RETAIL INVENTORY MANAGEMENT USING OBJECT DETECTION

REPORT

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The objective of our project is using real time object detection as the primary means for automating the retail inventory management systems and also for detecting stock-out conditions on shelves. In today's highly competitive retail landscape, being able to control and optimize retail execution at the point of sale has never been more critical. In-store audit solutions today are manual and time-consuming. Auditing takes approximately 15 minutes per product category, involving physical measurements that are prone to human errors and inaccuracies. It is an expensive solution and only a handful of the traditionally conservative FMCG players have embraced progressive technology to achieve this.

While purchase patterns can be obtained from the cash registers, measuring the shelving execution standards is a much more complicated task. The industry's ability to grow and compete has been challenged by infrastructure, increased competition, and most importantly, the absence of effective tracking and analysis tools for retail execution in the stores. With image recognition technology, manufacturers and retailers can now understand the marketplace and react in real-time. Using object detection can result in time-saving of audit process with better accuracy. This means you will have accurate and reliable data on your distribution, know which items are out of stock and the share of your products within the category as well as a wealth of other actionable insights at your fingertips.

Shelf-out-of-stock is one of the leading motivations of technology innovation towards the smart shelf of the future. Traditionally, for stock replenishment, visual check of shelves is performed by store staff, using a task known as planogram compliance monitoring. With image recognition capabilities, the in-store CCTV cameras can be leveraged to detect and count front-facing products at category, range, brand or SKU level to assist the store staff in this tedious process. This innovation takes the human error and processing scalability out of the equation.

Inventory management is another important application of object detection and counting.Manual inventory management often results in multiple errors and in a bid to avoid these errors retailers often end up conducting multiple checks which is both tedious and time consuming. An automated Inventory Management System helps to minimize the risk of error. In retail stores, an Inventory Management System also helps track theft of retail merchandise, providing valuable information about store profits and the need for theft-prevention systems. The idea of using the video camera to count the number of objects has been proposed as a new way of detection and counting approach.The surveillance video from the stores will be utilised by the AI model to detect the products on the shelves and count them using object counting methodologies in the morning before the store opens and there is a count taken at the end of the day post the closing of the shop. This data is stored in an excel file where the rest of the inventory management takes place. This reduces the possibility of errors and is both efficient and time saving.

Accurately counting objects instances in a given image or video frame has been a hard problem to solve in machine learning. Finding a proper solution to the problem of counting objects depends on many factors. Besides some challenges common to all image processing with Neural Networks - like size of the training data, its quality etc. Specific challenges to the counting objects problem are type of the objects to be counted, overlapping, perspective view, the minimum size of detected objects, training and testing speed. There exists a method in the field of machine learning and in Deep Learning with Convolutional Neural Networks in particular, called Region based Convolutional Neural Network (RCNN), where we identify multiple objects and their location on a given image. In this project object detection is performed using Faster R-CNN.

Faster R-CNN introduces a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, enabling nearly cost-free region proposals. The RPN component of this solution tells the unified network where to

look. For the same VGG-16 model, Faster R-CNN has a frame rate of 5 fps on a GPU while achieving state-of-the-art object detection accuracy.

1.2 PROBLEM STATEMENT

Management of a grocery store or supermarket is a challenging task entailing personnel busy in supervising shelves and the whole sale point. Technology advances may be deployed to provide more reliable information in real time to the store manager, so to coordinate human resources more effectively. Examples of tasks where innovation can improve current best practices are shelves analysis (e.g. verifying low in stock or misplaced items), security (e.g. reporting suspicious behaviors) and customer analysis (e.g. analyzing shopping patterns to improve customer experience). However, a promising technological solution can be deployed in real shops as long as it turns out viable from a cost perspective, modifies current practices moderately and does not affect customer experience adversely. Computer vision techniques may fulfil the above requirements due to potential reliance on cheap cameras either mounted noninvasively in the store or embedded within the hand-held computers routinely used by sales clerks.

1.3 OBJECTIVE

The main objective is visual shelf monitoring through computer vision techniques. Our project aims at using real time object detection as the primary means for automating the retail inventory management systems and also for detecting stock-out conditions on shelves. This project adopts an approach in which 24 hours video surveillance of the different aisles of the store continually keep a track of the number of each kind of product that is displayed on the shelves. A method in the field of Deep Learning with Convolutional Neural Networks in particular, called Region based Convolutional Neural Network (RCNN), where we identify multiple objects and their location on a given image. Object detection is performed using Faster R-CNN. This introduces a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, enabling nearly cost-free region proposals. The RPN is a kind of a fully convolutional network that can be trained end-to-end

specifically for the task of generating detection proposals and predict region proposals. Using the data collected from object counting we perform inventory management in excel. In this manner, efficiency and effectiveness of many previously manually-implemented tasks may be improved.

1.4 ORGANIZATION OF THE REPORT

The project which mainly focus on inventory management and stock-out predictions using real time object detection. There are five chapters that deal with the various design and implementation details.

CHAPTER 1

This chapter deals on the objective and overview of the project.

CHAPTER 2

This chapter includes all the features of the existing system and summary of the literature survey.

CHAPTER 3

This chapter deals with the system specification i.e. hardware and the software requirements with description.

CHAPTER 4

This chapter entirely deals about the system implementation on the details about the platforms, working of system and the result.

CHAPTER 5

This chapter deals with the conclusion of the project.

CHAPTER-2

LITERATURE SURVEY

2.1 INTRODUCTION

There is an extensive literature on image processing and object detection and here we explain a few relevant papers. Yun Ren, Changren Zhu, and Shunping Xiao take a more direct approach, experimentally showing how Faster R-CNN replaces selective search with a very small convolutional network called region proposal network to generate regions of interests. Faster R-CNN merges the proposed RPN and fast R-CNN into a single network by sharing their convolutional features.

A video stream or a single image is given as the input and the system is designed to find which object is moving and count the number of moving objects in the scene. This has been done in six stages namely image acquisition, image enhancement, image segmentation, image analysis, object counting and reports. The image is first acquired and then enhanced using gray scaling process which translates the RGB image into gray scale. Image analysis is done by using pattern matching technique which gives the number of objects in the frame. The number of objects detected was collected in the Excel File to be analyzed later. It was concluded that the best level of difference that we can get is between 0.4 and 0.45. From the experimental results, it was found that the maximum percentage of correctness reached by proposed system was 98%. It was also observed that this system came with a certain disadvantage owing to which detection of objects above the height of 5m produces inaccurate results.

2.2 LITERATURE SURVEY

2.2.1 Object Detection Based on Fast/Faster R-CNN Employing Fully Convolutional Architectures (2018).

Yun Ren, Changren Zhu, Shunping Xiao proposed that modern object detectors always include two major parts: a feature extractor and a feature classifier. The deeper and wider convolutional architectures are adopted as a feature extractor. It is beneficial for the detection performance to elaborately design deep convolutional networks of various depths for feature classification, especially using the fully convolutional architectures. This paper also demonstrates how to employ the fully convolutional architectures in the fast/faster R-CNN.Slowest part in fast R-CNN was selective search or edge boxes. Faster R-CNN replaces selective search with a very small convolutional network called region proposal network to generate regions of interests.Faster-R-CNN is 10 times faster than fast-R-CNN with similar accuracy of datasets like voc-2007. That's why faster-R-CNN has been one of the most accurate object detection algorithms. The feature classifier in fast/faster r-R-CNN includes the following 3 aspects. Firstly, the input size of the fully convolutional architectures must satisfy certain condition due to the concatenation of the convolutional and pooling layers. Faster R-CNN merges the proposed RPN and fast R-CNN into a single network by sharing their convolutional features. In Faster R-CNN the ROI pooling layer is not as precisely mapped to the regions of the original image.

