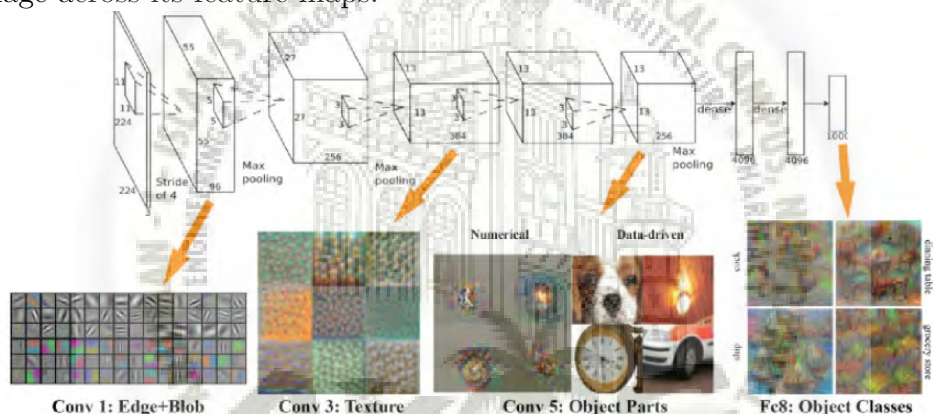


Fig 3.2.2.3 IOU

3. Hard Negative Mining

During training, as most of the bounding boxes will have low IoU and therefore be interpreted as negative training examples, we may end up with a disproportionate amount of negative examples in our training set. Therefore, instead of using all negative predictions, it is advised to keep a ratio of negative to positive examples of around 3:1. The reason why you need to keep negative samples is because the network also needs to learn and be explicitly told what constitutes an incorrect detection.

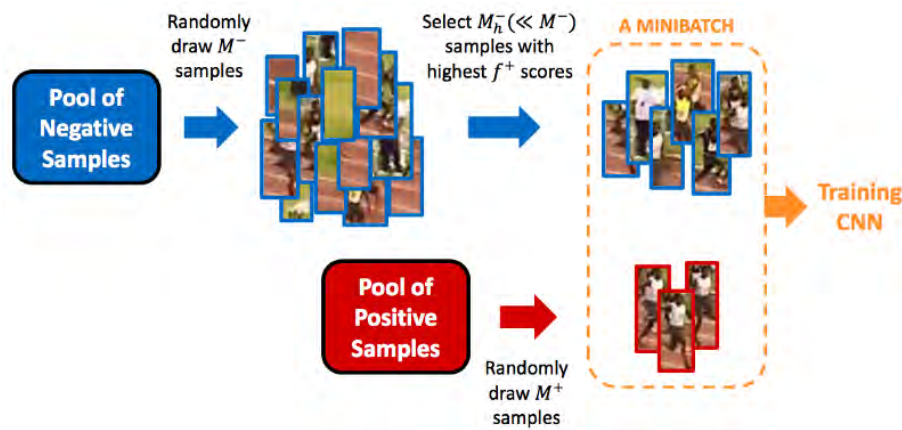


Fig 3.2.2.4 Hard Negative Mining

4. Non-Maximum Suppression (NMS)

Given the large number of boxes generated during a forward pass of SSD at inference time, it is essential to prune most of the bounding boxes by applying a technique known as non-maximum suppression: boxes with a



Fig 3.2.2.5 NMS

confidence loss threshold less than ct (e.g. 0.01) and IoU less than lt (e.g. 0.45) are discarded, and only the top N predictions are kept. This ensures only the most likely predictions are retained by the network, while the more noisier ones are removed.

Chapter 4

Proposed Methodology

4.1 Block Diagram

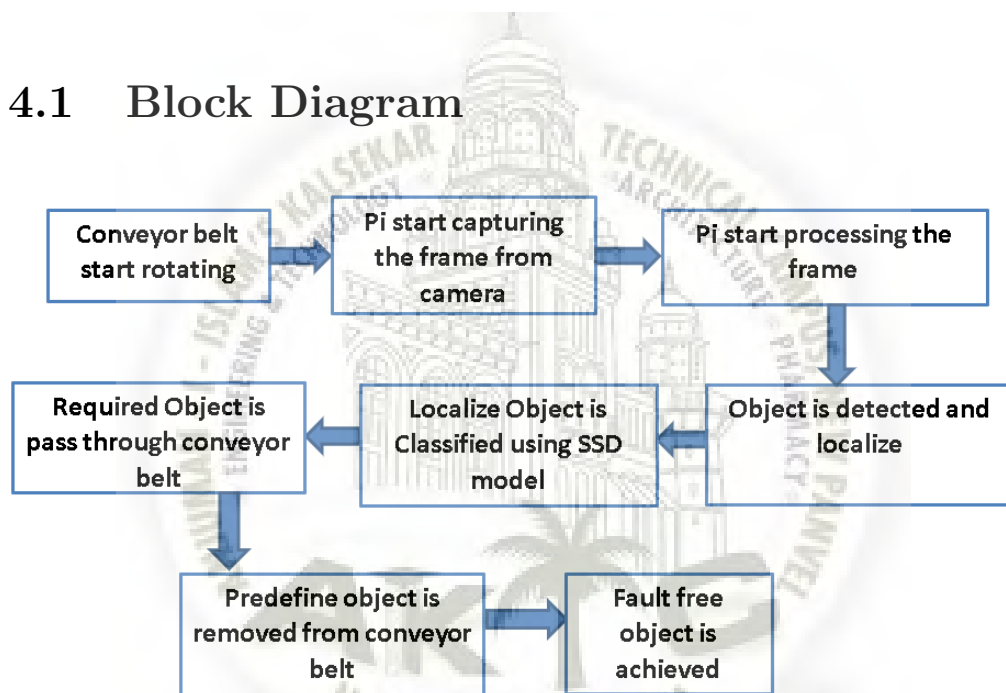


Fig 4.1 Block Diagram

- On the above block diagram, when a conveyor belt start rotating the object the camera start capturing the frame.
- Once the frame is captured, Raspberry Pi start processing the frame using Tensorflow and Open-CV.
- Object is detected and localize using box on the frame.
- After the localization the object is classified using Single Shot Multi-Box model.
- Once the object is identified, the program will extract the predefined object from conveyor belt.

