SAKARYA UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

A NEW DEEP LEARNING BASED OBJECT DETECTION SYSTEM FOR INCREASING SALESMAN PERFORMANCE

M.Sc. THESIS

Ahmad KUBAJI

Department	**:**	**COMPUTER AND INFORMATION ENGINEERING**
Supervisor	**:**	**Prof. Dr. Ahmet ZENGIN**

August 2022

SAKARYA UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY

A NEW DEEP LEARNING BASED OBJECT DETECTION SYSTEM FOR INCREASING SALESMAN PERFORMANCE

M.Sc. THESIS

Ahmad KUBAJI

Department	**:**	**COMPUTER AND INFORMATION ENGINEERING**
Field of Science	**:**	**COMPUTER ENGINEERING**
Supervisor	**:**	**Prof. Dr. Ahmet ZENGIN**

This thesis has been accepted unanimously by the examination committee on 04.08.2022

DECLERATION

I declare that all the data in this thesis was obtained by myself in academic rules, all visual and written information and results were presented in accordance with academic and ethical rules, there is no distortion in the presented data, in case of utilizing other people's works they were refereed properly to scientific norms, the data presented in this thesis has not been used in any other thesis in this university or in any other university.

Ahmad KUBAJI

~~16.02.2015~~

ACKNOWLEDGMENTS

In the name of ALLAH, the Most Gracious, The Most Merciful, Praise be to ALLAH, who enabled us to do this work and facilitated the path of knowledge for us, I extend my thanks and gratitude to everyone who helped me to complete this research, my thanks and appreciation for the big scientific edifice Sakarya University, my appreciation and gratitude for the incredible effort and support provided to me by my advisor Prof. Dr. Ahmet Zengin, to my dear mother, to my dear father, to my wife my partner in life, and to all my family members, I dedicate this scientific research to you.

TABLE OF CONTENTS

LIST OF SYMBOLS AND ABBREVIATIONS

AI	: Artificial Intelligence
ANN	: Artificial Neural Network
AP	: Average Precision
API	: Application Programming Interface
ADAS	: Advanced Driver Assistance Systems
BI	: Business Intelligence
CNN	: Convolutional Neural Network
COCO	: Common Object in Context
Colab	: Collaboratory
CPU	: Central Processing Unit
CUDA	: Compute Unified Device Architecture
FMCG	: Fast-Moving Consumer Goods
GPU	: Graphical Processing Unit
HR	: Human Resource
IoU	: Intersection Over Union
mAP	: Mean Average Precision
ML	: Machine Learning
OD	: Object Detection
ODS	: Object Detection System
PDE	: Partial Differential Equations
RAM	: Random Access Memory
RBM-CNN	: Restricted Boltzmann Machine - Convolutional Neural Network
ReLU	: Rectified Linear Unit
RNN	: Recurrent Neural Network
ROI	: Region of Interest

RPN	: Region Proposal Network
SOTA	: State of The Art
SSD	: Single-Shot Detector
SVM	: Support Vector Machine
TF	: TensorFlow
VM	: Virtual Machine
YOLO	: You Only Look Once

LIST OF FIGURES

LIST OF TABLES

SUMMARY

Keywords: Deep Learning, YOLO Algorithm, Cost Reduction, Food Distribution

Food distribution companies must be flexible and responsive to provide the best services to their customers. Operational costs and expenses are the biggest concern for these companies, as they rely on many employees to serve customers and achieve the company's goals. Increasing tasks creates continuous and growing pressure on the sales team and the rest of its supporting departments, which affects the time and effort of the salesperson in achieving daily tasks and thus achieving the expected monthly goal.

Deep learning technologies provide solutions to help reduce the effort expended by the sales team and, at the same time, contribute to reducing operating costs. In this thesis, an integrated system is proposed to use deep learning techniques to reduce the representative's effort and compensate the human support staff with deep learning algorithms to provide the necessary support to the representative.

YOLO algorithms are used to discover objects in real-time and Faster-RCNN algorithm is used to solve planogram issues on shelf. The results extracted from images are used from these algorithms and converted into data stored in the MySQL database. Google API techniques are used to transfer the results from the cloud to the company's servers and thus analyze the results received using the Microsoft Power BI tool. The results show that proposed system provides a complete solution to reduce the effort of the sales team and significantly reduce costs. Also, this system can be used to study the competitor productions visibility on shelves.

SATIŞ ELEMANI VERİMLİLİĞİ İÇİN YENİ BİR DERİN ÖĞRENME TABANLI NESNE TESPİT SİSTEMİ

ÖZET

Anahtar Kelimeler: Derin Öğrenme, YOLO, Maliyet Azaltma, Gıda Dağıtımı

Gıda dağıtım şirketleri, müşterilerine en iyi hizmeti sunmak için esnek ve duyarlı olmalıdır. Müşterilere hizmet vermek ve şirketin hedeflerine ulaşmak için çok sayıda çalışana ihtiyaç duyduklarından, bu şirketler için en büyük endişe operasyonel maliyetler ve harcamalardır. Artan görevler, satış ekibi ve diğer destek departmanları üzerinde sürekli ve artan bir baskı yaratır, bu da satış görevlisinin günlük görevleri yerine getirme ve dolayısıyla beklenen aylık hedefe ulaşma zamanını ve çabasını etkiler.

Derin öğrenme teknolojileri, satış ekibinin harcadığı çabayı azaltmaya yardımcı olacak çözümler sunar ve aynı zamanda işletme maliyetlerinin düşürülmesine de katkıda bulunur. Bu tezde, çalışanların iş yükünü azaltmak ve gerekli desteği sağlamak maksadıyla derin öğrenme tekniklerinin ve algoritmalarının kullanıldığı tümleşik bir sistem önerilmiştir.

Bir markette nesneleri ve ürünleri gerçek zamanlı olarak tespit etmek için YOLO algoritması kullanılmıştır. Daha sonra görüntülerden elde edilen sonuçlar bu algoritma kullanılarak işlenmiş ve Google API tekniklerinden yararlanılarak MySQL veritabanında depolanmıştır. Faster-RCNN algoritması ile sonuçlar buluttan şirketin sunucularına aktarılmış ve böylece Microsoft Power BI aracı ile alınan sonuçlar analiz edilmiştir. Sonuçlar, önerilen sistemin satış ekibinin iş yükünü azaltmak ve maliyetleri önemli ölçüde düşürmek için eksiksiz bir çözüm sağladığını göstermektedir. Ayrıca, bu platform, gelecekte rakip üretimlerin raflardaki görünürlüğünü incelemek için kullanılabilir.

CHAPTER 1. INTRODUCTION

The economy is currently witnessing a highly competitive ability between companies distributing food and commodity products, which requires a highly competitive effort that makes these companies flexible and quick to respond at the same time to provide the best service to their customers. Costs and operating expenses are some of the biggest challenges facing these companies. Reducing these expenses is the focus of all specialists in the research to find the best ways to gain time in customer service and thus reach the product to the customer's hand as soon possible. The reduction of costs and expenses should not affect the performance of the work team, which is the main structure of these companies. Accordingly, all tools and means that contribute to the success of this goal and return to the benefit of all parties must be provided [1][2].

When adding new tasks that contribute to customer service, the most significant pressure is placed on the work team, whether the sales team or the cadres that work in the office to support this team. Given the increase in tasks due to the competitive strength between companies and the change that occurs on a daily basis in global markets, the sales team will face a major challenge in achieving the annual sales plan. Thus, the sales representative's performance will be affected by the time completed in implementing the daily goal to visit customers and conduct successful sales. Consequently, it is reflected in the monthly evaluation of the representative's performance and thus increase the number of incomplete deals to increase the tasks for the representative in the same period.

Reducing the time for the tasks performed by the sales representative is one of the significant challenges facing the sales management in the Fast-Moving Consumer Goods (FMCG) companies, where the high competitiveness of many companies that have similar products in quality and diversified items. The product should be in the right place at the right time. Providing the necessary tools to sales representatives and the work team is the task of the company's management to maintain the performance of the representative's work according to the plan agreed upon between the sales department and the company's management.

Deep learning techniques have become the focus of specialists' search to address the problems of costs and expenses. It may help to improve the efficiency of sales representatives' work. There are many studies in which deep learning techniques have contributed to reducing costs in general, including raising the efficiency of the work team, only studies [1][2],

The use of object detection techniques using Faster-RCNN algorithms on salesmen's devices in grocery store environments contributed to raising the productivity of the salesmen's work and thus reducing the mistakes as a result of work pressure [1]. They use artificial intelligence to perform over their counterparts who rely on traditional methods [2]. The application of artificial intelligence has enabled companies to gain increased profits by enhancing consumer good industries and their overall operating activities, which will create a path toward organizational profitability and sustainability [3]. Artificial intelligence also contributes to reducing costs in many organizations, including health institutions. For example for past years, efforts have been mainly exerted to use these techniques to reduce costs in the health fields for the purpose of lowering costs and improving the efficiency of workers [4][5][6].

1.1. Literature Review

In [1], a system was developed to eliminate mistakes and boost efficiency by assisting salesmen to be more productive. The system uses Faster- RCNN algorithms as a deep learning object detection system to classify the products in the markets. System was tested in a variety of conditions in markets environments, and it was achieved an accuracy of 99%.

In [2], a study that improves that the FMCG industries have good results to reduce costs and increasing the effects of the sales team performance by using AI. By employing AI programming and algorithms, the FMCG industry may better understand and predict future business prospects by looking at operational data that has been acquired over the years. As a means of improving consumer goods' supply and distribution, inventory management should take into account anticipating sales and wasteful expenditures as well as losses. To boost operational revenue, one should replace product inventory with shortages in retail inventories so that customers' needs are met.

In [3], a study to improve that the ML and AI becomes crucial factor of success in modern sales organizations. According to the study, sales representatives that are using AI give more accurate results than the other sales representatives. At the same time, it emphasizes that human services cannot be dispensed within this field. Still, this field helps to obtain better services for customers and improve the way of communication between the customer and sales representatives.

In [7], an intelligent HRS system has been proposed to reduce errors and increase performance for the healthcare team. The system depends RBM-CNN deep learning method to present an opportunity for the health care industry, which shows fewer errors compared to other deep learning approaches. The system has been evaluated by reviewing the MAE and RSME results. In [4], MLPA solutions for healthcare efficiency that meet inclusion and exclusion criteria were found through systematic searches of relevant business news and academic research databases. The purpose of the study was to identify five predictions produced by MLPA products out of the 106 items discovered using the information available from the two product marketing businesses through the use of content analysis. According to the findings, the information gathered can be used as a reference point to enhance the analysis. The use of MLPA to improve health care and lower costs without compromising quality which establishes a reference to enrich the analysis of ethical and regulations. In [5], using a second-generation AI system, the digital pill system has been proposed. The digital pill is able to overcome the partial or complete loss of response and adherence caused by long-term drugs. With this system, the patient will receive an easy-to-follow therapy regimen via their cell phone. The digital pill lowers healthcare expenditures by boosting the reaction to currently prescribed drugs. According to studies, the use of a digital pill can lower healthcare expenses. Four decades, a total loss of USD 94 trillion was predicted for all NCDs.