2.2.2 Object detection and Counting System (2008).

Chomtip Pornpanomchai, Fuangchar Stheitsthienchai and Sorawat Rattanachuen proposed object detection and counting system using image processing. A video stream or a single image is given as the input and the system is designed to find which object is moving and count the number of moving objects in the scene. This has been done in six stages namely image acquisition, image enhancement, image segmentation, image analysis, object counting and reports. The image is first acquired and then enhanced using gray scaling process which translates the RGB image into gray scale. Image analysis is done by using pattern matching technique which gives the number of objects in the frame. The number of objects detected was collected in the Excel File to be analyzed later. It was concluded that the best level of difference that we can get is between 0.4 and 0.45. From the experimental results, it was found that the maximum percentage of correctness reached by proposed system was 98%. It was also observed that this system came with a certain disadvantage owing to which detection of objects above the height of 5m produces inaccurate results.

2.2.3 Research on Daily Objects Detection Based on Deep Neural Network (2018).

Shen Ding, Kun Zhao proposed that the deep learning algorithm is applied to the detection of daily objects and the main work of this paper includes collecting small dataset of daily objects, TensorFlow framework to build different models of object detection. The training process and effect of the model are improved by fine-tuning the model parameters. The evolution of detection algorithm is divided into two stages. Stage 1 is based on the traditional features of the solution and the second stage is the deep learning algorithm. Feature design methods include artificial feature design and neural network feature extraction. The selection of detection mainly includes exhaustive search, selective search and RPN method based on deep learning. The well trained model developed in this paper can be used in the mobile platform or in other intelligent devices to achieve accurate detection. By comparing the direct training and parameter adjustment model training, it was proven that the convergence speed and accuracy of object detection were improved by adjusting the parameters. With increasing amount of detection data the traditional detection method performance will become saturated. The detection performance will gradually improve, yet the improvement decreases after a certain amount of data.

2.2.4 Detection and counting of Red Blood Cells in Blood Cell Images using Hough Transform (2013).

Mausumi Maitra, Rahul Kumar Gupta and Manali Mukherjee proposed an approach to automatic segmentation and counting of Red Blood Cells in microscopic blood cell images using Hough Transform. The entire process was divided into three stages namely image enhancement and identification of RBC's, Hough Transform for object detection, counting the number of RBC's. Various pre-processing techniques like edge detection, spatial smoothing filtering and adaptive Histogram equalization were used to detect and extract the RBC's from the images. Feature extraction has been done through the Hough transform method which has been used to find out the RBC based on their sizes and shapes. Having successfully isolated the RBC's a counter was applied to count the number of RBC's in the image field. Results indicate that the counting of RBC's using Hough transform method has offered remarkable accuracy. This system is cost-effective and time saving and also it can identify overlapping blood cells and count them separately. However, this system does not count the effective number of RBC's which are partly in the image fields, thereby affecting the accuracy of the result.

2.2.5 Vehicle Detection and Counting System for Real-Time Traffic Surveillance (2018).

Boris A. Alpatov, Pavel V. Babayan, Maksim D. Ershov proposed the development of vehicle detection and counting system with the help of smart cameras using vehicle detection and counting algorithm, road marking detection algorithm. The algorithms are designed to process images obtained from a stationary camera. The work was devoted to solving problems of traffic control systems. It helps in detection of accidents and vehicles stopped in wrong space, compliance monitoring and registration of traffic violations and obtaining of traffic statistics. The final implementation of the developed vehicle detection and counting algorithm is oriented to work on an embedded platform of smart cameras. The algorithm was tested on the network camera AXIS M2026-LE Mk II with Ambarella's ARM-based processor. The processing time for one frame was 4 ms when observing five lanes of the road. This system has a high efficiency that is about 76%. Embedded online video processing reduces the amount of data transmitted to the traffic control centre. The requirements to communication channels and a computing server are reduced. Computing capabilities of such cameras are limited. This system is complex as it is a big system which includes many unites in it like Vehicle Detectors, Intersection Controller, Communication Network, Application Software, and Central (Regional) Control System.

2.2.6 An Efficient Multiple Object Detection and Tracking Framework for Automatic Counting and Video Surveillance Applications (2012)

Carlos R. del-Blanco, Fernando Jaureguizar, and Narciso Garcia proposed novel Bayesian tracking model that can manage multimodal distributions without explicitly computing the association between track objects and detections. It is based on off-the-shelf equipment, such as IP, web cameras, and PCs, and does not especial installation and configuration requirements. The background is modelled pixel by pixel using a mixture of Gaussians that can even represent non static background scenes. This moving object detector has also the ability to detect and remove shadows, great source of false alarms in this kind of visual detectors. It can also perform split detections and merge detections which make the trajectory estimation and object counting challenging task. The Expectation-Maximization algorithm and Levenberg-Marquardt has been modified to achieve two goals. The first one is to limit the number of ellipses per object to one and the second one is to assist in the estimation of the ellipse parameters restricting their values according to a predetermined range of possible objects sizes and orientations. In addition, the two previous adaptations reduce the complexity of the fitting process, making the algorithm more suitable for fulfilled real-time restrictions. However the Bayesian model does not tell you how to select a prior. There is no correct way to choose a prior. Bayesian inferences require skills to translate subjective prior beliefs into a mathematically formulated prior. If you do not proceed with caution, you can generate misleading results. It often comes with a high computational cost, especially in models with a large number of parameters.

2.2.7 Detection of Shapes and Counting In Toy Manufacturing Industry With Help Of Python (2017)

J. Uma, P. Yuvarani proposed work focused on detection of shapes of toy and its count in the particular area to segregate the items manufactured in the toy industry for packing. The identification of shapes using Ramer –Douglas-Peucker algorithm in Python language is a technique implemented with the help of open CV image processing tool. The prototype of the model is developed by giving a sample input image with different shapes as file. The screen in the python language shows the name of the shapes with its count in the input sample file. The proposed idea may extend in tool manufacturing industry to identify the different shapes of part which is difficult to segregate and consume more man power for the particular process.The main focus of the algorithm is used to identify and to detect the number of shapes in the manufacturing industry of the toy shapes. With the help of contour approximation algorithm in Open CV tool used for computer vision development and applications. The Open CV tool is fastest compile tool and its mostly available as open source tool. The code was slow, especially for non-random access iterators, the code had become too complex with all its bookkeeping.

2.2.8 Divide and Count: Generic Object Counting by Image Divisions (2019).

Tobias Stahl, Silvia L. Pintea, and Jan C. Van Gemert proposed a general object counting method that does not use any prior category information. This method separates the input image into a set of image divisions—each fully covering the image. Each image division is composed of a set of region proposals or uniform grid cells. The approach learns in an end-to-end deep learning architecture to predict global image-level counts from local image divisions. The method incorporates a counting layer which predicts object counts in the complete image, by enforcing consistency in counts when dealing with overlapping image regions. The counting layer is based on the inclusion–exclusion principle from set theory. The individual building blocks of this proposed approach was analysed on PascalVOC2007 and the method was evaluated on the MS-COCO large scale generic object data set as well as on three class-specific counting data sets: UCSD pedestrian data set, and CARPK, and PUCPR+ car data sets. The approach on MS-COCO large-scale generic object dataset. The model was trained on the training set of MS-COCO and predict on val. Counting objects on a generic dataset is more challenging than counting objects for a specific class, because the method must be able to count equally well objects that typically appear in large numbers such as sheep and bottles, as well as objects that are photographed alone, such as cats and dogs. If the set contains no repeating values, the mode is irrelevant. In contrast, if there are many values that have the same count, then mode can be meaningless. It did not include the range in the tabs above because it is not really a measure of central tendency.