- Thus, quality of the object in an organization is verified and unwanted object is extracted from the system.

4.2 System Flow Diagram

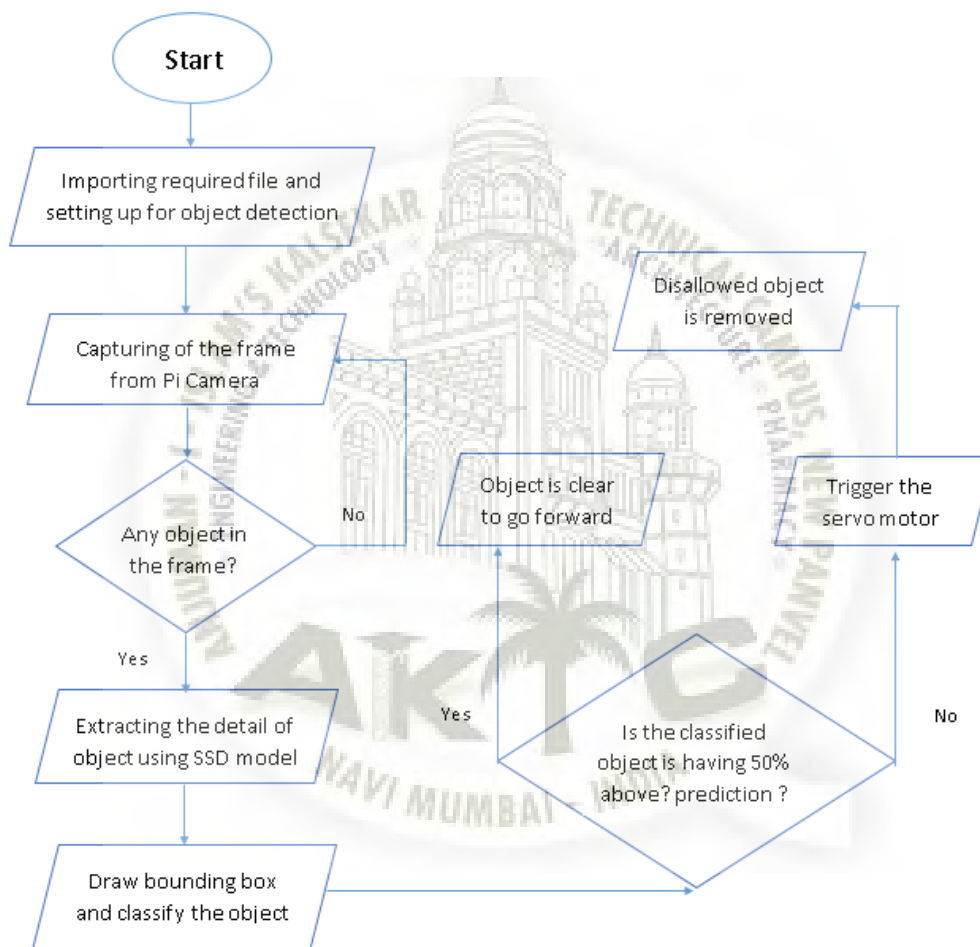


Fig 4.2 System Flow Diagram

- When program is initiated object start moving on a conveyor belt.
- Pi camera start capturing the frame.
- When Card is capture by the camera in the frame,localizing of the card is done.

- Features of the card is extracted using VGG-16 layer.
- Object is detect using pre trained SDD model for playing card detection by filtering from number of convolution layer.
- Some card is disallowed and predefine in program.
- If the card wanted to extract from conveyor belt it is predefine in program.
- We can also set prediction percentage threshold value to remove the lower predicted cards from the conveyor belt.
- If unwanted card is detected the servo motor is activated and pusher mechanism will push the card from the conveyor belt.

4.3 Project Requirements

4.3.1 Software Requirements

1. Tensorflow

TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

2. Anaconda environment with python 3.6

Anaconda is a free and open source distribution of the Python and R programming languages for data science and machine learning related applications (large-scale data processing, predictive analytics, scientific computing), that aims to simplify package management and deployment. Package versions are managed by the package management systemconda. The Anaconda distribution is used by over 6 million users, and it includes more than 250 popular data science packages suitable for Windows, Linux, and MacOS.

3. LabelImg

LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. LabelImg is a great tool for labelling images. LabelImg saves a .xml file containing the label data for each image. These .xml files will be used to generate TFRecords, which are one of the inputs to the TensorFlow trainer.

4. Python Libraries

- Jupyter

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

- Lxml

Python lxml is an easy to use and feature rich library to process and parse XML and HTML documents. lxml is really nice API as it provides literally everything to process these 2 types of data. The two main points which make lxml stand out are: Ease of use: It has very easy syntax than any other library present. Performance: Processing even large XML files takes very less time.

- Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code.

- Pandas

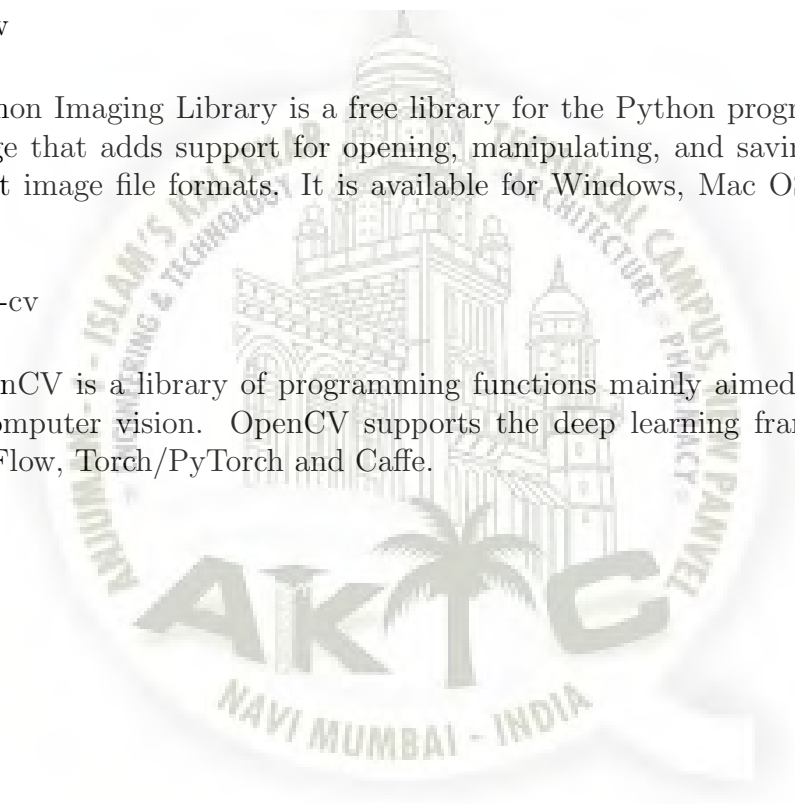
Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