In [8], a proposal was made to use downscaled hardware to detect vehicles and people in real-time to reduce accidents. The model was designed based on the YOLOv3 deep learning algorithm. These technologies reduce the high costs of driving devices and technologies, which are considered expensive and whose prices are increasing day by day. Reducing cost is an essential factor in enhancing ADAS systems to avoid accidents. Combining YOLOv3 technologies with NVIDIA series real-time object detection for use in ADAS systems, the model accuracy was found to be 96% for real-time object detection. The work lays a solid foundation for conducting transport analysis research based on ADAS and deep learning. For intelligent transportation systems such as real-time congestion estimation and accident detection.

In [9], a method is built consisting of an automatic encoder for long-term memory and a mixture of advanced classifiers. By extracting information from electric power market data that complain of price equilibrium, the algorithms proposed by artificial intelligence alter production forecasts and prevent sudden cost differences from causing imbalances. The results of the strategy work well, and its effects can be relied upon, which reach 6.258% to 11.195% as a reduction in the budget cost of several wind power plants that have been tested.

In [10], a comparative study was conducted between machine learning techniques to take advantage of the results in reducing advertising costs and reducing expenses. Companies suffered from a lack of human resources can resort to artificial intelligence methods to make decisions on appropriate direction of expenditures and costs. Therefore, the researchers compared more than one artificial intelligence algorithm such as ANN algorithms, LSTM, ML, Random Forest Regression (RFR) and Support Vector Regression (SVR) and Decision Tree Regressor (DTR). The results concluded that increasing the tendency of companies to increase advertisements will contribute to measuring the return on investment in commercial advertisements, and thus reduce costs by not leading to useless advertisements.

In [11], using ML algorithms will result in accurate demand forecasting. Businesses may avoid stocking too much or too little of a particular item and take advantage of sales possibilities, robust replenishment systems and new product planning, and reduction of costs, among other benefits.

Deep Learning has become an important feature in business field to perform best analysis for make decisions. Many of the platforms mentioned before, supports business field or on-line deep learning methods. For example, YOLO algorithm has the ability to help in business field that the reason to use it in our thesis to solve a real business case in FMCG filed because it has the best feature to detect the objects in real-time.

1.2. Motivation

The main concern of Fast-Moving Consumer Goods (FMCG) companies has always been to search for the best ways to reduce operating costs for such projects. Through the experience for many years in managing FMCG companies, most companies working in this field focus most of the time on the factor of cost reduction without going too deep. In the impact of this matter on the performance of the work team, it is known that FMCG companies focus all their efforts in supporting the sales team, which is the basic building block for achieving revenue for such companies. If any new task or plan is assigned to the sales team, it must be supported by the best means for the purpose of maintaining quality work and at the same time achieving the desired goal, which is to reduce operational costs.

Therefore, it was necessary to find auxiliary solutions that contribute accurately to avoiding making such mistakes. It was essential to keep pace with the development in the field of artificial intelligence, especially the field of deep learning, which contributes to providing remarkable support to the human element in solving complex problems in the areas of life. It was noted that there is a lack of literature studies that employ deep learning methods to contribute to Fast-Moving Consumer Goods (FMCG) activities. It contributes directly or indirectly to improving the efficiency of the work team and at the same time working to reduce costs in many areas. Object detection techniques are played a significant role in identifying things and helping the human element to make accurate decisions, including algorithms YOLO V3 and Faster - RCNN, which can be of great use in supporting the work of the sales team in FMCG companies.

1.3. Objective

This thesis aims to engage deep learning techniques in object detection by using YOLOv3 and Faster-RCNN algorithms techniques to reduce operational costs and improve the performance of the work of sales representatives. All scenarios are implemented using Google Colab [46], Anaconda Spyder [47], and Google Drive environments [48]. A comparison was made of the performance of sales representatives before and after using deep learning algorithms and a comparison of operating costs before and after using these techniques.

1.4. Methodology

There are many studies about object detection. Still, all of these studies did not focus directly on the problems of FMCG companies to study the problem of inflation in financial spending and consider the increase in the efficiency of the affiliates. On this data YOLOv3 and Faster-RCNN, manual data labelling techniques were used using different tools such as lblImage and RoboFlow Framework. After completing the necessary data collection and initialization of the required dataset, the stages of deep learning model modifications corresponding to the data set were implemented, and then the results were evaluated based on the metrics of the object detection system with the type of platforms used to transform the work into a fully systematic work.

1.5. Contribution

This study contributed in several aspects that can be summarized in several points:

- Introducing the field of deep learning and object detection such as YOLOv3 and Faster-RCNN algorithms in the work system of commercial sector companies, especially FMCG companies.

- Develop an integrated system to convert the detected objects in the images into readable values and store them in a database without the intervention of the human element and with a minimal error rate.

- Determining the appropriate algorithms in the field of object detection and determining the scope of work of each algorithm, whether it is the YOLOV3 algorithm or Faster-RCNN.

- Determining the appropriate mode for taking pictures in proportion to the algorithm detecting the objects to be used.

- Contributing to reducing operational costs by 99% with regard to projects of object detection in the work system for commercial companies.

1.6. Thesis Organization

This thesis is divided into six chapters:

- Chapter one: introduces the work and the motivation with the current literature on object detection algorithms.

- Chapter two: will discuss the architecture of the Convolutional Neural Network and Deep Learning.

- Chapter three: is the general description of the architecture of the object detection system algorithms the object detection metrics and the pre-trained models.

- Chapter four: will include the implementation of our proposed systems, with a comparison of the performance of the two proposed systems.

- Chapter five: will summarize and discuss the final results.

- Chapter six: the thesis will be concluded with a summary of the research and the future works.

CHAPTER 2. CONVOLUTIONAL NEURAL NETWORK AND DEEP LEARNING

Due to its ability to handle large amounts of data, deep learning has emerged as a crucial tool in recent decades. The use of hidden layers in pattern recognition has overtaken traditional methods. Convolutional neural networks are widely used deep neural networks [12]. Researchers have been battling to create an AI system that can understand visual input since the 1950s, when AI was still in its inception. Eventually, computer vision was renamed to reflect its growing importance in the industry. At the University of Toronto in 2012, a team of researchers produced an AI model that outperformed the top picture-recognition algorithms by a wide margin. With an astounding 85% accuracy, AlexNet AI (named after its founder, Alex Krizhevsky) took first place in the 2012 ImageNet computer vision challenge. The second-place finisher managed an impressive 74% on the exam. Convolutional neural networks, a form of neural network that mimics human vision, built AlexNet. Convolutional neural networks (CNNs) have grown significantly in many computer vision applications in recent years [13].

2.1. Convolutional Neural Network Layers

There is an increasing need to extract useful information from ever-increasing amounts of visual input. With a convolutional neural network (CNN) model, accessing this critical information with a lower number of perceptrons is not conceivable. CNN is now being used to extract visual data features [14]. In computer vision, object identification, image processing, and classification, CNN is a deep learning method based on ANN [15]. Feature extraction is not used by CNN. It is through the use of convolutional layers that the model learns how to extract features accurately. CNN's rise in popularity can be attributed to the growth of computing resource hardware equipment, focused on graphical processing units (GPUs) and image processing domains connected to image filters. With all of these improvements over time, CNN is currently being used to classify images and videos using its many layers. Preparing data for convolution in advance ensures that the process will run smoothly and without interruption. Adjusting picture and video sizes could be part of this setup. The convolutional and pooling layers of a typical CNN are applied in a precise order. There are two layers of features: convolution and pooling. In the flattening and related layers, characteristics are received. The classification is determined by the Softmax activation layer, which is applied at the end of the procedure. Figure 2.1. displays CNN's overall design and structure.

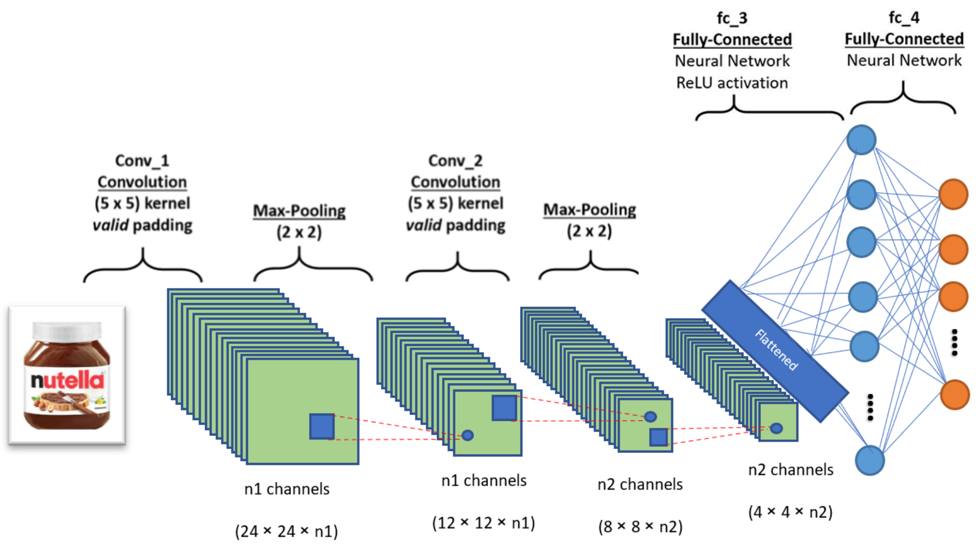


Figure 2.1. Convolutional Layers Architecture

2.1.1. Convolutional layer

A convolutional neural network is a deep neural network that recognizes patterns in images in deep learning. They can be used for several activities, including geographical data analysis, computer vision, natural language processing, signal processing, and more. The architecture of a convolutional network is inspired by the arrangement of the Visual Cortex and fits the neuronal connection pattern in the human brain. Convolution is a process that transforms one function into another by integrating data from two different sources in a systematic way. Convolution is a key procedure in an ANN that gives rise to this particular sort of Artificial Neural Network. There are a variety of different uses for convolutions besides blurring and sharpening pictures in image processing (for example, emboss and enhance the edges). CNNs enforce the local connection arrangement between neurons in surrounding layers. CNNs employ filters (also known as kernels) to determine whether specific properties, such as edges, are present throughout a picture. There are several primary operations in a CNN: Convolutional Non-linearity (ReLU) Pooling or subsampling classification (Fully Connected Layer). The initial layer of a CNN is usually a Convolutional Layer. Before transferring the result to the next layer, convolutional layers perform a convolution on the input. In convolution, all of the pixels in the receptive region are transformed into a single value. A picture may be condensed into one pixel via convolution, which concurrently decreases the image's size (see Figure 2.2.). The convolutional layer produces a vector as its final output. In this thesis a number of convolutions are used depending on the sort of issue attempting to solve and the features want to learn [15].