2.3 SUMMARY OF LITERATURE SURVEY

Table-2.1Overview of Literature Survey

NAME OF THE AUTHOR	TITLE OF THE PAPER	PUBLICATION/YEAR	CONCEPT	ADVANTAGES	DISADVANTAGES
1. Chomtip Pornpanomchai, Fuangchar Stheitsthienchai and Sorawat Rattanachuen	Object detection and Counting System	IEEE/2008	Mainly focuses object detection and counting system using image processing. A video stream or a single image is given as the input and the system is designed to find which object is moving and count the number of moving objects in the scene.	The best level of difference ranges between 0.4 and 0.45. The maximum percentage of correctness reached by proposed system was 98%.	This system came with a certain disadvantage owing to which detection of objects above the height of 5m produces inaccurate results.

2.Yun ren, changren zhu, shunping xiao	Object Detection Based on Fast/Faster R-CNN Employing Fully Convolutional Architectures	Hindawi Publishing Corporation /2018	Mainly demonstrates how to employ the fully convolutional architectures in the fast/faster R-CNN.	Faster-R-CNN is 10 times faster than fast-R-CNN with similar accuracy of datasets like voc-2007. Faster-R-CNN has been one of the most accurate object detection algorithms.	In Faster R-CNN the ROI pooling layer is not as precisely mapped to the regions of the original image.
3.Shen Ding, Kun Zhao	Research on Daily Objects Detection Based on Deep Neural Network	IOP /2018	The main work includes collecting small dataset of daily objects, TensorFlow framework to build different models of object detection.	The well-trained model developed can be used in the mobile platform or in other intelligent devices to achieve accurate detection.	With increasing amount of detection data, the traditional detection method performance will become saturated. The detection performance will gradually improve, yet the improvement decreases after a certain

					amount of data.
4.Mausumi Maitra, Rahul Kumar Gupta and Manali Mukherjee	Detection and counting of Red Blood Cells in Blood Cell Images using Hough Transform	IEEE/2013	Proposed an approach to automatic segmentation and counting of Red Blood Cells in microscopic blood cell images using Hough Transform.	This system is cost-effective and time saving and also it can identify overlapping blood cells and count them separately.	This system does not count the effective number of RBC's which are partly in the image fields, thereby affecting the accuracy of the result.
5. Boris A. Alpatov, Pavel V. Babayan, Maksim D. Ershov	Vehicle Detection and Counting System for Real-Time Traffic Surveillanc e	IEEE/2018	Main work includes the development of vehicle detection and counting system with the help of smart cameras using vehicle detection and counting algorithm, road marking	This system has a high efficiency and the requirements to communication channels and a computing server are reduced.	This system is complex as it is a big system which includes manyunites in it like Vehicle Detectors, Intersection Controller, etc.

			detection algorithm.		
6. Carlos R. del-Blanco, Fernando Jaureguizar ,and Narciso Garcia	An Efficient Multiple Object Detection and Tracking Framework for Automatic Counting and Video Surveillance Applications	IEEE/2012	Proposed novel Bayesian tracking model that can manage multimodal distributions without explicitly computing the association between track objects and detections. It is based on off-the-shelf equipment, such as IP, web cameras, and PCs, and does not especial installation and configuration requirements.	Reduce the complexity of the fitting process, makingthe algorithm more suitable for fulfilled real-time restrictions.	It often comes with a high computational cost, especially in models with a large number of parameters.

7. J. Uma, P.Yuvarani	Detection of Shapes and Counting in Toy Manufacturing Industry with Help of Python.	International Conference on Electrical, Instrumentation and CommunicationEngineering / 2017	The main work focuses on detection of shapes of toy and its count in the particular area to segregate the items manufactured in the toy industry for packing. The identification of shapes using Ramer–Douglas-Peucker algorithm in Python language is a technique implemented with the help of open CV image processing tool.	The Open CV tool is the fastest compiler tool and its mostly available as open source tool.	The code was slow, especially for non-random access iterators, the code had become too complex.

8.Tobias Stahl, Silvia L. Pintea, and Jan C. van Gemert	Divide and Count: Generic Object Counting by Image Divisions.	IEEE/2019	Proposed a general object counting method that does not use anyprior category information.The method incorporates a counting layer which predicts object counts in the complete image, by enforcing consistency in counts when dealing with overlapping image regions.	The system developed with this approach proved to be simple, flexible and highly accurate	The method has a large variance on this dataset, which believed to be caused by the very limited number of training examples:in the range one hundred.

2.4 CONCLUSION

In the above papers, the authors have proposed various detecting techniques of objects and counting them. An accurate and efficient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. Custom dataset was created using labelingand the evaluation was consistent and can be used in real-time applications which requireobject detection for pre-processing in their pipeline. Various algorithms like Faster R-CNN, Ramer –Douglas-Peucker, Expectation-Maximization, and Levenberg-Marquardt are used to perform object detection. They have presented how to use the prevailing fully convolutional architectures in the notable object detection systems such as Fast/Faster RCNN. They have also derived a general formula for accurately designing the input size of the various fully convolutional networks in which the convolutional layer and pooling layer are concatenated with their strides being greater than 1 and have proposed an efficient architecture of skip connection to accelerate the training process. With this detailed information, a fast and effective approach can proposed to the problem of recognizing grocery products on store shelves.

CHAPTER 3

SYSTEM SPECIFICATION

The system requirements for our project “Retail Inventory Management and Stock-Out Predictions Using Real Time Object Detection” are explained below.The system requirements for any project or research make up the basis of that particularproject or research. It lays the foundation for the same. And in more specified way, isthe backbone of the research work being done. The system requirements are keyelements that give us a brief idea of how to proceed about with the work. They consistof the hardware and the software requirements. Together both of them make the systemrequirements.System requirements comprise of the basic elements and the key concept on how tostart on the specific topic and execute the same. The hardware requirements layemphasis on the tangible parts needed to execute the particular idea being sought.

3.1 HARDWARE REQUIREMENTS

1. *System*: To tell about the specifications of the system platform being used.
2. *Drive*: The memory space of the drive needs to be known as accordingly we can proceed with the same.
3. *GPU*: To tell about the graphics card being used.
4. *RAM*: RAM memory is being used.

All the above details come under the “Hardware Requirements” category.

It consists of the System of **Core i5 Processor 8th Generation**. It has frequency of 1.6 GHz and a Bus Speed of 4 GT/s OPI. Among the architectural details it as Data Width of 64 Bit. (information regarding Intel Core i5 Processor 8th Generation.)

The Drive used has *1 TB* memory space for accommodation purpose. And the RAM is of *8 GB*.

NVIDIA GeForce GTX 10 Seriesis the GPU used for our project. The GeForce 10 series is a series of graphics processing units developed by Nvidia, initially based on the Pascal microarchitecture announced in March 2014.

This design series succeeded the GeForce 900 series, and is succeeded by the GeForce 16 series and GeForce 20 series using Turing microarchitecture.On March 18, 2019 Nvidia announced that in a driver update due for April 2019 they would enable DirectX Raytracing on 10 series cards starting with the GTX 1060 6GB, and in the 16 series cards, a feature reserved to the Turing-based RTX series up to that point.

3.2 SOFTWARE REQUIREMENTS

The software requirements consist of the software part of the processing. They are the needed things which shape up the "software area" of the research work being pursued.A software requirement specification is a detailed description of the software to be developed and its functional and non-functional areas. It contains all necessary requirements needed for the project development. Foremost we need to have a proper understanding of all the technical areas. The software requirements will interact with all other modules, hardware, communication and all other programs areas.

The software requirements include the three basic areas:

1. *Operating System*: Tells about the Operating System being used.
2. *Coding Language*: The code used for the work.
3. *Toolkit:* Tells about the software platforms and tools for the implementation.

The software requirements consist of the following points:

The operating system is Windows 10. Improved application compatibility and shims compared to Windows 7. A number of new features in Windows 10 including your phone app, cloud clipboard, new screen capture utility, new search panel from start button, dark mode for file explorer, stop auto play in edge browser and more, swipe touch text entry with swift key, new game bar.