- Pillow

Python Imaging Library is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

- Open-cv

OpenCV is a library of programming functions mainly aimed at real-time computer vision. OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.



4.3.2 Hardware Requirements

1. Raspberry Pi 3 Model B+



Fig 4.1 Raspberry Pi

Specifications:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN,
- Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- 4-pole stereo output and composite video port

- Micro SD port for loading your operating system and storing data
- 5V/2.5A DC power input
- Power-over-Ethernet (PoE) support (requires separate PoE HAT)

2. Pi Camera



Fig 4.2 Pi Camera Module

The Raspberry Pi Camera Board plugs directly into the CSI connector on the Raspberry Pi. It's able to deliver a crystal clear 5MP resolution image, or 1080p HD video recording at 30fps. Custom designed and manufactured by the Raspberry Pi Foundation in the UK, the Raspberry Pi Camera Board features a 5MP (2592x1944 pixels) Omnivision 5647 sensor in a fixed focus module. The module attaches to Raspberry Pi, by way of a 15 Pin Ribbon Cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI), which was designed especially for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the BCM2835 processor. The board itself is tiny, at around 25mm x 20mm x 9mm, and weighs just over 3g, making it perfect for mobile or other applications where size and weight are important. The sensor itself has a native resolution of 5 megapixel, and has a fixed focus lens onboard. In terms of still images, the camera is capable of 2592 x 1944 pixel static images, and also supports 1080p @ 30fps, 720p @ 60fps and

640x480p 60/90 video recording. The camera is supported in the latest version of Raspbian, the Raspberry Pi's preferred operating system. The Raspberry Pi Camera Board Features:

- 5MP Omnivision 5647 Camera Module
- Still Picture Resolution: 2592 x 1944
- Video: Supports 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 Recording
- 15-pin MIPI Camera Serial Interface - Plugs Directly into the Raspberry Pi Board
- Size: 20 x 25 x 9mm
- Weight 3g

3. Conveyor belt



Fig 4.3 Conveyor Belt

A conveyor belt is the carrying medium of a belt conveyor system (often shortened to belt conveyor). A belt conveyor system is one of many types of conveyor systems. A belt conveyor system consists of two or more pulleys

(sometimes referred to as drums), with an endless loop of carrying medium the conveyor belt that rotates about them. One or both of the pulleys are powered, moving the belt and the material on the belt forward. The powered pulley is called the drive pulley while the unpowered pulley is called the idler pulley. There are two main industrial classes of belt conveyors; Those in general material handling such as those moving boxes along inside a factory and bulk material handling such as those used to transport large volumes of resources and agricultural materials, such as grain, salt, coal, ore, sand, overburden and more.



Chapter 5

Results & Discussion

5.1 Implementation

Training of specific model is divided into following steps:

1. Installing TensorFlow
2. Setting up the Object Detection directory structure and Anaconda Virtual Environment
3. Gathering and labeling pictures
4. Generating training data
5. Creating a label map and configuring training
6. Training
7. Exporting the inference graph

1. TensorFlow-GPU allows your PC to use the video card to provide extra processing power while training. Using TensorFlow-GPU instead of regular TensorFlow reduces training time by a factor of about 8 (3 hours to train instead of 24 hours). TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

2. The TensorFlow Object Detection API requires using the specific directory structure provided in its GitHub repository. It also requires several additional Python packages, specific additions to the PATH and PYTHON-PATH variables, and a few extra setup commands to get everything set up to run or train an object detection model.

TensorFlow provides several object detection models (pre-trained classifiers with specific neural network architectures) in its model zoo. Some models (such as the SSD-MobileNet model) have an architecture that allows for faster detection but with less accuracy, while some models (such as the Faster-RCNN model) give slower detection but with more accuracy. You can choose which model to train your object detection classifier on. If you are planning on using the object detector on a device with low computational power (such as a smart phone or Raspberry Pi), use the SSD-MobileNet model. If you will be running your detector on a decently powered laptop or desktop PC, use one of the RCNN models.



Fig 5.1 Comparison between faster r-cnn and single shot multibox detector

Set up new Anaconda virtual environment

Next, we'll work on setting up a virtual environment in Anaconda for tensorflow.

In the command terminal that pops up, create a new virtual environment called "tensorflow1" by issuing the following command:

C: conda create -n tensorflow1 pip python=3.5 Then, activate the environment by issuing:

C: activate tensorflow1 Install tensorflow-gpu in this environment by issuing: (tensorflow1) C: pip install --ignore-installed --upgrade tensorflow-gpu

Install the other necessary packages by issuing the following commands:

(tensorflow1) C: conda install -c anaconda protobuf

(tensorflow1) C: pip install pillow

(tensorflow1) C: pip install lxml

(tensorflow1) C: pip install Cython

(tensorflow1) C: pip install jupyter

(tensorflow1) C: pip install matplotlib

(tensorflow1) C: pip install pandas

(tensorflow1) C: pip install opencv-python

A PYTHONPATH variable must be created that points to the /models, /models/research, and /models/research/slim directories. Do this by issuing the following commands (from any directory):

(tensorflow1) C: set PYTHONPATH=C:/tensorflow1/models;

C:/tensorflow1/models/research;C:/tensorflow1/models/research/slim

Next, compile the Protobuf files, which are used by TensorFlow to configure model and training parameters. Unfortunately, the short protoc compilation command posted on TensorFlow's Object Detection API installation page does not work on Windows. Every .proto file in the /object-detection/protos directory must be called out individually by the command.