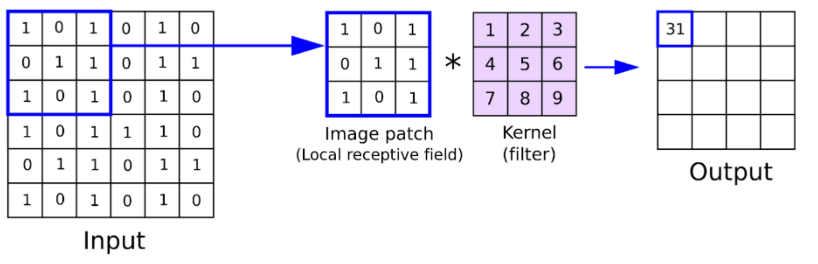


Figure 2.2. Convolution operation

2.1.2. Pooling layer

Pooling layers come in various shapes and sizes in different CNN architectures. Still, they all have the same goal: to gradually shrink the network's spatial extent, which reduces the network's parameters and overall computation. The Pooling layer is implemented using Max-pooling and Mean-Pooling, as shown in Figure 2.3. Choosing a kernel size and stride length during architecture design is required for the Max-Pooling process. Once selected, the operation slides the kernel over the input with the specified stride, only setting the most significant value from the input to produce an output value at each kernel slice. Because it mixes pixels in close proximity in the pooling layer reduces the size of the image by converting a specified area into a single representative value. Pooling is a typical image processing technique that has been applied in a variety of other systems. It is important to learn how to choose the pooling pixels from the image and calculate the representative value before proceed with the activities in the pooling layer. Neighbouring pixels are frequently selected through the square matrix, which can have a variable number of entries. The representative value is usually the mean or maximum of the selected pixels. The pooling layer's operation is very straightforward. Because this is a two-dimensional procedure, and a text explanation may cause additional confusion. For example, consider the 4×4 pixel input image, represented in Figure 2.3. by the matrix. The pooling layers of the VGG architecture, for example, use a 2x2 kernel with a stride of two with these criteria, 75% of the activations are deleted, making VGG more computationally efficient and preventing overfitting by eliminating so many variables [15].

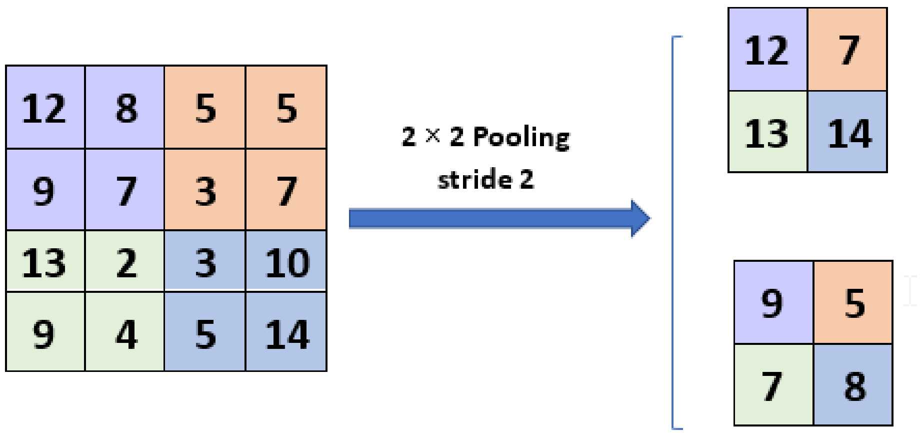


Figure 2.3. Pooling Layer

2.1.3. Batch normalization

Because the input from previous layers may fluctuate as a result of weight adjustments for each mini-batch, batch normalization is used to standardize the inputs to a layer. Deep neural networks are difficult to train because they are sensitive to weights and contain numerous layers to the learning algorithm's initial randomness. Batch normalization is a deep neural network training strategy that assigns a layer to each mini-input batch. This significantly decreases the number of training epochs required to develop deep networks by stabilizing the learning process. By expediting training (in some cases by halving or enhancing epochs) and providing some regularization, batch normalization reduces generalization error [16].

2.1.4. Dropout layout

The dropout layer is vital to avoid overfitting by ignoring parameters with a specific rate between 0 and 1. Training performance may be improved by deleting some of the network parameters. This layer can be used to randomly remove connections between perceptron in networks with large structures during training. In most cases, this layer is sandwiched between two other layers. A dropout process is presented in Figure 2.4., which depicts a connected layer net and how it is converted after applying dropout operations.

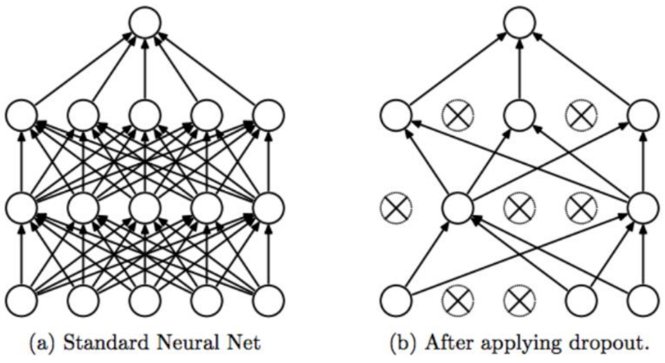


Figure 2.4. before and after Dropout Layer

2.1.5. Fully connected layer

Fully connected layer it is the layer that precedes the last layer in the model, which is made up of the previous layers (all convolutions and pooling layers). It is considered as the input layer for the last layer of the model by setting all previous values into one vector as presented in Figure 2.5. and Figure 2.6. All inputs to all neurons are sent through the flattening layer's completely connected layer. Multilayer ANN calculations are identical to the processes done in fully connected layers. Once these layers are in place and ready to be activated, the architecture will be complete to solve classification issues. A SoftMax function selects the optimal multi-output choice with the highest probability [17].

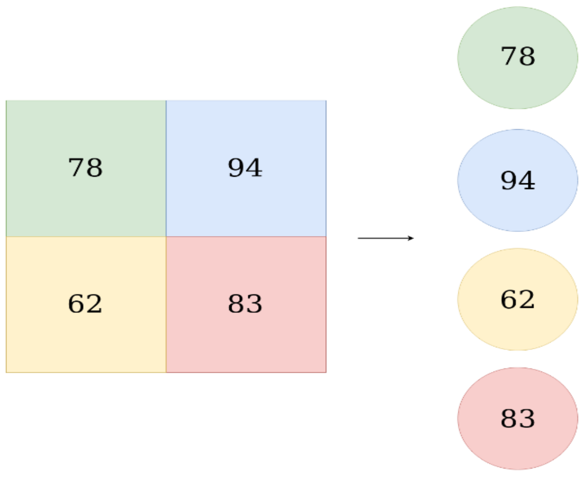


Figure 2.5. Flattening

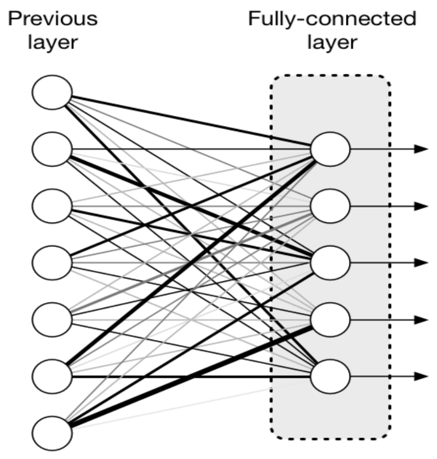


Figure 2.6. Fully Connected Layer

2.2. Activation functions

When an artificial neural network is trained to recognize complicated patterns in data, an activation function is included in the network. Compared to the neuron-based that exist in our brains, the activation function ultimately decides what signals are sent to the next neuron. The activation function is precisely the same in an ANN. Output signal are taken in and converted into something that can be input to the following cell [18].

2.2.1. ReLU

To activate a node, the activation function takes the total of its weighted inputs and turns it into the node's activation or output. It is possible to use the ReLU activation function or ReLU for short to immediately output the input value if it is positive; otherwise, is obtained zero. It has been the default activation function for many neural networks since it is faster to train and typically produces superior results. It can be written as f(x)=max(0,x) and it is represented graphically as appears in Figure 2.7. [19].

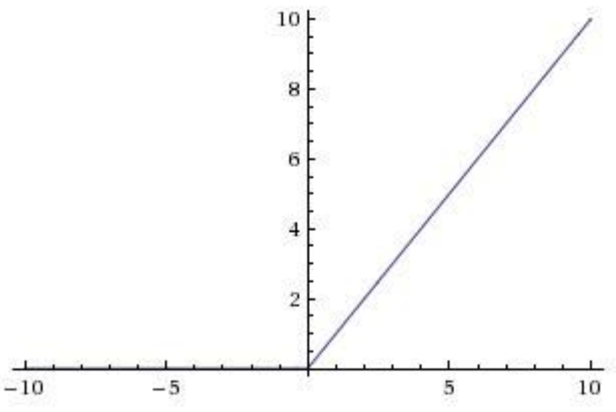


Figure 2.7. ReLU Function

2.2.2. SoftMax

The SoftMax mathematical function, which multiplies all the values in an array by the scale, may be used to express a collection of integers as probabilities. In neural network models for applied machine learning, the SoftMax function is most commonly utilized as an activation function. N values are needed to complete the classification job, one for each class. The weighted sum values are converted to probabilities that total to one using the SoftMax algorithm. The output of the SoftMax function (see Figure 2.8.) is used to determine the likelihood of belonging to each class [19].

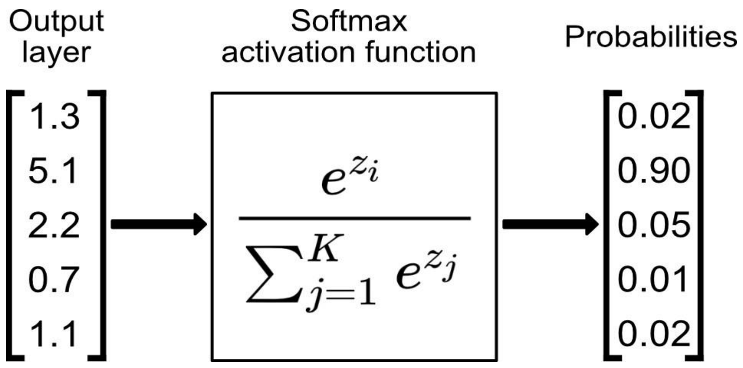

Output layer
1.3
5.1
2.2
0.7
1.1
Softmax activation function
$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
Probabilities
0.02
0.90
0.05
0.01
0.02

Figure 2.8. Softmax activation function

CHAPTER 3. OBJECT DETECTION TECHNOLOGIES

3.1. Object Detection Algorithms

It might be difficult to tell apart computer vision tasks that are closely related. The distinctions between object localization and object detection, for example, might be perplexing when all tasks are referred to as "object recognition." In image classification, a label is assigned to an image, whereas object localization is the process of drawing a box around one or almost more objects in an image. There are two jobs involved in object detection: drawing a box around each object in the image and assigning it a name. Object recognition encompasses all of these issues. This section goes through the current status of deep learning models for object recognition and how they work in this study [20]. RCNNs and YOLO Algorithms exposed The term "object recognition" refers to a group of interconnected tasks aimed at identifying things in digital images. Object localization and recognition tasks can be addressed with R-CNNs, or Region-Based Convolutional Neural Networks, which are built for model performance. YOLO, an acronym for "You Only Look Once," is a group of object identification methods geared toward speed and real-time use. As a result, object detection networks can identify the type of object in a picture as well as its precise location. Same with image classification networks, they use convolution layers to identify visual characteristics. It is common practice for object detection networks to reuse an image classification CNN. As a supervised machine learning task, detecting objects necessitates using samples with labels. Object boundaries and classes must be included in an accompanying file for each training image. Object detection annotations can be generated using various open-source technologies such as labelImg tool and RoboFlow platform [21].

3.1.1. R-CNN

Images have been classified using CNNs on a large scale. Making accurate bounding boxes for objects in images is a difficult challenge to tackle. The R-CNN algorithm was developed in 2014 to address this issue. There have been numerous new algorithms developed based on R-CNN since R-CNN, such as Fast RNN and Faster RNN [22]. Consequently, object detection performance was improved as a result. The object detecting system is made up of three parts. The first one generates region suggestions that aren't categorized. Detector will be able to choose from a list of possible detections based on these proposals. Convolutional neural networks are used to extract feature vectors from each region of an image. Then the third module is a set of linear SVMs for each of the classes [23].