Python is the coding language that is used for work. Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until stepping down as leader in July 2018.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural. It also has a comprehensive standard library.

Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit Python Software Foundation.

The toolkitsthat are used for our projects are CUDA toolkit 9.0, cuDNN 7.0.5, Anaconda (Python), TensorFlow and Jupyter notebook. A brief description of all the toolkits are provided below.

CUDA is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

CUDA was developed with several design goals in mind:

- Provide a small set of extensions to standard programming languages, like C, that enable a straightforward implementation of parallel algorithms. With CUDA C/C++, programmers can focus on the task of parallelization of the algorithms rather than spending time on their implementation.
- Support heterogeneous computation where applications use both the CPU and GPU. Serial portions of applications are run on the CPU, and parallel portions are offloaded to the GPU. As such, CUDA can be incrementally applied to existing applications.

- CUDA-capable GPUs have hundreds of cores that can collectively run thousands of computing threads. These cores have shared resources including a register file and a shared memory. The on-chip shared memory allows parallel tasks running on these cores to share data without sending it over the system memory bus.

The **NVIDIA CUDA Deep Neural Network library (cuDNN)** is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers.

Deep learning researchers and framework developers worldwide rely on cuDNN for high-performance GPU acceleration. It allows them to focus on training neural networks and developing software applications rather than spending time on low-level GPU performance tuning. cuDNN accelerates widely used deep learning frameworks, including Caffe,Caffe2, Chainer, Keras,MATLAB, MxNet, TensorFlow, and PyTorch.

Deep learning frameworks using cuDNN 7.5 and later, can leverage new features and performance of the Volta and Turing architectures to deliver faster training performance. cuDNN 7.5 highlights include:

- Up to 3x faster training of ResNet-50 and GNMT on Tesla V100 vs. Tesla P100
- Improved depth-wise separable convolution for training models such as Xception and Mobilenet
- Multi-Head Attention for accelerating popular models such as Transformer
- New tensor folding APIs for accelerated performance on models such as Mask R-CNN, GANs and DeepSpeech2

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package

management system conda. The Anaconda distribution is used by over 12 million users and includes more than 1400 popular data-science packages suitable for Windows, Linux, and MacOS.

Anaconda distribution comes with more than 1,400 packages as well as the Conda package and virtual environment manager, called Anaconda Navigator, so it eliminates the need to learn to install each library independently.

The open source packages can be individually installed from the Anaconda repository with the conda install command or using the pip install command that is installed with Anaconda. Pip packages provide many of the features of conda packages and in most cases they can work together.

Custom packages can be made using the conda build command, and can be shared with others by uploading them to Anaconda Cloud, PyPI or other repositories.The default installation of Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7.

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

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.

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for as many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release (December 2011), renamed to Jupyter notebook in 2015 (IPython 4.0 – Jupyter 1.0). Jupyter

Notebook is similar to the notebook interface of other programs such as Maple, Mathematica, and SageMath, a computational interface style that originated with Mathematica in the 1980s. According to The Atlantic, Jupyter interest overtook the popularity of the Mathematica notebook interface in early 2018.

HARDWARE REQUIREMENTS

Table 3.1 Hardware Requirements

System	Intel Core i5 Processor 8th Generation
Drive	1 TB
GPU	NVIDIA GeForce GTX 10 Series
Ram	8 GB

SOFTWARE REQUIREMENTS

Table 3.2 Software Requirements

Operating system	Windows 10
Coding Language	Python
Toolkits	CUDA toolkit 9.0, cuDNN 7.0.5, Anaconda (Python), TensorFlow, Jupyter notebook

CHAPTER 4

SYSTEM DESIGN

4.1 PROPOSED WORK

Faster R-CNN is an extension of the R-CNN and Fast R-CNN object detection techniques. All three of these techniques use convolutional neural networks (CNN). The difference between them is how they select regions to process and how those regions are classified. R-CNN and Fast R-CNN use a region proposal algorithm as a pre-processing step before running the CNN. The proposal algorithms are typically techniques such as EdgeBoxes or Selective Search, which are independent of the CNN. In the case of Fast R-CNN, the use of these techniques becomes the processing bottleneck compared to running the CNN. Faster R-CNN addresses this issue by implementing the region proposal mechanism using the CNN and thereby making region proposal a part of the CNN training and prediction steps. In this project, an object detector is trained using the faster_rcnn_inception_v2_coco_2018_01_28 model.

Figure-4.1 Detecting Stock-Out

4.1.1 OBJECT DETECTION

Our project primarily utilizes the concepts of object detection and counting. Object detection is a technique related to computer vision and image processing which deals with the detection of semantic objects of a certain type or class in images and surveillance videos. Object detection methodologies can be classified as machine learning-based approaches or deep learning-based approaches. For Machine Learning approaches, defining the features first and using a method like support vector machine (SVM) to do the classification is mandatory. On the other hand, deep learning techniques are those that are able to do end-to-end object detection without specifically defining features, and are typically based on convolutional neural networks (CNN). Typically, there are three steps involved in every object detection framework.

1. The regions of interest or regions where the model proposes the existence of an object are first generated. These region proposals are given as a set of bounding boxes spanning the entire image.
2. The second step consists of the extraction of features for each bounding box and their evaluation to determine which object is present in the proposed region based on visual features.
3. In the final post-processing step, overlapping boxes are combined into a single bounding box (that is, non-maximum suppression).

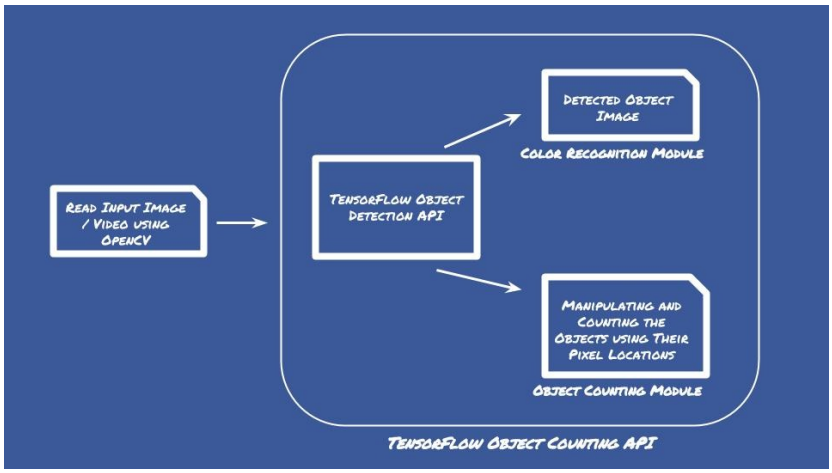


Figure-4.2 TensorFlow Object Counting API Architecture

4.1.2 FASTER R-CNN

Faster R-CNN is an advanced form of the R-CNN and Fast R-CNN techniques of object detection. Convolutional neural networks (CNN) is used in all of the above techniques. The difference between them is how the regions to be processed are selected and how the classification is done. Before running the CNN R-CNN and Fast R-CNN use a region proposal algorithm. The proposal algorithms used methods like EdgeBoxes and Selective Search, which are not dependent on the CNN. In the case of Fast R-CNN, the use of these techniques becomes the processing bottleneck compared to running the CNN. This issue is addressed in Faster R-CNN with the usage of CNN for implementing the region proposal mechanism, which makes region proposal a part of the CNN training and prediction steps. The model used for training the object detector in this project is the faster_rcnn_inception_v2_coco_2018_01_28 model. Faster R-CNN was developed by researchers at Microsoft. It is based on R-CNN which used a multi-phased approach to object detection. R-CNN used Selective search to determine region proposals, pushed these through a classification network and then used an SVM to classify the different regions.