3. Gather and Label Picture

TensorFlow needs hundreds of images of an object to train a good detection classifier. To train a robust classifier, the training images should have random objects in the image along with the desired objects, and should have a variety of backgrounds and lighting conditions. There should be some images where the desired object is partially obscured, overlapped with something else, or only halfway in the picture. For my Pinochle Card Detection classifier, I have six different objects I want to detect (the card ranks nine, ten, jack, queen, king, and ace – I am not trying to detect suit, just rank). I used my iPhone to take about 40 pictures of each card on its own, with various other non-desired objects in the pictures. Then, I took about another 100 pictures with multiple cards in the picture. I know I want to be able to detect the cards when they're overlapping, so I made sure to have the cards be overlapped in many images.

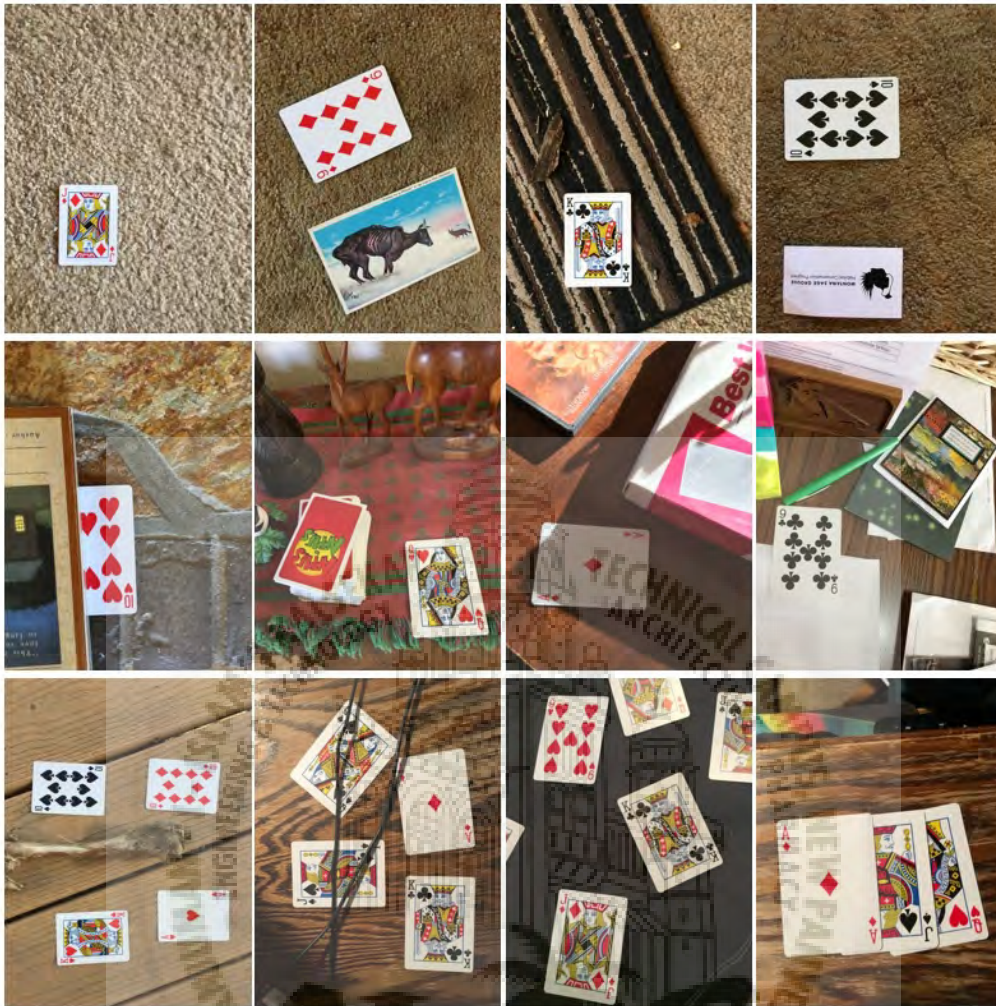


Fig 5.2 Sample Images

You can use your phone to take pictures of the objects or download images of the objects from Google Image Search. I recommend having at least 200 pictures overall. Make sure the images aren't too large. They should be less than 200KB each, and their resolution shouldn't be more than 720x1280. The larger the images are, the longer it will take to train the classifier.

After you have all the pictures you need, move 20 percentage for testing and 80 percentage for training. Make sure there are a variety of pictures in both the test and train directories.

Labellmg is a great tool for labeling images. Download and install LabelImg and then draw a box around each object in each image. Repeat the process for all the images.