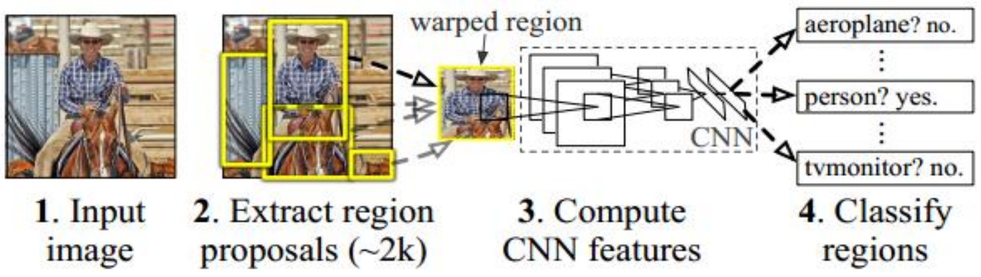


Figure 3.1. The Procedure of R-CNN [22]

As seen in Figure 3.1. approximately 2000 regions were extracted as input from the image that used this method. It is then handed on to a CNN as a fixed-size input for each region proposal. A fixed-length feature vector is generated by the CNN for each region suggestion. In order to classify region proposals, category-specific linear SVM is utilized. Refinement of the bounding boxes is done using bounding box regression to ensure that the object is caught by the box [22].

3.1.2. Fast R-CNN

The object detector created by Microsoft's AI researcher named Ross Girshick [24]. Fast R-CNN solves a number of R-CNN problems. The Fast R-CNN has one benefit over R-CNN in terms of speed. All ROIs in the same image will be extracted into a new layer called ROI Pooling, which extracts feature vectors of equal length from all proposals (or ROIs). Comparatively, Faster R-CNN creates a single-stage network, while R-CNN has several steps (region proposal generation, feature extraction, and classification using SVM) [24]. An R-CNN that is more efficient uses convolutional layer calculations (i.e. computations) over all of the proposals (i.e. ROIs). As a result, Fast R-CNN outperforms regular R-CNN in terms of speed. Unlike R-CNN, which necessitates hundreds of gigabytes of storage, Fast R-CNN does not cache the extracted features. Fast R-CNN outperforms R-CNN in terms of accuracy.

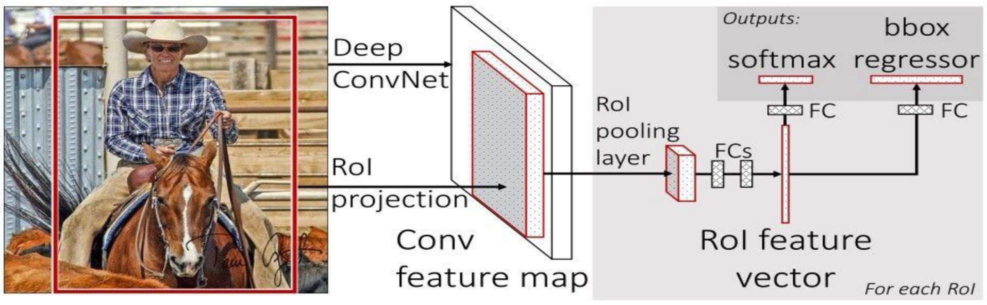


Figure 3.2. Fast R-CNN general architecture [24]

A more outstanding detection quality (mAP) can be achieved with Fast R-CNN, although it is still limited by the need to identify objects. Therefore, Faster R-CNN was presented to tackle this difficulty. Fast R-CNN's architecture is shown in Figure 3.2. In contrast to R-CNN, which has three stages, this model has only one step. There is nothing more to it than passing in an image and getting back the object's class probability and bounding box as output [24].

3.1.3. Faster R-CNN

The extension name of Fast R-CNN is "Faster R-CNN". Because of the region proposal network (RPN), Faster R-CNN outperforms Fast R-CNN [25]. A region proposal network (RPN) and a network that uses these proposals to recognize objects make up the Faster R-CNN. Fast R- CNN differs from this approach in that it generates region recommendations through selected search rather than random search. When RPN shares the majority of processing with the object detection network, the time it takes to generate region recommendations is substantially smaller than in selective search. Briefly, RPN ranks region boxes (called anchors) and proposes the ones most likely containing items. The Figure 3.3. is a breakdown of the design [26].

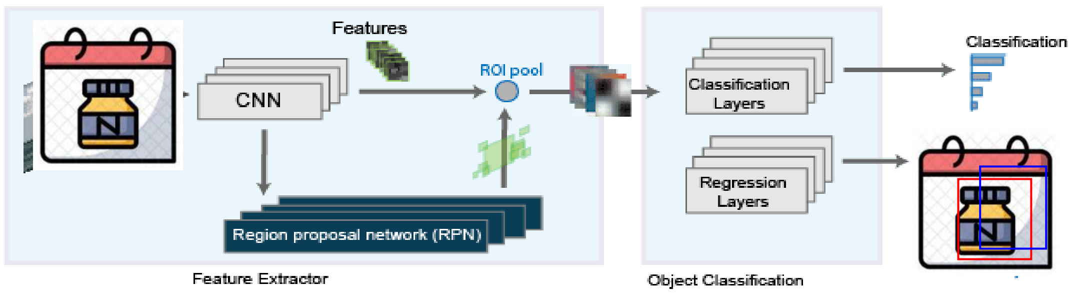


Figure 3.3. Faster R-CNN Architecture

Faster R-CNN relies heavily on the anchors. An anchor is a box. Faster R-CNN is preconfigured with nine anchors at each image location. Analyzing an image with a size of (600, 800), it can be seen 9 anchors at the location (320, 320), as shown in Figure 3.4. [25].

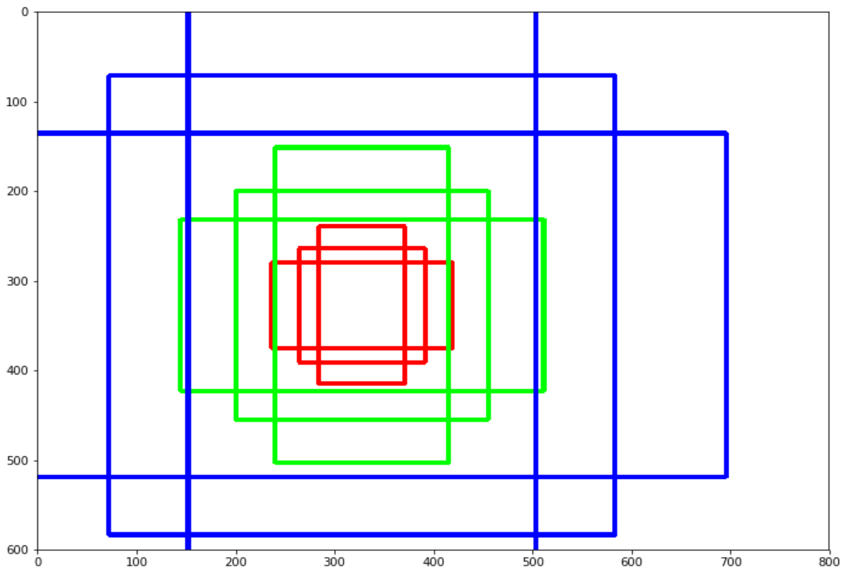


Figure 3.4. Faster R-CNN Anchors

3.1.4. YOLO

To see anything, we rely on our eyes, which gather data from the scene and convey it to our brains. We only need to look to gain an understanding of the objects we're observing. But the processing our brain does for it is just beyond compare. We can reverse this concept on machines. As a result, the machine's job will be made considerably more accessible, and it will be able to interact with the items in its immediate environment much more effectively. Ultimately, the goal is to build machines that are more human-like in their interaction with humans. Object detection is one of the core areas of computer vision study. It deals with detecting instances of numerous types of things (such as a person, book, chair, automobile, bus, etc.). Object detection has found its applications in different sectors, including self-driving automobiles, robots, video surveillance, object tracking, etc. This area is further separated into sub-domains, including face detection, activity recognition, image annotation, and many more [27]. YOLO is a real-time object detection algorithm based on neural networks. Because of its speed and accuracy, this algorithm has gained widespread adoption. It's been put to good use spotting traffic lights, people, parking meters, and animals, among other things. Convolutional neural networks may be used for a variety of fascinating tasks, one of which is image classification. There are many exciting challenges in computer vision beyond simple image categorization, with object detection being one of the most engaging of these. YOLO ("You Only Look Once") is an effective real-time object recognition method, initially published in the seminal year 2015 on paper by Joseph Redmon et al. The YOLO method and one of its open-source implementations, Darknet and TF-2 Model Zoo, are introduced in [28].

The R-CNN approaches broadly use regions to localize the objects in an image. The network does not look at the entire image, only at the areas of the images which have a higher likelihood of containing an object. The YOLO algorithm (You Only Look Once), on the other side, handles object detection in a new attitude. It takes the complete image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes. YOLO's greatest strength is its lightning-quick processing speed of 45 frames per second. YOLO also recognizes generalized object representation. This is one of the best approaches for object detection and has shown a substantially good correlation to the R-CNN techniques. A variety of YOLO algorithm strategies will be discussed in the following sections [29].

3.1.4.1. YOLOv2

The YOLOv2 object detector uses a single-stage object detection network. In comparison to other object detection algorithms, such as Faster R-CNNs, YOLOv2 is significantly faster. YOLO is one of the top models in object identification, able to distinguish objects and process up to 150 FPS as a rate for small networks. With training on PASCAL VOC 2007 and 2012, YOLO's mAP (mean average precision) is a respectable 63% compared to the current best model. With a mAP of 71%, it outperformed models like the Faster R-CNN and SSD [30]. To provide network predictions, as input, the YOLOv2 model employs a CNN algorithm. In order to build bounding boxes, the object detector decodes the predictions. YOLOv2 employs anchor boxes to detect classes of items in an image. Intersection over Union (IoU) forecasts the objectness score of each anchor box, anchor box offsets refine the position of the anchor box, The class probability forecasts the class label assigned to each anchor box in the YOLOv2. A feature extractor network is used to get the model started, which can be initialized using a pre-trained CNN or trained from scratch. The YOLOv2 transform layer and YOLOv2 Output Layer objects are the first two layers of the detection subnetwork, followed by the Conv, Batch norm, and ReLu layers. The YOLOv2 transform layer turns the raw CNN output into a form required to create object detections. YOLOv2 Output Layer defines the anchor box settings and implements the loss function needed to train the detector. Later YOLOv3 is presented to boost the performance and accuracy of (mAP) for the YOLO algorithm (see Figure 3.5.) [31].