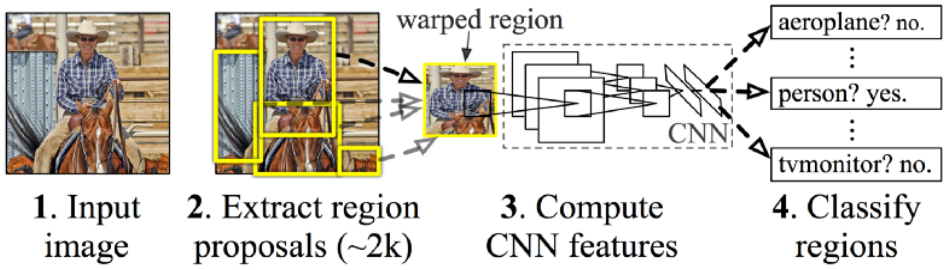
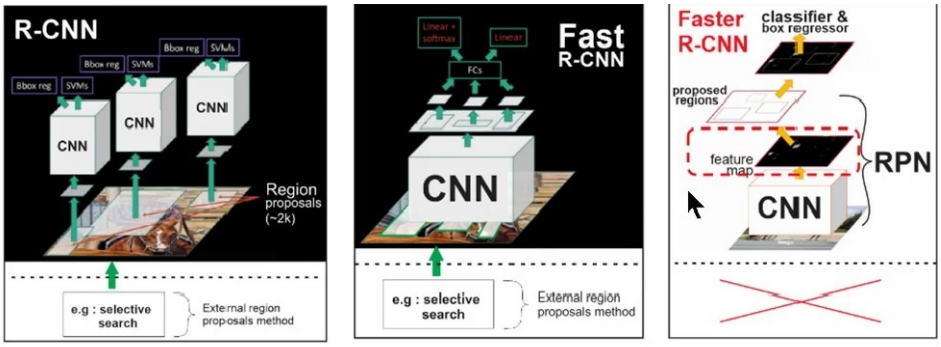


Figure-4.3 Overview of R-CNN Architecture

Faster R-CNN like to SSD, is an end-to-end approach. Instead of using default bounding boxes, Faster R-CNN has a Region Proposal Network (RPN) to generate a fixed set of regions. The RPN enables nearly cost-free region proposals by using the convolutional features from the classification network. The RPN is implemented as a

fully convolutional network which generates object bounding boxes and predicts objectless scores at every position.

RPN has a similar setup as the SSD network. A set of proposals are computed with various scales and aspect ratios, at each sliding-window location or anchor. The outcome of the RPN are adjusted bounding boxes which are based on the anchors.



	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	50 seconds	2 seconds	0.2 seconds
Speed-up	1x	25x	250x
mAP (VOC 2007)	66.0%	66.9%	66.9%

Figure-4.4Comparisons of R-CNN

4.1.3. REGION PROPOSALS

Several different approaches exist to generate region proposals. Selective search was the algorithm originally used to generate object proposals. It's a clustering-based approach that groups pixels and based on the generated clusters it generates proposals.

There are few other approaches that make use of more complex visual features extracted to generate region proposals or adopt a brute-force approach to region generation. These regions are automatically generated, without considering the image features.

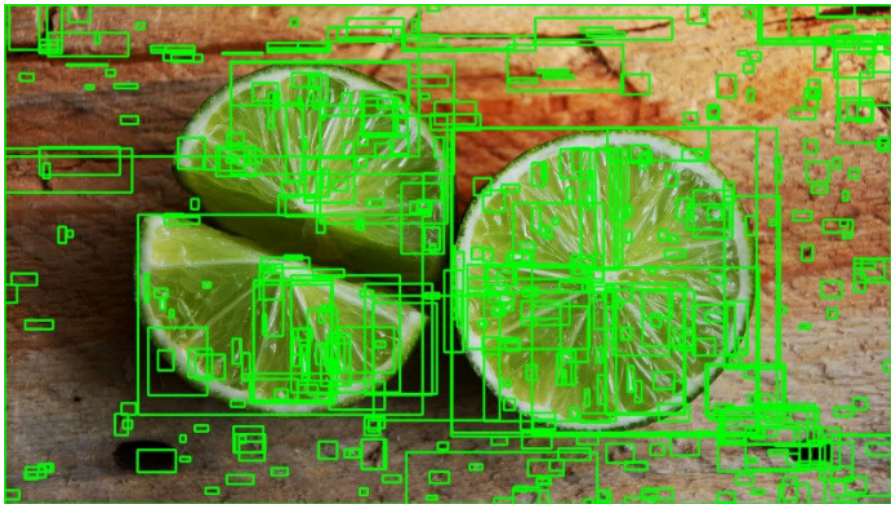


Figure-4.5 Selective Search Applied to an Image

The number of regions vs. the computational complexity is an important trade-off that is made with generation of region proposals. The more the number of regions generated, the more likely you are to find the object. On the other hand, if you exhaustively generate all proposals, it is possible to run the object detector in real-time.

4.1.4. METHODOLOGY

1. Gathering Pictures

The training images have unwanted objects in the image along with the desired products, and a variety of backgrounds and lighting conditions to train a robust classifier. There are images with products that are overlapping with other products or that are partially obscured. Such flawed images facilitate proper training of the object detector thereby resulting in an efficient software.

2. Labeling Pictures

We use the LabelImg tool for labeling our images. Once each image has been saved and labeled, there is one .xml file for each image in the \test and \train directories. Also, sizeChecker.py was run in order to check if the bounding boxes were of the appropriate size is checked if it is correct by running.

3. Generating Training Data

The TFRecords which are the input data to TensorFlow training model were generated with the help of the labelled images. The xml_to_csv.py and generate_tfrecord.py scripts from Dat Tran's Raccoon Detector dataset are used with some slight modifications to work with the directory structure.

4. Creating Label Map and Configuring Training

Before training a label, map was created and then the training configuration file was edited. From the label map the trainer learns about the object by defining a mapping of class names to class ID numbers. Finally, the object detection training pipeline that defines the model and parameters used in training.

5. Running the Object Counting Model

The trained object counting software is now run and the number of products of each kind are accurately counted. In every five minutes checkpoints are saved by the training routine.

6. Inventory Management and Stock-out Prediction

The products count that are obtained from the object counting model are entered into a spreadsheet where inventory management is performed. These counts as observed before the opening and after the closing of the retail outlet are used for conducting inventory while the product counts taken continuously by the surveillance camera are used to predict stock-out conditions from time to time.