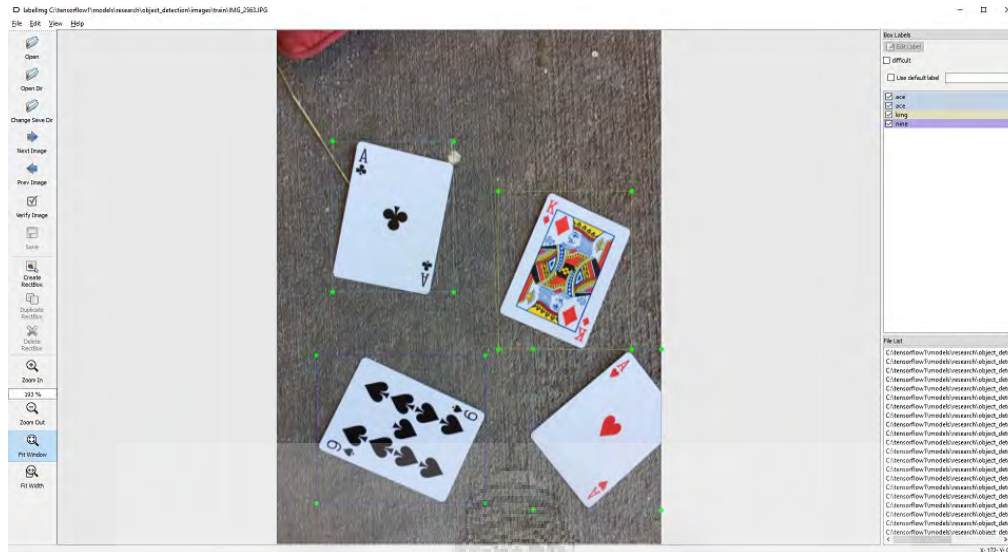


Fig 5.3 Labelling sample image

LabellingImg saves a .xml file containing the label data for each image. These .xml files will be used to generate TFRecords, which are one of the inputs to the TensorFlow trainer. Once you have labeled and saved each image, there will be one .xml file for each image in the /test and /train directories. With the images labeled, it's time to generate the TFRecords that serve as input data to the TensorFlow training model. First, the image .xml data will be used to create .csv files containing all the data for the train and test images. Then generate the TFRecord files for test and train images.

5. Create Label Map and Configure Training The label map tells the trainer what each object is by defining a mapping of class names to class ID numbers. Finally, the object detection training pipeline must be configured. It defines which model and what parameters will be used for training. This is the last step before running training.

6. Run the Training Each step of training reports the loss. It will start high and get lower and lower as training progresses. For my training on the Faster-RCNN-Inception-V2 model, it started at about 3.0 and quickly dropped below 0.8. I recommend allowing your model to train until the loss consistently drops below 0.05, which will take about 40,000 steps, or about 2 hours (depending on how powerful your CPU and GPU are). Note: The loss numbers will be different if a different model is used. MobileNet-SSD starts with a loss of about 20, and should be trained until the loss is consistently under 2.


```

C:\WINDOWS\system32\cmd.exe - python train.py --logtostderr --train_dir=training/ --pipeline_config_path=training/faster_rcnn_inception_v2_pets.c...
[device.cc:1195] Creating TensorFlow device (/device:GPU:0) -> (device: 0, name: GeForce GTX 1060 6GB, pci bus id: 0000:01:00:00, compute capability: 6.1)
INFO:tensorflow:Restoring parameters from C:/tensorflow1/models/research/object_detection/faster_rcnn_inception_v2_coco_2018_01_28/model.ckpt
INFO:tensorflow:Starting Session.
INFO:tensorflow:Saving checkpoint to path training/model.ckpt
INFO:tensorflow:Starting Queues.
INFO:tensorflow:global_step/sec: 0
INFO:tensorflow:Recording summary at step 0.
INFO:tensorflow:global step 1: loss = 2.6708 (5.383 sec/step)
INFO:tensorflow:global step 2: loss = 3.0352 (0.251 sec/step)
INFO:tensorflow:global step 3: loss = 3.4884 (0.204 sec/step)
INFO:tensorflow:global step 4: loss = 2.9733 (0.193 sec/step)
INFO:tensorflow:global step 5: loss = 2.2184 (0.191 sec/step)
INFO:tensorflow:global step 6: loss = 2.0321 (0.554 sec/step)
INFO:tensorflow:global step 7: loss = 2.0424 (0.211 sec/step)
INFO:tensorflow:global step 8: loss = 2.0252 (0.208 sec/step)
INFO:tensorflow:global step 9: loss = 2.0053 (0.194 sec/step)
INFO:tensorflow:global step 10: loss = 1.3622 (0.193 sec/step)
INFO:tensorflow:global step 11: loss = 1.8027 (0.197 sec/step)
INFO:tensorflow:global step 12: loss = 1.2485 (0.196 sec/step)
INFO:tensorflow:global step 13: loss = 1.0712 (0.193 sec/step)
INFO:tensorflow:global step 14: loss = 1.6604 (0.189 sec/step)
INFO:tensorflow:global step 15: loss = 1.2657 (0.192 sec/step)
INFO:tensorflow:global step 16: loss = 1.4351 (0.193 sec/step)
INFO:tensorflow:global step 17: loss = 1.2152 (0.192 sec/step)
INFO:tensorflow:global step 18: loss = 1.1165 (0.197 sec/step)
INFO:tensorflow:global step 19: loss = 1.6557 (0.192 sec/step)
INFO:tensorflow:global step 20: loss = 1.7777 (0.200 sec/step)

```

Fig 5.4 Training of model

The training routine periodically saves checkpoints about every five minutes. You can terminate the training by pressing Ctrl+C while in the command prompt window. I typically wait until just after a checkpoint has been saved to terminate the training. You can terminate training and start it later, and it will restart from the last saved checkpoint. The checkpoint at the highest number of steps will be used to generate the frozen inference graph.

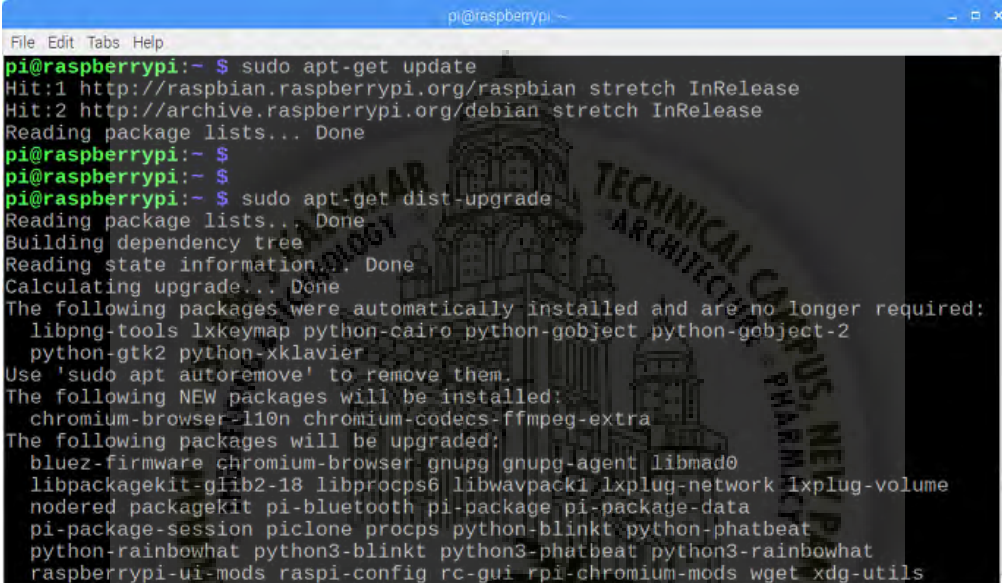
7. Export Inference Graph Now that training is complete, the last step is to generate the frozen inference graph (.pb file). From the /objectdetection folder, issue the following command, where “XXXX” in “model.ckpt-XXXX” should be replaced with the highest-numbered .ckpt file. The .pb file contains the object detection classifier.