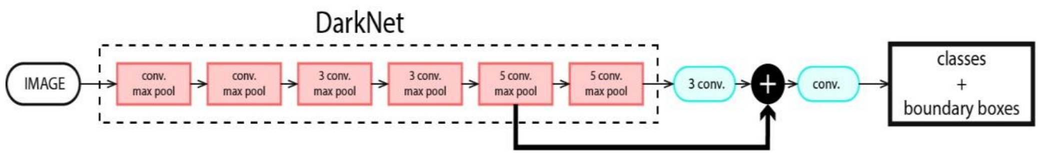
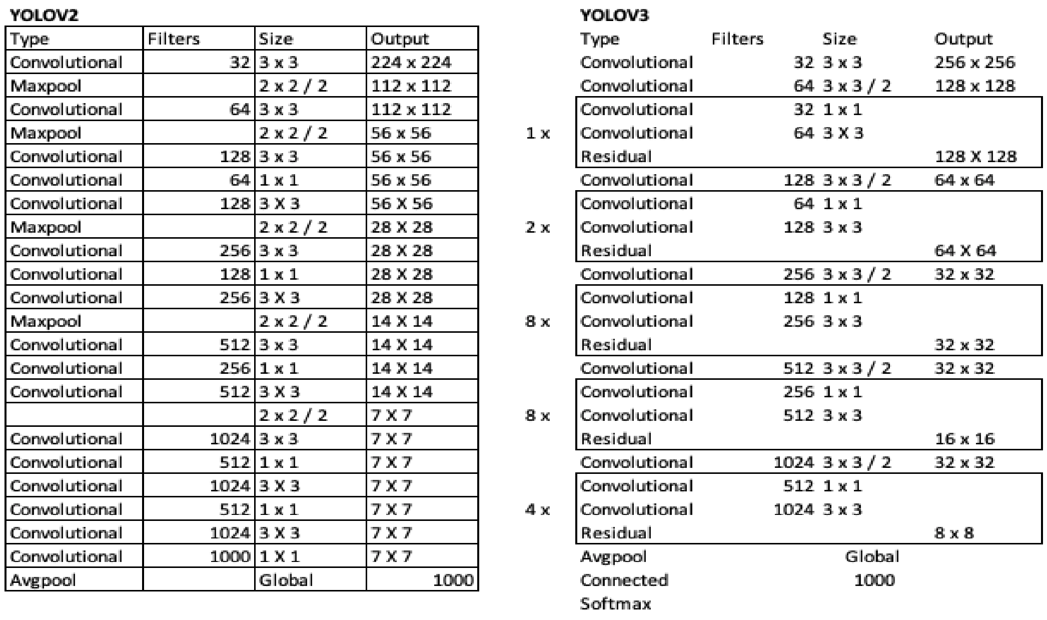


Figure 3.5. YOLOv2 with Darknet

3.1.4.2. YOLOv3

There are 106 detecting layers in YOLOv3, making it faster than YOLOv2 algorithm. YOLOv3 is a neural network with residual blocks and average pooling networks. While YOLOv2 utilizes the DarkNet-19 as the model architecture, YOLOv3 employs a significantly more complicated DarkNet-53 as the model backbone YOLOv3 was proposed by J. Redmon and A. Farhadi J. Redmon and A. Farhadi [32]. It is possible to make YOLO even more effective by using current CNNs that utilize residual networks and skipping connections. For these predictions, the feature maps at layers 82, 94, and 106 were extracted by YOLOv3 because of their unique architectural design [32]. YOLOv3 makes up for YOLOv2 and YOLO's inadequacies, particularly in the identification of smaller objects, by recognizing features at three different scales. Because of the architecture, it is possible to combine the upsampled layer outputs with characteristics from previous layers, the fine-grained characteristics that have been obtained are preserved, thus making the detection of tiny objects easier. There are only three bounding boxes predicted per cell in YOLOv3 (as opposed to five in YOLOv2). However, there are three predictions made at three distinct scales, adding up to nine anchor boxes in all. Because of its speed and accuracy, YOLO is an excellent alternative to other social media networks. As a result, the model's predictions take into account the image's overall context when being tested [33]. YOLO and other convolutional neural network algorithms "score" regions based on their resemblance to predefined classes. Positive detections of the class they most closely resemble are noted in high-scoring locations. According to the video's regions that score highly against predetermined classes of cars, YOLO may be used to recognize different types of automobiles on a live feed of traffic [34]. In Figure 3.6., given the architecture of YOLOv3, it is possible to see how YOLOv3 is more powerful than since YOLOv3 by dependent on residual blocks, YOLOv3 detects at three independent levels: 82, 94, and 106, even though the core convolution layers are the same [35].



YOLOV2

Type	Filters	Size	Output
Convolutional	32	3 x 3	224 x 224
Maxpool		2 x 2 / 2	112 x 112
Convolutional	64	3 x 3	112 x 112
Maxpool		2 x 2 / 2	56 x 56
Convolutional	128	3 x 3	56 x 56
Convolutional	64	1 x 1	56 x 56
Convolutional	128	3 X 3	56 X 56
Maxpool		2 x 2 / 2	28 X 28
Convolutional	256	3 x 3	28 X 28
Convolutional	128	1 x 1	28 X 28
Convolutional	256	3 X 3	28 X 28
Maxpool		2 x 2 / 2	14 X 14
Convolutional	512	3 x 3	14 X 14
Convolutional	256	1 x 1	14 X 14
Convolutional	512	3 X 3	14 X 14
		2 x 2 / 2	7 X 7
Convolutional	1024	3 x 3	7 X 7
Convolutional	512	1 x 1	7 X 7
Convolutional	1024	3 X 3	7 X 7
Convolutional	512	1 x 1	7 X 7
Convolutional	1024	3 X 3	7 X 7
Convolutional	1000	1 X 1	7 X 7
Avgpool		Global	1000

YOLOV3

	Type	Filters	Size	Output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3 / 2	128 x 128
	Convolutional	32	1 x 1	
1 x	Convolutional	64	3 X 3	
	Residual			128 X 128
	Convolutional	128	3 x 3 / 2	64 x 64
	Convolutional	64	1 x 1	
2 x	Convolutional	128	3 x 3	
	Residual			64 X 64
	Convolutional	256	3 x 3 / 2	32 x 32
	Convolutional	128	1 x 1	
8 x	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3 / 2	32 x 32
	Convolutional	256	1 x 1	
8 x	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3 / 2	32 x 32
	Convolutional	512	1 x 1	
4 x	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.6. YOLOv3 vs YOLOv3 Architecture [30][33]

In YOLO v3, classes are more detailed and have numerous bounding boxes because of a multilabel approach. The probability of each value in the vector is proportional to the relative scale of each value in YOLOv2's vector, while a SoftMax is a mathematical function utilized by YOLOv2 [35]. When comparing YOLO with RetinaNet, the YOLOv3 AP does show a trade-off between speed and accuracy because the RetinaNet training time is longer than the YOLOv3 training time. However, the accuracy of detecting objects with YOLOv3 may be equivalent to the accuracy when using RetinaNet by having a larger dataset, making it a great alternative for models trained with massive datasets. An example of this would be typical detection models like traffic detection, where enough data may be used to train the model since the number of photos of different vehicles is plentiful. YOLOv3, on the other hand, may not be the best choice for specialty models with limited access to large datasets [35].

3.2. Metrics in Object Detection

In the term computer vision, object detection is one of the significant algorithms which helps in the classification and localization of an item. The big challenge of detecting objects is drawing the bounding boxes around each detected object. While deep-diving through research papers, you may notice these acronyms (AP, IOU, and mAP). These are nothing but object detection metrics that help in finding effective models. In this study, it is illustrated how we can get a clear understanding of those terms and the benefit of each one of them.

3.2.1. Intersection over union (IOU)

Intersection over union is an evaluation metric used to quantify the accuracy of an item detector on a particular dataset. Any technique that delivers predicted bounding boxes can be assessed using IoU, and this metric is employed in most object detection algorithms. Using the model's confidence scores generates numerous bounding boxes for each object and then eliminates any that aren't necessarily based on the confidence score thresholds [36]. We need to define the threshold value based on criteria. Formal definition: to assess an object detector using intersection over Union, we need the hand-labeled bounding boxes from the testing set to represent the ground truth bounding boxes, which indicate where an object is located in an image and the model's estimated boundary boxes. We can perform the Intersection over union calculation since we have two sets of bounding boxes. Figure 3.8. is an example of a ground-truth bounding box against a forecasted bounding box, and Figure 3.9. represents the equation of IoU [36]. Finally, if the IOU score is more significant than 0.5 is considered a good prediction (see Figure 3.7.).

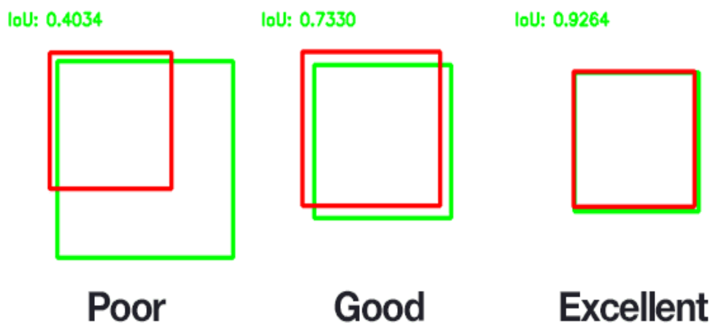


Figure 3.7. Evaluation of prediction using IoU

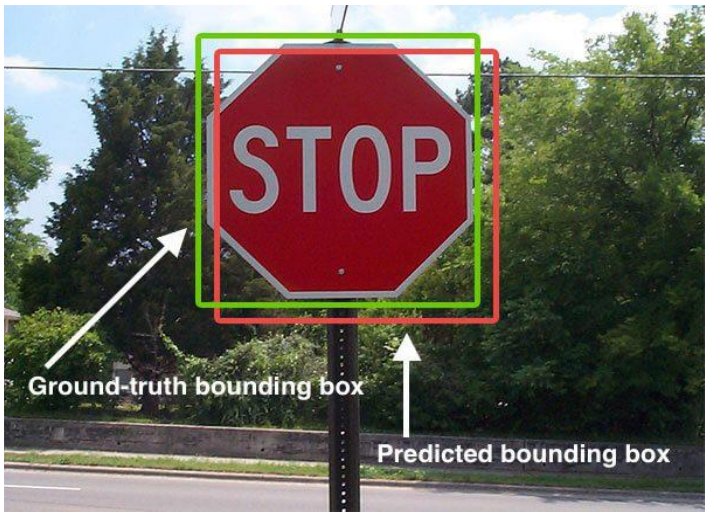


Figure 3.8. How to get Intersection Over Union

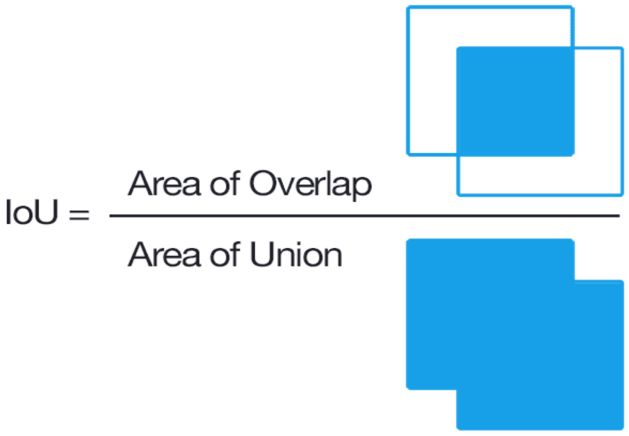


Figure 3.9. IoU Equation

3.2.2. Average precision

Average Precision (AP) can only be fully appreciated by first grasping the concepts of precision and recall. In other words, how accurate are your projections? It determines how many of your model's predictions were correct. Since we know the values for the True Positive (TP) and The False Positive (FP), we can calculate TP, which is shown in Figure 3.10. The True Positive Means predicted as positive as was correct and the False positive means predicted as positive but was incorrect [37].