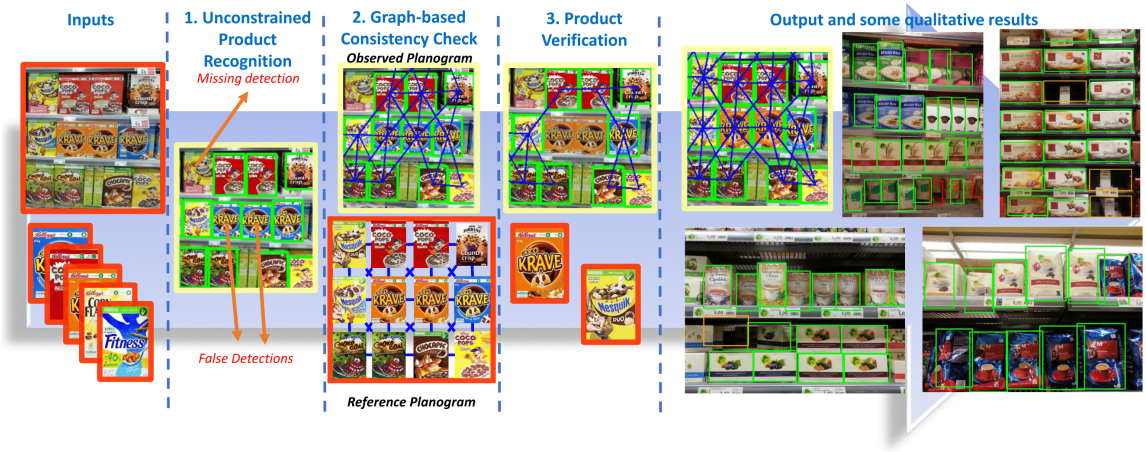


Figure-4.6Object and Stock-Out Detection

4.2 SYSTEM ARCHITECTURE

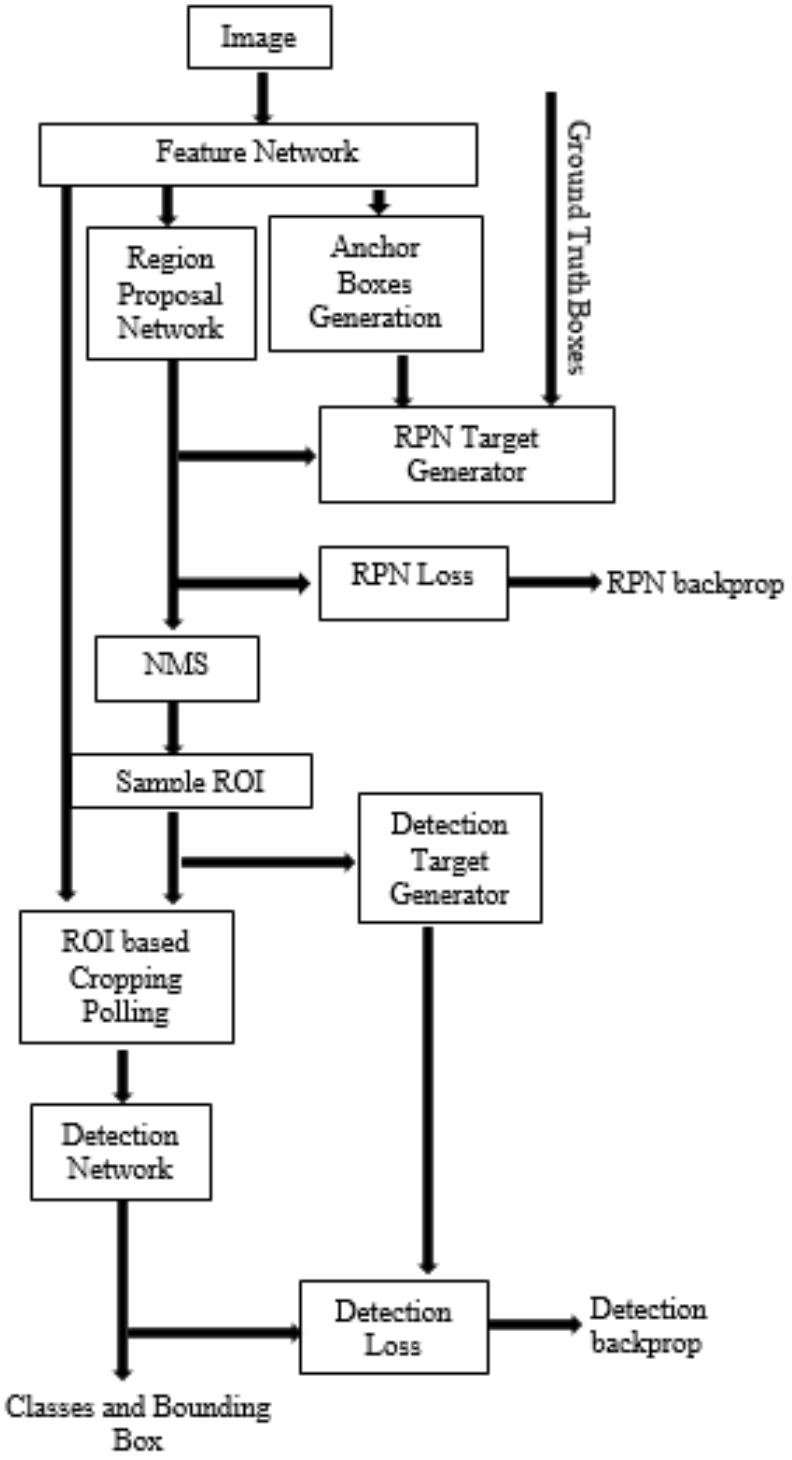


Figure-4.7Object Detection Architecture

- The image from which the objects need to be detected is given as an input to the feature network.
- The feature network rescales the image given as input and detects the features of objects present in the image.
- The image is then sent into the Region Proposal Network which is the most essential part of the object detection system.
- To generate the proposals for the region where the object lies, a small network is slide over a convolutional feature map that is the output by the last convolutional layer. RPN has a classifier and a regressor. Classifier determines the probability of a proposal having the target object. Regression regresses the coordinates of the proposals.
- Anchor boxes generator generates multiple boxes in the probable regions where there is a high expectancy of the presence of an object.
- After going through the RPN and anchor boxes generation network the image now reaches the NMS expanded as Non-Maximum Supression. NMS is used to make sure that in object detection, a particular object is identified only once. It first discards all those cells where probability of object being present (calculated in final softmax layer) is <= 0.6. Then it takes the cell with largest probability among candidates for object as a prediction. Finally we discard any remaining cell with Intersection over union value >= 0.5 with the prediction cell.
- The attention of the neural network on a particular region in the image is known as the Region of Interest (RoI) it is inserted inside the net as a binary map. For every region of interest from the input list, it takes a section of the input feature map that corresponds to it and scales it to some pre-defined size.
- After clear identification of the regions of interest the image goes into the detection network where the final detection of the object and classification into pre defined classes is performed.

4.3 RESULTS

We have reported the comparison of different methods of object detection. We train the Faster R-CNN object detection model in a store. The objects are detected with approximate detection values.\

Table-4.1 Observation of Sample Images with various approaches

Sample Images	Actual Count	CNN		R-CNN		Fast R-CNN		Faster R-CNN	
		Estimated Count	Error in Count	Estimated Count	Error in Count	Estimated Count	Error in Count	Estimated Count	Error in Count
1	3	1	2	2	2	3	1	3	3
2	3	3	2	1	1	2	2	3	3
3	4	2	1	3	3	3	0	4	4
4	3	0	3	1	0	1	3	3	3

The values in the table determine the accuracy of the object being detected. This test is done using other object detection approaches like CNN, R-CNN, and fast R-CNN to compare their results with our proposed faster R-CNN object detection model. The test has been performed for different classes of images for better comparison. As a result we have found out that faster R-CNN has matched with the actual values and makes it better than other object detection models.

Figure-4.8 Object Detection in store

We have represented the results in graph to understand the results of our proposed object detection model.