In raspberry Pi 3 we have implemented object detection of model ssdlite mobilenet v2 coco using tensorflow, OpenCv and Protobuf. Pi camera is used as camera model. Running of object detection on raspberry pi is divided into following steps:

Steps:

1. Update the Raspberry Pi
2. Install TensorFlow
3. Install OpenCV
4. Compile and install Protobuf
5. Set up TensorFlow directory structure and the PYTHONPATH variable
6. Detecting objects

1. Update the Raspberry Pi First, the Raspberry Pi needs to be fully updated. Open a terminal and issue: `sudo apt-get update` `sudo apt-get dist-upgrade` Depending on how long it's been since you've updated your Pi, the upgrade could take anywhere between a minute and an hour.



```

pi@raspberrypi:~$ sudo apt-get update
Hit:1 http://raspbian.raspberrypi.org/raspbian stretch InRelease
Hit:2 http://archive.raspberrypi.org/debian stretch InRelease
Reading package lists... Done
pi@raspberrypi:~$ sudo apt-get dist-upgrade
Reading package lists... Done
Building dependency tree
Reading state information... Done
Calculating upgrade... Done
The following packages were automatically installed and are no longer required:
  libpng-tools lxkeymap python-cairo python-gobject python-gobject-2
  python-gtk2 python-xklavier
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  chromium-browser-l10n chromium-codecs-ffmpeg-extra
The following packages will be upgraded:
  bluez-firmware chromium-browser gnupg gnupg-agent libmad0
  libpackagekit-glib2-18 libprocps6 libwavpack1 lxplug-network lxplug-volume
  nodered packagekit pi-bluetooth pi-package pi-package-data
  pi-package-session piclone procps python-blinkt python-phatbeat
  python-rainbowhat python3-blinkt python3-phatbeat python3-rainbowhat
  raspberrypi-ui-mods raspi-config rc-gui rpi-chromium-mods wget xdg-utils

```

Fig 5.5 Upgrading Raspberry Pi

2. Install TensorFlow A pre-built, Raspberry Pi-compatible wheel file for installing the latest version of TensorFlow is available in the “TensorFlow for ARM” GitHub repository. GitHub user lhelontra updates the repository with pre-compiled installation packages each time a new TensorFlow is released. TensorFlow also needs the LibAtlas package. Also install other libraries pillow, lxml, jupyter, matplotlib and cython.

3. Install Open-CV TensorFlow’s object detection examples typically use matplotlib to display images, but I prefer to use Open-CV because it’s easier to work with and less error prone. To get OpenCV working on the Raspberry Pi, there’s quite a few dependencies that need to be installed through apt-get before running the object detection in python.

4. Compile and Install Protobuf The TensorFlow object detection API uses Protobuf, a package that implements Google's Protocol Buffer data format. We have to compile it from source ourselves and then install it.

5. Set up TensorFlow Directory Structure and PYTHONPATH Variable we need to modify the PYTHONPATH environment variable to point at some directories inside the TensorFlow repository. We want PYTHONPATH to be set every time we open a terminal.

6. Detect Objects.



Fig 5.6 Object Detection using COCO database 1



Fig 5.7 Object Detection using COCO database 2



Fig 5.8 Object Detection using COCO database 3

5.2 Result



Fig 6.1 Object Detection of Playing Cards

The object is detection playing cards. It is detecting the Ace, King, Queen, Jack and numbers from 2 to 10. We have applied 3 cases, 1st we put white background with 32cm distance between camera module and ground, 2nd we put black background with 32cm distance between camera module and ground and 3rd we put black background with 24cm distance between camera module and ground. We are getting 73.08% accuracy in first case, 59.62% accuracy in second case and 90.39% accuracy in third case. As the ground and camera distance is increased the accuracy is decreased because of smaller object on the frame.

Some of the perfect matching are:



Fig 6.2 Right Prediction of Cards

Some of the wrong predictions are:



Fig 6.3 Wrong Prediction of Cards

Cards Prediction Percentages in all cases :

| Cards | Spade | | | Club | | | Diamond | | | Heart | | |
|-------|-------|-------|-------|-------|-------|-------|---------|-------|-------|-------|-------|-------|
| | Case1 | Case2 | Case3 | Case1 | Case2 | Case3 | Case1 | Case2 | Case3 | Case1 | Case2 | Case3 |
| Ace | 97% | 99% | 99% | 86% | 99% | 99% | 93% | 99% | 99% | 96% | 99% | 99% |
| 2 | 0% | 51% | 99% | 91% | 99% | 99% | 46% | 95% | 89% | 45% | 85% | 94% |
| 3 | 57% | 0% | 99% | 65% | 49% | 99% | 53% | 0% | 95% | 0% | 0% | 0% |
| 4 | 92% | 57% | 99% | 96% | 61% | 99% | 85% | 99% | 99% | 67% | 73% | 99% |
| 5 | 0% | 0% | 99% | 47% | 42% | 98% | 55% | 75% | 99% | 88% | 97% | 99% |
| 6 | 76% | 0% | 99% | 98% | 0% | 98% | 0% | 0% | 0% | 0% | 0% | 0% |
| 7 | 87% | 86% | 99% | 91% | 0% | 99% | 76% | 85% | 99% | 97% | 91% | 99% |
| 8 | 96% | 87% | 99% | 94% | 99% | 98% | 87% | 53% | 96% | 84% | 0% | 97% |
| 9 | 94% | 43% | 99% | 0% | 0% | 82% | 0% | 80% | 98% | 0% | 0% | 97% |
| 10 | 95% | 65% | 99% | 0% | 82% | 0% | 67% | 99% | 99% | 0% | 42% | 93% |
| king | 45% | 5% | 0% | 0% | 0% | 0% | 0% | 0% | 98% | 0% | 0% | 99% |
| Queen | 51% | 93% | 99% | 92% | 98% | 99% | 56% | 99% | 99% | 73% | 98% | 99% |
| Jack | 41% | 0% | 96% | 65% | 0% | 89% | 93% | 71% | 81% | 0% | 0% | 99% |