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

Figure 3.10. Calculation of Precision

On the other hand, recall is a metric that assesses how effectively you can discover all the positive aspects of a given situation. For instance, our top K predictions reveal 80 percent of the probable favorable outcomes. True Positive values and or False Negative (FN) values are needed to calculate this metric. How to compute recall is depicted in Figure 3.11.

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

Figure 3.11. Calculation of Recall

Precision-recall curves are widely used for this purpose, but since the average precision yields numerical values, it is easier to compare the performance of different models. It summarizes the weighted mean of precisions for each threshold with the rise in recall using the precision-recall curve. Each object's average precision (AP) is computed. Figure 3.12. shows how we can calculate the AP [37].

$$\boldsymbol{AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k)}$$
$$\boldsymbol{Recalls(n) = 0, Precisions(n) = 1}$$
$$\boldsymbol{n = Number\ of\ thresholds.}$$

Figure 3.12. Calculation of Average Precision

3.2.3. Mean average precision(mAP)

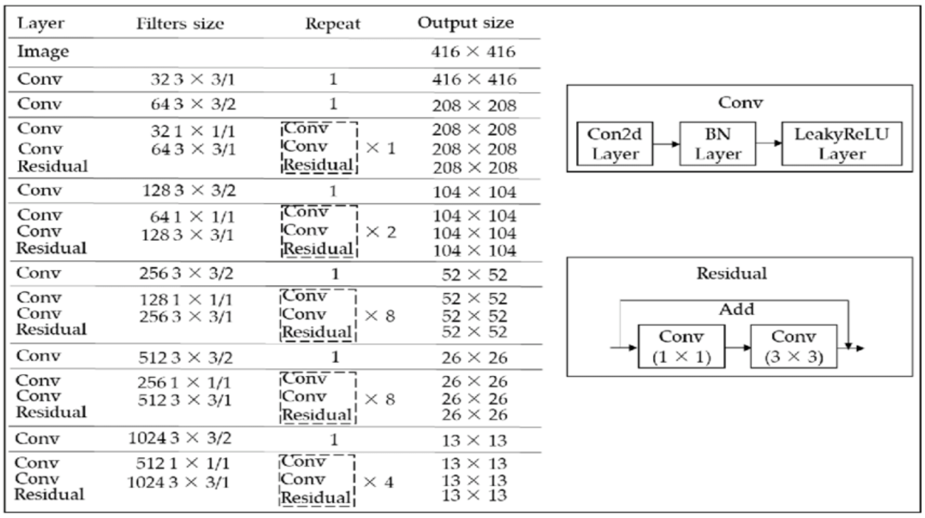
Mean average precision (mAP) is an extend of average precision (AP). The major difference between (AP) and (mAP) is, in mAP, the precision of the entire model is calculated, whereas, in average precision (AP), just the precision of individual objects is calculated. The mAP is being used to calculate the model's accuracy rate. In computer vision, mAP is a common evaluation metric used for object detection such as localization and classification to be reflected later to values for classes and boxes drawn around the object (e.g., a dog or cat example) [37]. We can see more than one extension of these metrics in the research, like mAP iou = 0.5 or mAP iou = 0.75, and sometimes, mAP small, medium, and large. There are explain to each one of these extensions we will clearly demonstrate explain. The mAP iou=0.5 signifies that the model has evaluated the value 0.5 as a threshold value to eliminate extraneous bounding boxes. It is the usual threshold value for most of the models [38]. We can achieve accurate results by deleting bounding boxes with less than 25% of the intersection with the ground truth picture when we choose mAP iou=0.75 as our threshold value. The mAP small denotes that the model's mAP score is based on the data's smaller items. mAP large denotes the model has provided a mAP score based on more oversized items in the data [38].

3.3. Pre-trained models

Pre-trained models are a good source of support for those wishing to learn an algorithm or try out an existing framework. When constructing a model from scratch isn't an option due to time or computing constraints, pre-trained models are a great alternative. You can use a pre-trained model as a standard against which to measure the performance of your custom model. The potential and possibilities are tremendous. There are a lot of pre-trained models available to use, but in our study, because we use the YOLO algorithm and Faster - RCNN, we will focus on Darknet and TF2-model Zoo as pre-trained models to achieve our goal. Later we talk about each one of these models separately.

3.3.1. Darknet model

In the previous paragraphs, we talked about the YOLO algorithm. We have mentioned that the YOLOv2 and v3 have been built on Darknet pre-trained model. YOLOv2 was built on Darknet-19, whereas YOLOv3 was made on Darknet-53. Therefore, Darknet is considered the vital model in real-time object detection using YOLO algorithms [32]. For the development of neural networks, Darknet is a high-performance open-source framework. Written in C and CUDA (a software layer that gives direct access to the GPUs), it is also possible to combine Darknet with CPUs and GPUs. Larger deployments of deep neural networks may be done utilizing Darknet. ImageNet classification, recurrent neural networks (RNNs), and You Only Look Once (YOLO) for real-time object recognition are a few examples of these techniques [39]. To better understand Yolov3, we'll discuss the architecture of Darknet-53, a convolutional neural network with 53 layers that we can use to train on more than a million photos from the ImageNet database. Darknet-53 is a pre-trained model version of the network. More than a thousand item classifications may be achieved using the pre-trained network, such as planes, cars, and numerous animals. Consequently, the network has amassed a diverse set of feature representations. The network has an image input size of 256×256. DarkNet-53 is commonly used as the foundation for object detection difficulties and YOLO procedures. Figure 3.13. describes the architecture of Darknet-53 [32].



Layer	Filters	size	Repeat	Output size
Image				416 × 416
Conv	32	3 × 3/1	1	416 × 416
Conv	64	3 × 3/2	1	208 × 208
Conv	32	1 × 1/1	Conv × 1	208 × 208
Conv	64	3 × 3/1	Conv	208 × 208
Residual			Residual	208 × 208
Conv	128	3 × 3/2	1	104 × 104
Conv	64	1 × 1/1	Conv × 2	104 × 104
Conv	128	3 × 3/1	Conv	104 × 104
Residual			Residual	104 × 104
Conv	256	3 × 3/2	1	52 × 52
Conv	128	1 × 1/1	Conv × 8	52 × 52
Conv	256	3 × 3/1	Conv	52 × 52
Residual			Residual	52 × 52
Conv	512	3 × 3/2	1	26 × 26
Conv	256	1 × 1/1	Conv × 8	26 × 26
Conv	512	3 × 3/1	Conv	26 × 26
Residual			Residual	26 × 26
Conv	1024	3 × 3/2	1	13 × 13
Conv	512	1 × 1/1	Conv × 4	13 × 13
Conv	1024	3 × 3/1	Conv	13 × 13
Residual			Residual	13 × 13

Figure 3.13. Darknet-53 Architecture [32]

3.3.2. TF-2 model zoo

To understand the models in TF2-Model Zoo, we must first understand TensorFlow. TensorFlow leverages data flow graphs, an open-source artificial intelligence framework to create models. It permits the building of multi-layered, large-scale neural networks. TensorFlow is most commonly used for classification, perception, understanding, discovery, prediction, and creation [40]. Using TensorFlow, you can build and run deep neural networks for a variety of tasks, including handwritten digit classification and image recognition, word embeddings, recurrent neural networks, machine translation sequence-to-sequence models, natural language processing, and simulations based on partial differential equations. (PDEs). Open-sourced object detection models may be found in the TF OD API, which is utilized by deep learning researchers to recognize objects in images. NLP, object identification, and many more subjects benefit from Tensorflow, the technology that underlies many SOTA models in these areas [40]. Debugging object detection models in the new TF2 OD API is significantly more accessible because of eager execution and new SOTA models that are supported in the new API (TF2 Model Zoo). Model zoo, a pre-trained model on the MS COCO 17 data set and can detect and classify 90 objects [41]. For out-of-the-box inference or for initializing your models while training on new datasets, pre-trained models might be handy. This is true if we are interested in categories already present in the dataset. There are a lot of available models in the TF2-Model Zoo. In our study, we will focus on the Models that give bounding boxes as a result of object detection, especially those that use Faster – RCNN architecture [40].

3.4. Dataset Preparing Tools

It is essential for various applications such as machine learning-based image interpretations, computer vision, robotic vision, and facial recognition. It is necessary to associate photos with metadata such as IDs, captions, or keywords to train these solutions. There is a wide range of applications that require large amounts of annotated images, from self-driving cars and machines that select and sort products to healthcare apps that automatically diagnose medical disorders. Annotation improves the accuracy and precision of these systems by efficiently teaching them. In this study, we will learn about the essential tools to achieve this purpose [42].

3.4.1. labelImg tool

Deep neural networks are trained using training datasets, which are collections of tagged pictures. There are a variety of ways to prepare and annotate datasets for object-detection training purposes. The most popular formats for annotating datasets are Pascal VOC and Microsoft COCO. Because we are utilizing YOLOv3 in our study, we need to annotate our data set in the Pascal VOC annotation format, and the ideal tool for this is LabelImg. You may use LabelImg to graphically label pictures for free and open-source. Python is used for the code, while QT is used to create the program's graphical user interface. It's a simple and cost-free method of labeling a large number of photos for the purpose of object detection. Figure 3.14. showing how LabelImg is working [43].

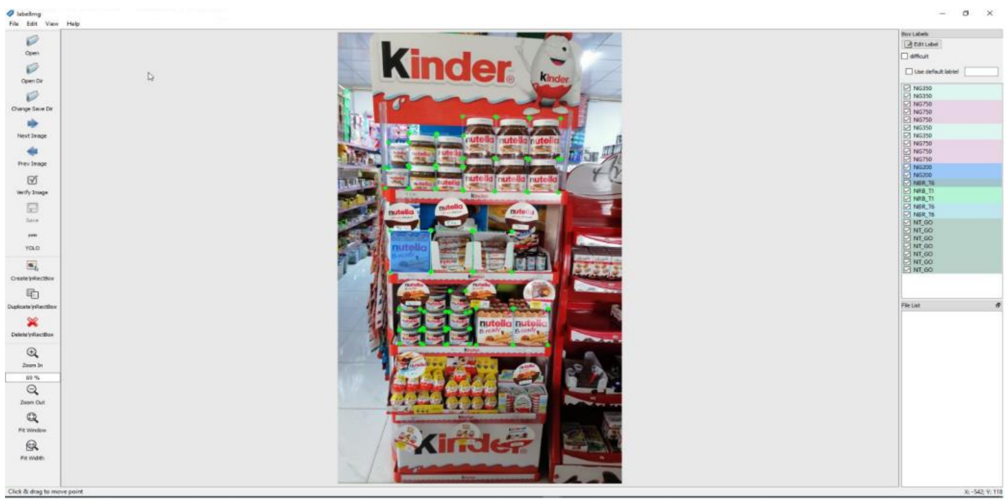


Figure 3.14. LabelImg tool

3.4.2. Roboflow framework

As we mentioned in the previous paragraph, the most popular formats for annotating datasets are Pascal VOC and Microsoft COCO. Because working with TF2-Model

Zoo pre-trained models that are pre-trained on COCO Dataset, we will prepare our dataset with the Microsoft COCO dataset annotated format. For this reason, we will depend on RoboFlow to achieve our goal. Computer vision developers can use Roboflow's data gathering, preprocessing, and model training tools to improve their workflows. It is possible for users to add their own data to Roboflow, as well as public datasets. A variety of annotation formats are supported by Roboflow. Steps such as image rotation and resizing are all part of the data preprocessing process, as contrast adjustments and data augmentations. There are teams who can manage the entire process. Model libraries like EfficientNet, MobileNet, Yolo, TensorFlow, PyTorch, and others are available for model training. As a result, the whole state-of-the-art can be accessed via model deployment and visualization choices [44].