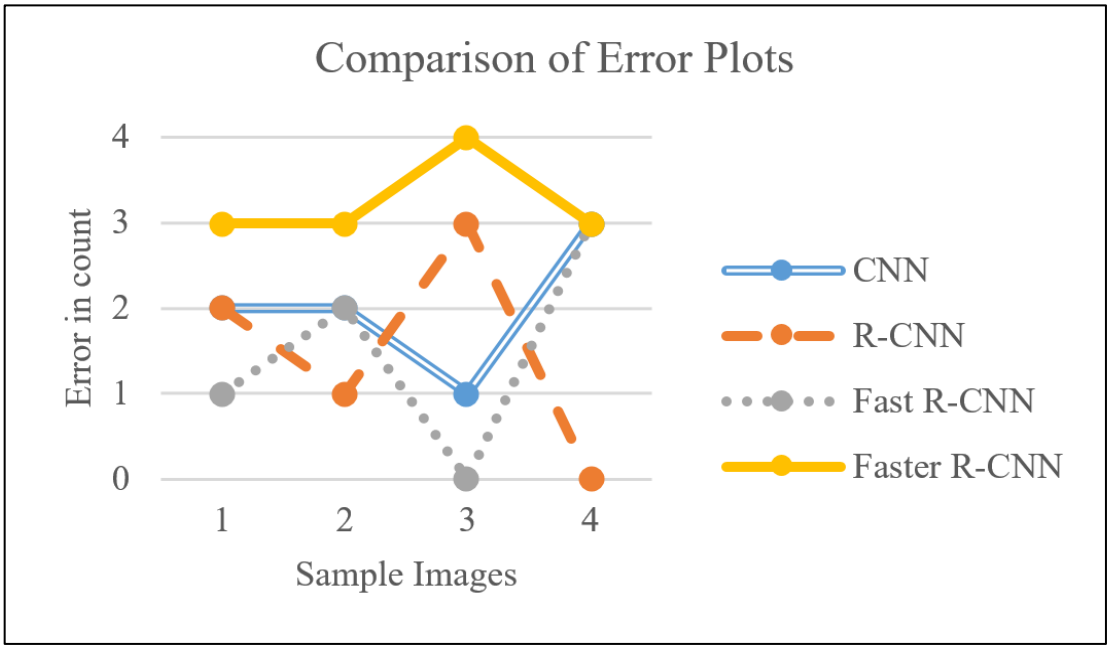


Figure-4.9 Comparison of Error Plots for various approaches

4.4 OUTPUT

- In the below images we observe the output generated by the object detection system.
- The soft drink cans are detected and classified depending on their types and a count is taken of number of objects in each of the classes.
- In figure 4.11 one object is removed off the shelf and the object count for that class is observed to decrease.