Fig 6.4

Graph of Cards Prediction in all cases :

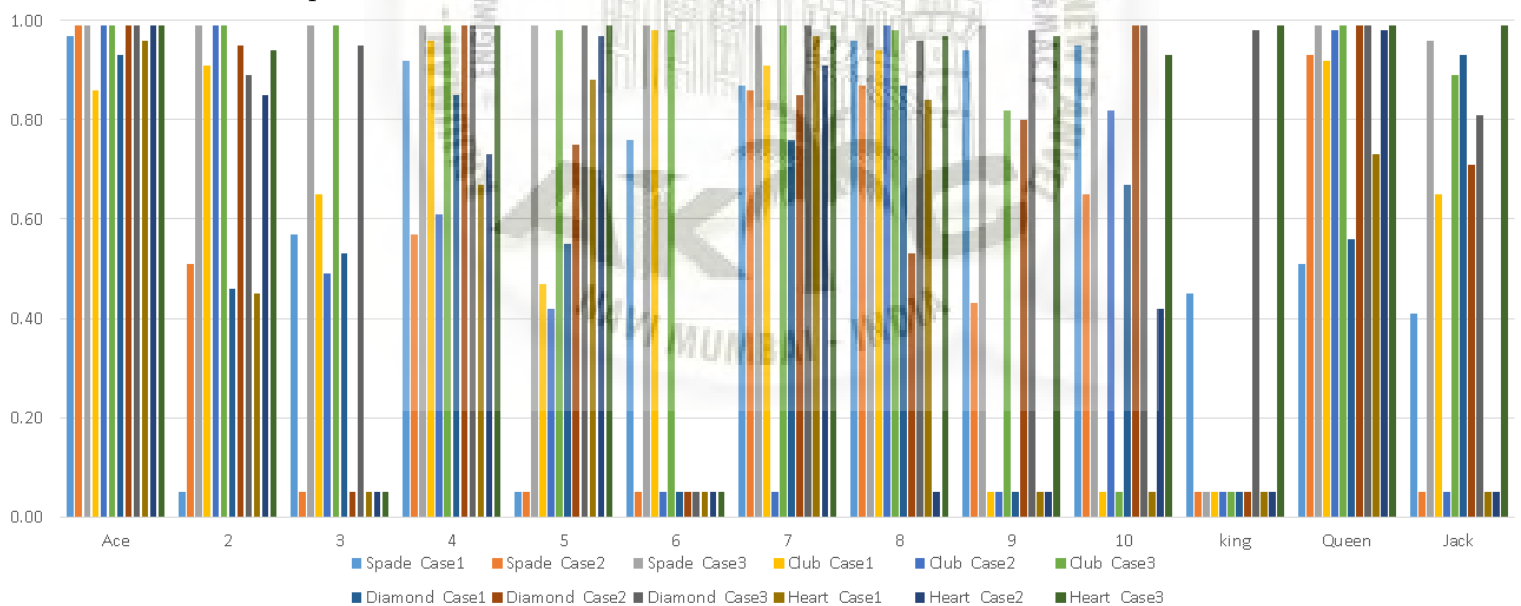


Fig 6.5

Comparison between cards categories and all cases :

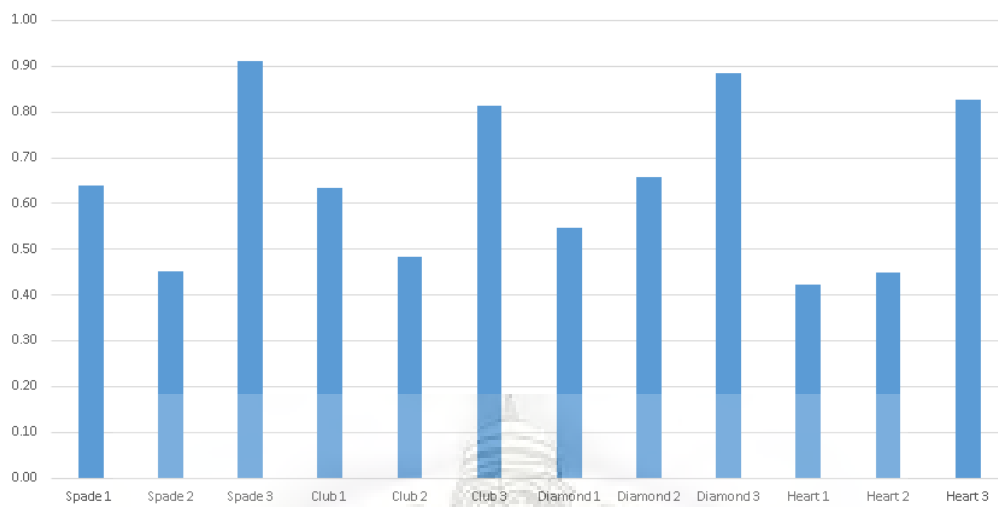


Fig 6.6

Prediction between all cases :