CHAPTER 4. DESIGN AND IMPLEMENTATION OF THE OBJECT DETECTION SYSTEM FOR FMCG COMPAINES MANAGEMENT

As mentioned in Chapter 1, many studies provide solutions using various deep learning algorithms. Accordingly, we will discuss in this chapter of our research to clarify the steps necessary to use deep learning techniques, particularly the object detection system. To reach our desired goal, the first process in this chapter is the creation and initialization of the data set. After that, YOLOv3 and Faster-RCNN techniques were used to train the model on our data, and then this model was used to detect objects in images that the model had not previously seen. Google Colab platform was used to train the models. The Google Drive API was also relied on to extract results from images and convert the results into readable values placed inside the MySQL database and then display the results using Python libraries and Microsoft Power BI. Figure 4.1. shows the brief about our implementation steps of our object detection system (ODS).

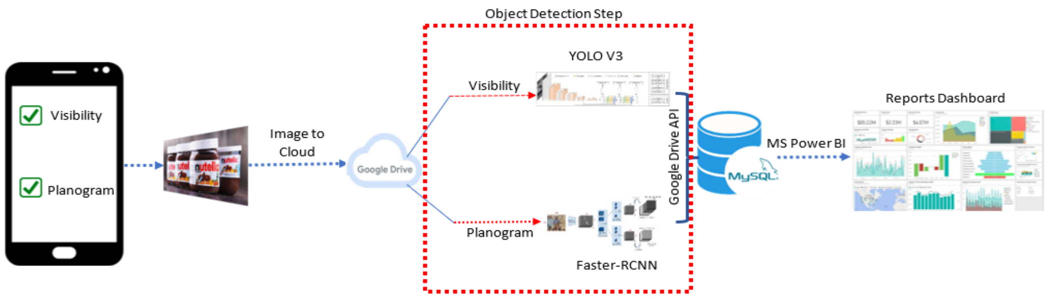
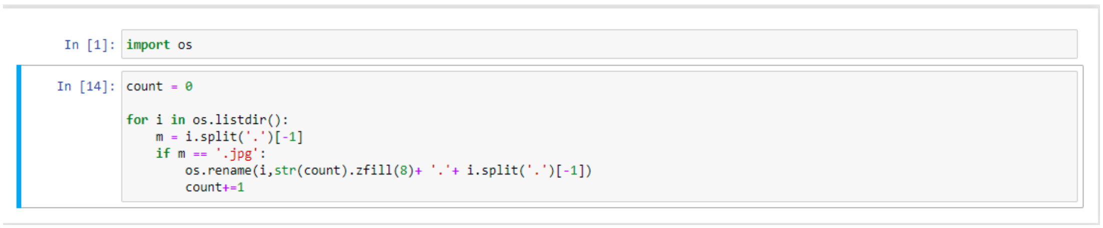


Figure 4.1. Steps of Object Detection System (ODS)

4.1. Preparing Dataset

The object detection system for an object or a group of objects at one time is one of the most important and focused matters in our study. This requires collecting a sufficient amount of images about these objects, which are the main element in our research, in cooperation with one of the FMCG companies. A group of photos are taken from seven of the company's products for the purpose of conducting the necessary research on them. Live photos were taken from the sales outlets in cooperation with the sales team. 158 images were collected. The goal here was to focus on making the dataset somewhat small to study the future challenges of other products. The diversity of images and angles of capture was adopted in order for the data collection to be somewhat rich with different angles of the products. After collecting the data, we need to change the size of the images to the size that suits the type of input to the algorithm required to train the data, and we have different sizes of images for a group. The data in the case of using the YOLO V3 algorithm, where the image size is (448 × 448), then the size of the images in the input data set for the Faster-RCNN algorithm, where the image size was changed to (640 × 640). After the resizing process, the data was collected in one folder for the annotation and labeling of the objects on which the model will be trained, and we should not forget to rename the images with and sequenced name. Figure 4.2. shown the Python code written to rename the images in our dataset.



```
In [1]: import os
```

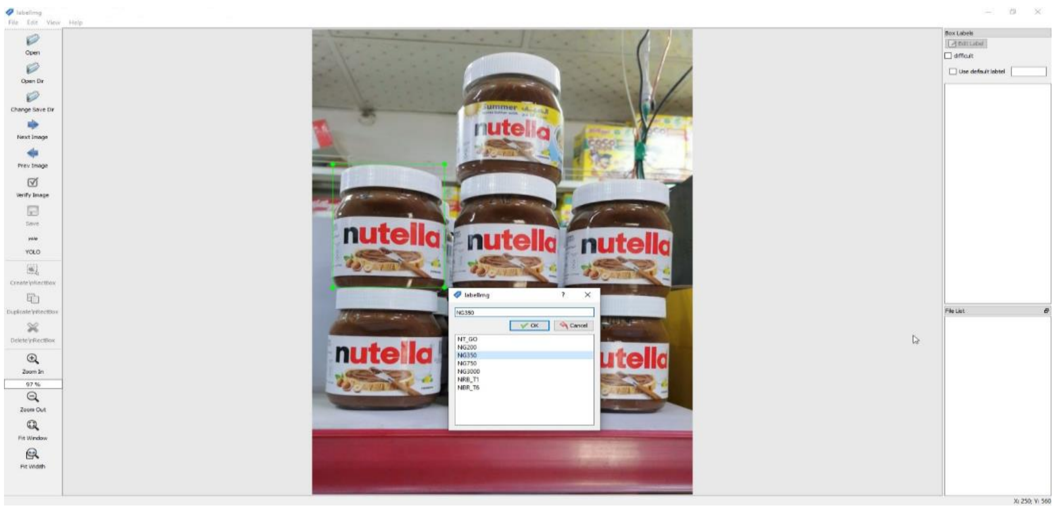
```
In [14]: count = 0

         for i in os.listdir():
             m = i.split('.')[-1]
             if m == '.jpg':
                 os.rename(i,str(count).zfill(8)+ '.'+ i.split('.')[-1])
                 count+=1
```

Figure 4.2. Python Code to rename images using Jupyter notebook

4.1.1. Labeling objects in the images

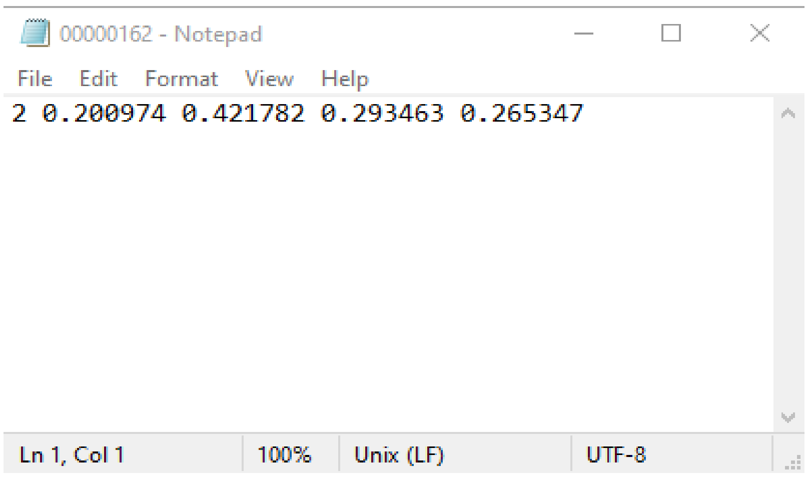
After collecting the images in a unified folder and preparing the data set that we will work on, the essential part begins, which is labeling objects within the images. We used two methods for this process, and both of them give the same results, with one of them superior to the other. As we will mention later, we used the LblImage application, which we explained in the previous chapters, whereby this application draws a box around the object and names it so that we eventually have a TXT file. This file contains the numbers that represent the number of the group to which the object belongs, with the coordinates of the location of the object in the image. Finally these values will be used to train our model. This application was used to initialize the data set of the YOLOv3 algorithm. On the other hand, the Roboflow framework was used to label the data of the Faster-RCNN algorithm. Where this environment helps to produce a variety of outputs according to the algorithm required to work on it, the Roboflow framework is preferred in order to create centralization of data. Figure 4.3. shows us how to labeling the objects using LblImage. Figure 4.4. shows the result obtained from labeling the objects inside the image, which is the dataset on which the model is trained. Figure 4.5. shows how to label the objects using the Roboflow framework. Figure 4.6. shows us the obtained result as the main data file for training the model using the Roboflow framework.



NG350

NT_GO
NG200
NG350
NG750
NG3000
NRB_T1
NBR_T6

Figure 4.3. Labeling objects using LblImage application



```
2 0.200974 0.421782 0.293463 0.265347
```

Figure 4.4. LblImage result

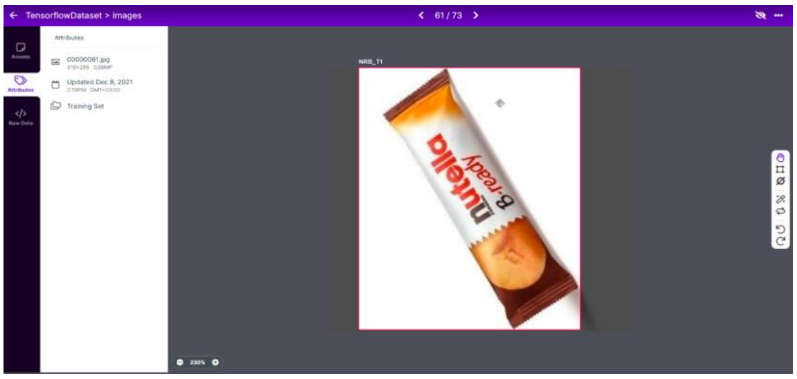
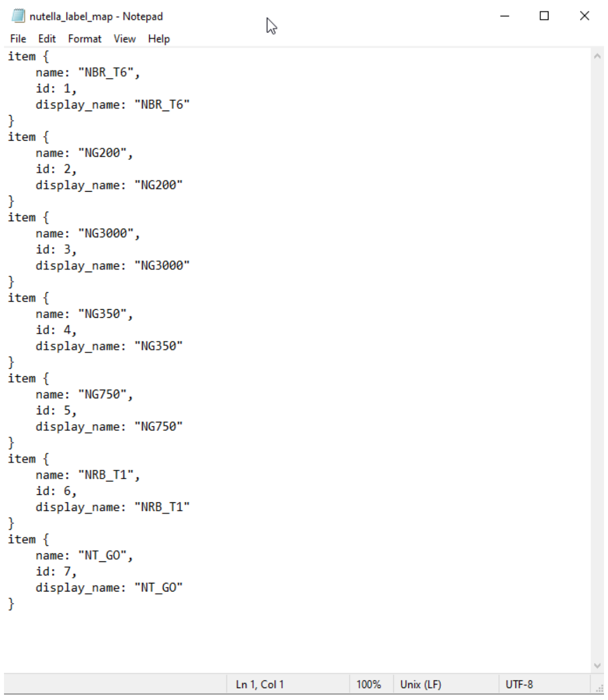