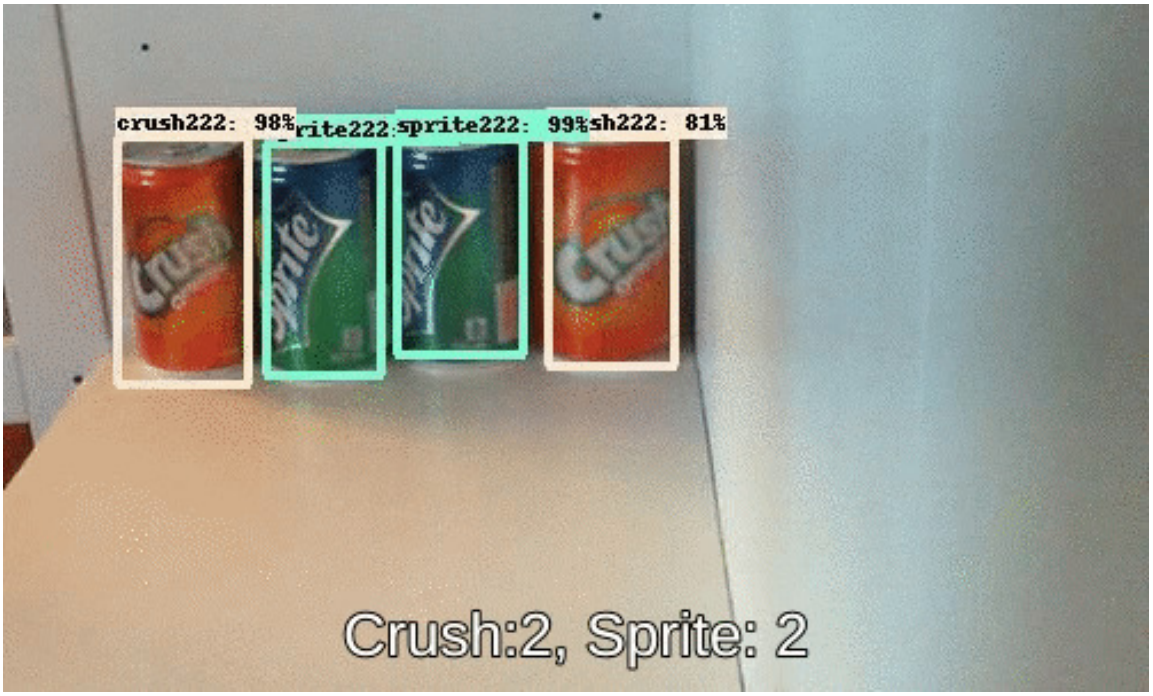


Figure-4.10 Output 1

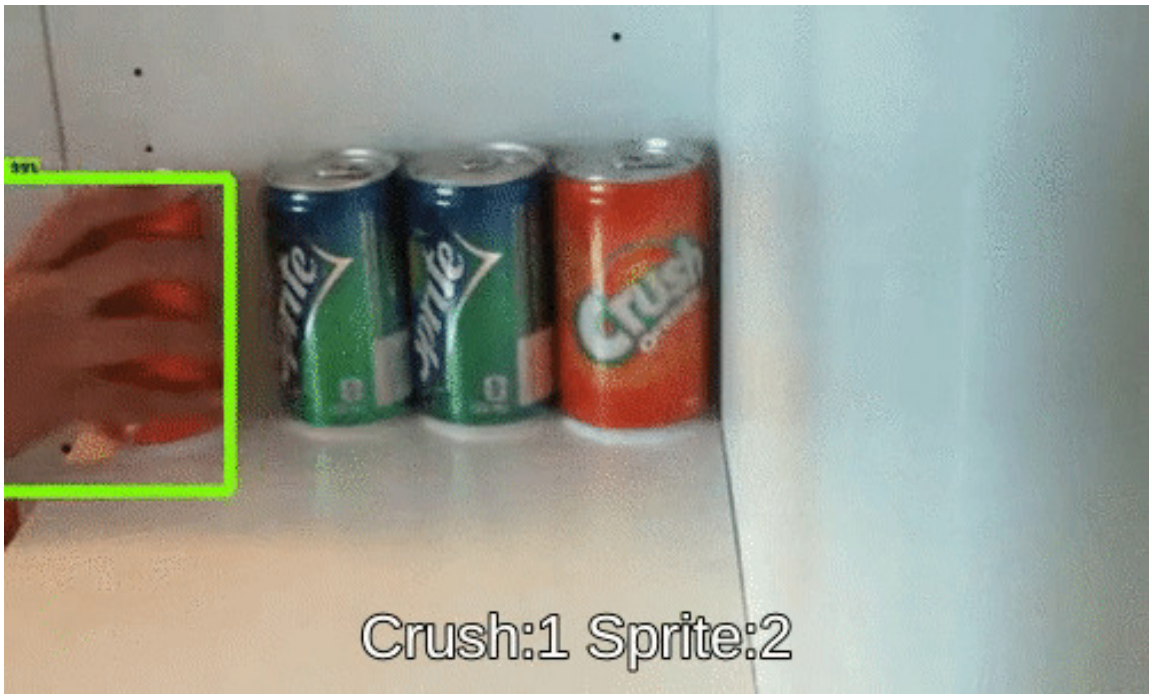


Figure-4.11 Output 2

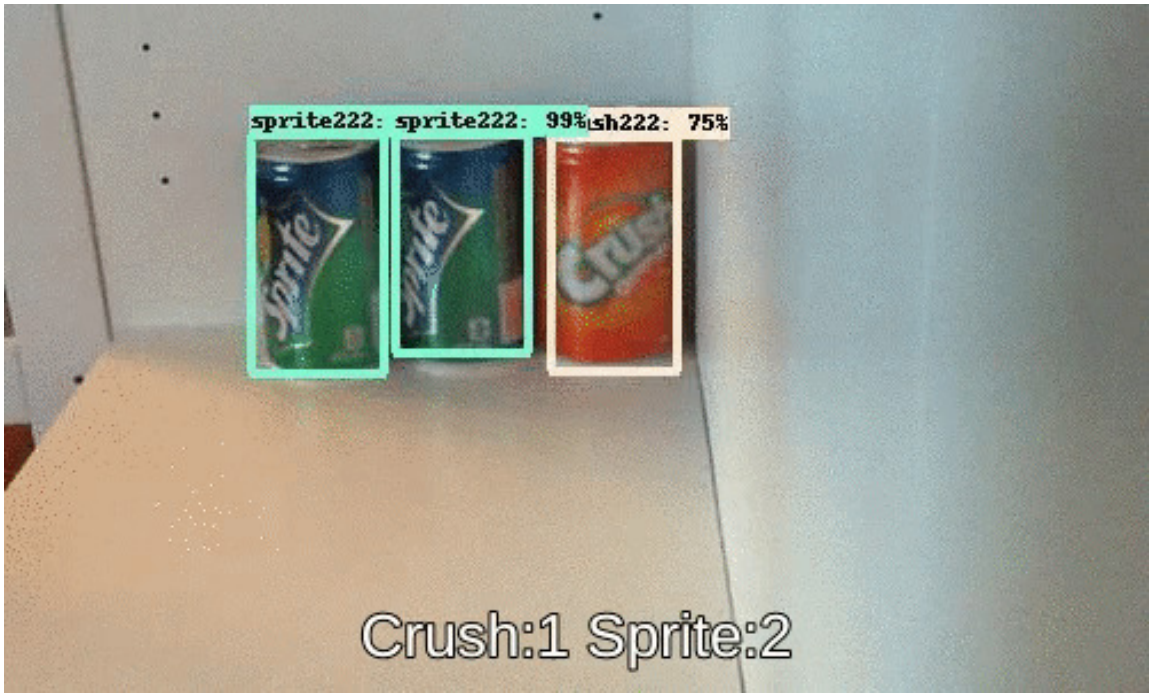


Figure-4.12 Output 3

Figure-4.13 Output 4

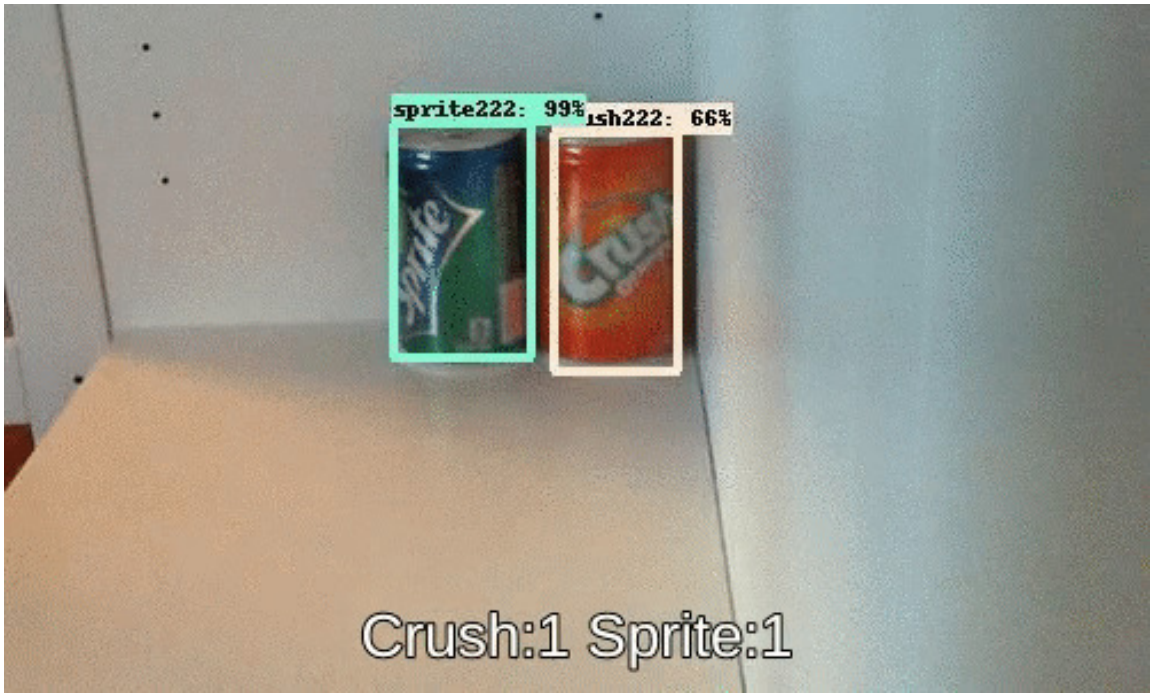


Figure-4.14 Output 5

CHAPTER 5

CONCLUSION

This object detection system is one of the most efficient and effective ones so far as it uses Faster R-CNN. The inventory management is a clean and error free operation when done using this software. Images were analysed and multiple stock out features within the image were detected. If such features are detected an indication is provided about the stock out condition. As a result, the present methodology automates multiple tasks which were previously implemented manually, thereby enhancing efficiency and effectiveness of inventory management operations to a great extent.

APPENDIX

Codes

```
import os
import cv2
import numpy as np
import tensorflow as tf
import sys

sys.path.append("..")

from utils import label_map_util
from utils import visualization_utils as vis_util

MODEL_NAME = 'inference_graph'
IMAGE_NAME = 'test1.jpg'

CWD_PATH = os.getcwd()

PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,'frozen_inference_graph.pb')

PATH_TO_LABELS = os.path.join(CWD_PATH,'training','labelmap.pbtxt')

PATH_TO_IMAGE = os.path.join(CWD_PATH,IMAGE_NAME)

NUM_CLASSES = 6

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASSES, use_display_name=True)
```

```
category_index = label_map_util.create_category_index(categories)

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

    sess = tf.Session(graph=detection_graph)

image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')

detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

num_detections = detection_graph.get_tensor_by_name('num_detections:0')

image = cv2.imread(PATH_TO_IMAGE)
image_expanded = np.expand_dims(image, axis=0)

(boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes, num_detections],
    feed_dict={image_tensor: image_expanded})

vis_util.visualize_boxes_and_labels_on_image_array(
    image,
    np.squeeze(boxes),
```

```
    np.squeeze(classes).astype(np.int32),
    np.squeeze(scores),
    category_index,
    use_normalized_coordinates=True,
    line_thickness=8,
    min_score_thresh=0.60)

cv2.imshow('Object detector', image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

REFERENCES

[1] Ren, Yun & Zhu, Changren & Xiao, Shunping. (2018) “Object Detection Based on Fast/Faster RCNN Employing Fully Convolutional Architectures”,Mathematical Problems in Engineering. 2018, pp 1-7.

[2] Pornpanomchai, Chomtip & Stheitsthienchai, Fuangchat & Rattanachuen, Sorawat. (2008) “Object Detection and Counting System”,IEEE, pp. 61 - 65.

[3] Maitra, Mausumi & Kumar Gupta, Rahul & Mukherjee, Manali. (2012), “Detection and Counting of Red Blood Cells in Blood Cell Images using Hough Transform”,International Journal of Computer Applications, pp 13-17.

[4] A. Alpatov, Boris & Babayan, Pavel & Ershov, Maksim (2018),”Vehicle detection and counting system for real-time traffic surveillance”, pp 1-4.

[5] Del-Blanco, Carlos & Jaureguizar, Fernando & Garcia, Narciso (2012), “An Efficient Multiple Object Detection and Tracking Framework for Automatic Counting and Video Surveillance Applications” Consumer Electronics, IEEE Transactions on. 58. 10.1109/TCE.2012.6311328.

[6] Stahl, Tobias & Pintea, Silvia & Gemert, Jan. (2018), “Divide and Count: Generic Object Counting by Image Divisions”, IEEE Transactions on Image Processing, pp. 1-1.

[7] https://youtu.be/Rgpfk6eYxJA,”How to Train an Object Detection Classifier Using TensorFlow 1.5 (GPU) on Windows 10”.

[8] https://patents.google.com/patent/US20090063307A1/en“Detection Of Stock Out Conditions Based On Image Processing”.