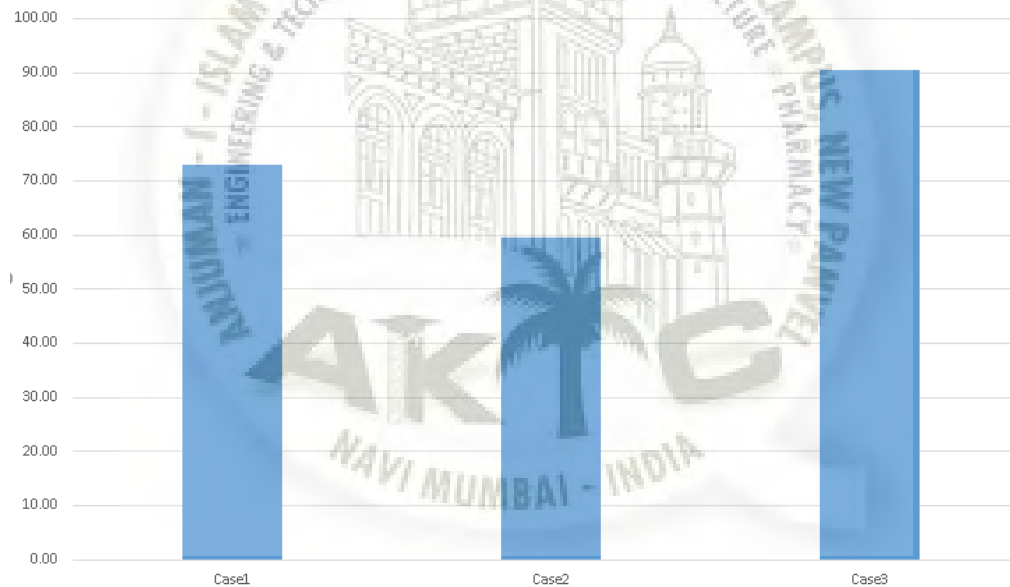


Fig 6.7

Chapter 6

Conclusion & Future Scope

6.1 Conclusion

In this project, the predefine or defect product is removed from the system using SSD model of object detection. The product defects are in the form of unwanted object is extract by scanning the object and extracting the information from the frames. Using case 3 we achieved 90.39% accuracy and frame rate of 0.8 to 1.0 . This project is also successful in exploring and implementing Multi-task learning to solve class problems of classifying different card.

6.2 Future Scope

In order to improve the detection performance, it is imperative to replace VGG by more effective networks, such as ResNet and DenseNet.Future work will be focused on development and integration of how to improve the inference speed of these deep backbones and to increase frame rate per second for real time processing. In addition, there are still some false and omissive detections in our visualized results. Future work will also be focused on development and integration of multiple features from equal area normalization descriptor to further strengthen it in terms of accuracy and efficiency.

References

- [1] P.Arjun1, T.T.Mirnalinee, "Machine parts recognition and defect detection in automated assembly systems using computer vision techniques", Rev. Téc. Ing. Univ. Zulia. Vol. 39, No 1, 71 - 80, 2016.
- [2] NursabillilahMohdAlie, MohdSafirin Karis et al., "Quality Checking And Inspection Based On Machine Vision Technique To Determine Tolerancevalue Using Single Ceramic Cup", ARPN Journal of Engineering and Applied Sciences, VOL. 12, NO. 8, APRIL 2017.
- [3] Rasika R. Chavan, Swati A. Chavan et al., "Quality Control of PCB using Image Processing", International Journal of Computer Applications (0975 – 8887) Volume 141 – No.5, May 2016.
- [4] Aqsa Khan, Mr. N. J. Janw, "Review on Moving Object Detection in Video Surveillance", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 6, Issue 5, May 2017.
- [5] Athanasios Voulodimos, Nikolaos Doulamis et al., "Deep Learning for Computer Vision: A Brief Review", Computational Intelligence and Neuroscience, Volume 2018, Article ID 7068349, 13 pages.
- [6] Sheng Ding,, Kun Zhao, "Research on Daily Objects Detection Based on Deep Neural Network", Materials Science and Engineering 322 ,2018.
- [7] Shaoqing Ren, Kaiming He, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", Computer Vision and Pattern Recognition, Vol 3, 4 Jun 2015.
- [8] Wei Liu, Dragomir Anguelov et al., "SSD:Single Shot MultiBox Detector", Computer Vision and Pattern Recognition, Volume 2018.
- [9] Ryan Joshua H. Liwag, Kevin Jeff T. Cepria et al., "Single Shot Multi-Box Detector with Multi Task Convolutional Network for Carabao Mango Detection and Classification using Tensorflow", Presented at the 5th DLSU Innovation and Technology,2017.

- [10] Mingliang Xu, Lisha Cui et al., "*MDSSD: Multi-scale Deconvolutional Single Shot Detector for Small Objects*", IEEE Transaction on Image Processing, 2017.
- [11] Girshick, R., Donahue et al., "*Rich feature hierarchies for accurate object detection and semantic segmentation*", CVPR, 2014.
- [12] Simonyan, K., Zisserman et al., "*Very deep convolutional networks for large scale image recognition*", NIPS, 2015.
- [13] Uijlings, J.R et al., "*Selective search for object recognition*", IJCV, 2013.
- [14] Erhan, D., Szegedy, C. et al., "*Scalable object detection using deep neural networks*", CVPR, 2014.
- [15] Long, J., Shelhamer et al., "*Fully convolutional networks for semantic segmentation*", CVPR, 2015.
- [16] S. Song, V. Chandrasekhar et al., "*Multi modal Multi Stream Deep Learning for Egocentric Activity Recognition*", IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015.
- [17] R. Kavi, V. Kulathumani et al., "*Multiview fusion for activity recognition using deep neural networks*", Journal of Electronic Imaging, vol. 25, no. 4, Article ID 043010, 2016.
- [18] Y. LeCun, B. Boser et al., "*Handwritten digit recognition with a back-propagation network*", in Advances in Neural Information Processing Systems 2 (NIPS*89), D. Touretzky, Ed., Denver, CO, USA, 1990.
- [19] N. Doulamis, "*Adaptable deep learning structures for object labelling/tracking under dynamic visual environments*", Multi media Tools and Applications, pp. 1–39, 2017.
- [20] L. Lin, K. Wang et al., "*A deep structured model with radius-margin bound for 3D human activity recognition*", International Journal of Computer Vision, vol. 118, no. 2, pp. 256–273, 2016.
- [21] S. Cao and R. Nevatia, "*Exploring deep learning based solutions in fine grained activity recognition in the wild*", Proceedings of the 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 384–389, Cancun, December 2016.