Figure 4.5. Using RoboFlow Framework to labeling objects

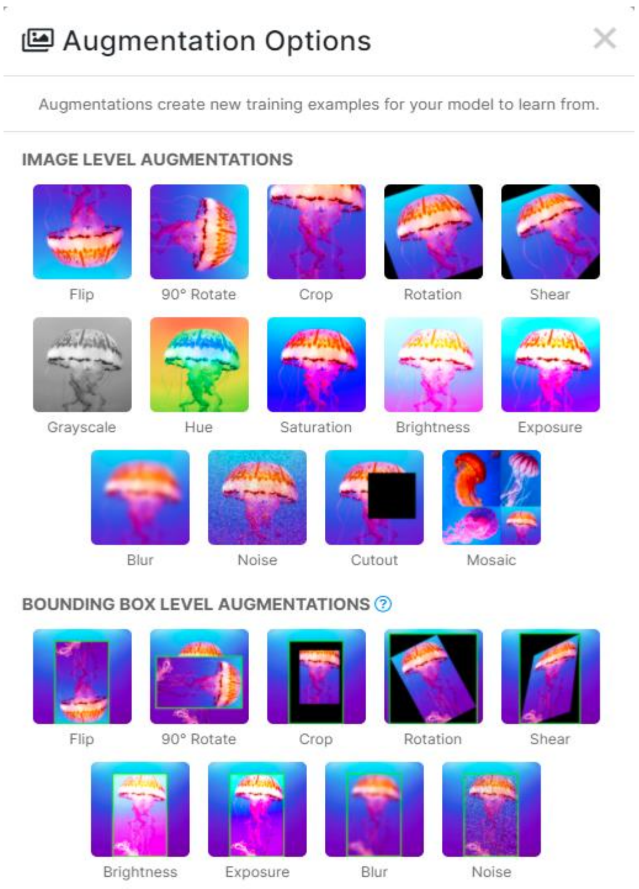


```
item {
    name: "NBR_T6",
    id: 1,
    display_name: "NBR_T6"
}
item {
    name: "NG200",
    id: 2,
    display_name: "NG200"
}
item {
    name: "NG3000",
    id: 3,
    display_name: "NG3000"
}
item {
    name: "NG350",
    id: 4,
    display_name: "NG350"
}
item {
    name: "NG750",
    id: 5,
    display_name: "NG750"
}
item {
    name: "NRB_T1",
    id: 6,
    display_name: "NRB_T1"
}
item {
    name: "NT_GO",
    id: 7,
    display_name: "NT_GO"
}
```

Figure 4.6. RoboFlow Labeling Result

4.1.2. Dataset preprocessing

Preprocessing is the step that precedes the process of starting to train the model on the data set that was collected. As mentioned earlier in the preparing dataset section because one of the preprocessing steps is the resize process, we modified the size of the images in proportion to the type of algorithms through which the data will be trained. One of the crucial steps in the data preprocessing process is data augmentation, where this step increases the volume of data in the data set using rotation, scaling, shear, etc. techniques. Figure 4.7. shows the types of data augmentation. As we indicated at the beginning of this chapter, we need this step that we rely on a small data set. In such cases, data augmentation is the best solution to increase the data to obtain better prediction results.


Augmentation Options

Augmentations create new training examples for your model to learn from.

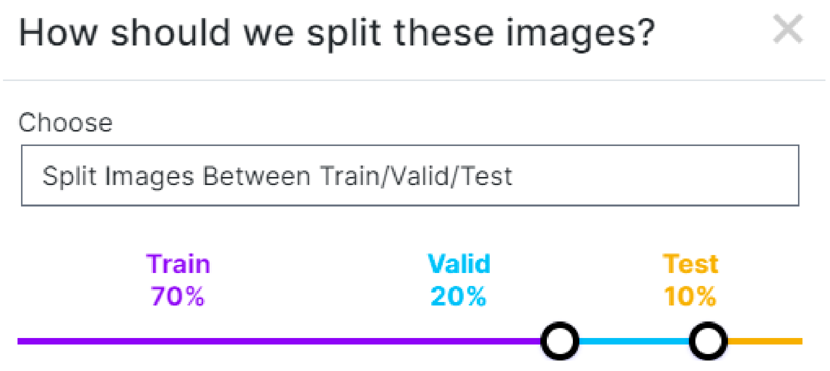
IMAGE LEVEL AUGMENTATIONS

BOUNDING BOX LEVEL AUGMENTATIONS

Figure 4.7. Using RoboFlow Data Augmentation

4.1.3. Training and testing the models

Before starting the training process, we must divide the dataset into parts, a part for training the model, another part for checking the model's efficiency, then the last part for testing the model. Figure 4.8. shows the splitting of the dataset using the RoboFlow framework. We used the split 70% training and 20% validation, and 10% for test.


How should we split these images?

Choose

Split Images Between Train/Valid/Test

Train 70%

Valid 20%

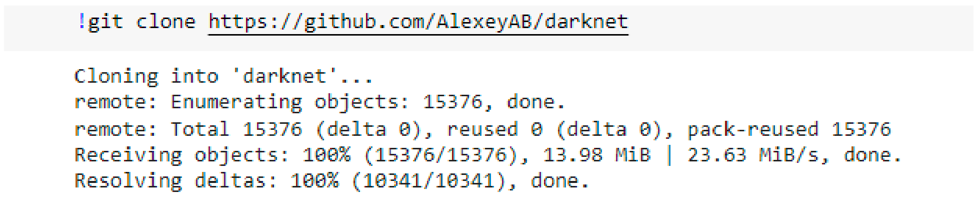
Test 10%

Figure 4.8. Using RoboFlow to split the Dataset

In the coming paragraphs, we will explain the process of training each model separately and how the results of each model were tested, with YOLOv3 or Faster-RCNN.

4.1.3.1. Using darknet pre-trained model

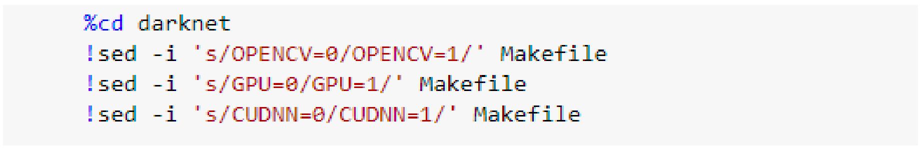
In this step, we use google collaboratory (Google Colab) as a platform to prepare the Darknet model to train our dataset on it. The benefit of using Google Colab is to create a virtual machine (VM) on the cloud, which enables us to use high GPU and RAM on Google cloud. In the beginning, we need to install the required environment on the virtual machine. First, we clone the darknet install file from the official repository on GitHub, as shown in Figure 4.9.


```
!git clone https://github.com/AlexeyAB/darknet

Cloning into 'darknet'...
remote: Enumerating objects: 15376, done.
remote: Total 15376 (delta 0), reused 0 (delta 0), pack-reused 15376
Receiving objects: 100% (15376/15376), 13.98 MiB | 23.63 MiB/s, done.
Resolving deltas: 100% (10341/10341), done.
```

Figure 4.9. Cloning Darknet model into Google VM

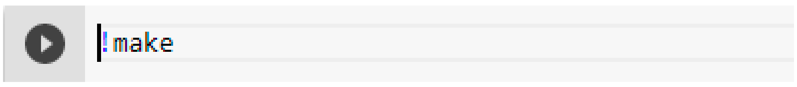
This file contains the main file responsible for the global parameter of training the Darknet model. The name of the file is MakeFile. In this file, we need to enable OpenCV library, and also, we need to enable GPU, which is very important to use when our training dataset contains images (see Figure 4.10.).



```
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
```

Figure 4.10. Change MakeFile to have GPU and OpenCV enabled

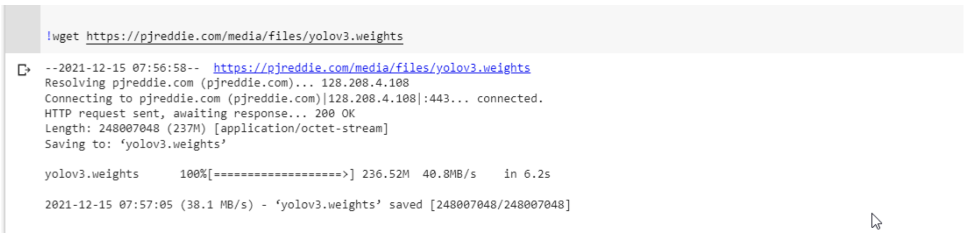
After this, we need to start to install the Darknet environment on the Google VM using the command (!make), which is to begin building the Darknet as per the parameters that we mentioned in the MakeFile (see Figure 4.11.).



```
!make
```

Figure 4.11. Command to build Darknet

Now, this is the fundamental step to start using YOLOv3. The YOLOv3 pre-trained weights should be downloaded from the official website, as shown in Figure 4.12.



```
!wget https://pjreddie.com/media/files/yolov3.weights

--2021-12-15 07:56:58--  https://pjreddie.com/media/files/yolov3.weights
Resolving pjreddie.com (pjreddie.com)... 128.208.4.108
Connecting to pjreddie.com (pjreddie.com)|128.208.4.108|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 248007048 (237M) [application/octet-stream]
Saving to: ‘yolov3.weights’

yolov3.weights      100%[===================>] 236.52M  40.8MB/s    in 6.2s

2021-12-15 07:57:05 (38.1 MB/s) - ‘yolov3.weights’ saved [248007048/248007048]
```

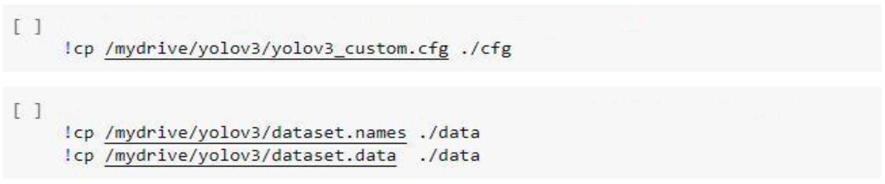
Figure 4.12. Download YOLOv3 pre-trained weights

Also, we need to download the pre-trained convolutional layer weights(darknet53.conv.74) using the command line (!wget http://pjreddie.com/media/files/darknet53.conv.74). This represents the whole model that we will train our dataset using the pre-trained YOLOv3 weights to avoid building the weights from scratch. Now we are going to set the crucial parameters we need when we use YOLOv3. We need to open the file yolov3.cfg, and we will update these parameters as shown in Table 4.1.

Table 4.1. YOLO parameters

Parameter Name	Value	Description
Batch	64	The iteration requiring a certain number of photos and label files.
Subdivisions	16	Number of block /batches.
Classes	7	Number of classes we use when we train and test the model. In our case, we have seven classes
Max_batches	14000	This number coming from the equation (2000* No. of classes)
Filters	36	filters = (classes + 5) * 3 = (classes + width + height + x + y + confidence) * mask = (classes + 1+1+1+1+1) * mask = (classes + 5) * mask

Now we are ready to copy the prepared dataset and the configuration files from Google drive to the Google Colab VM to start training the model, as shown in Figure 4.13.

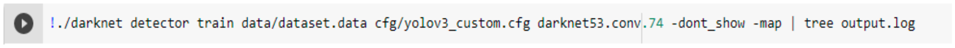


```
[ ]
    !cp /mydrive/yolov3/yolov3_custom.cfg ./cfg
```

```
[ ]
    !cp /mydrive/yolov3/dataset.names ./data
    !cp /mydrive/yolov3/dataset.data  ./data
```

Figure 4.13. Copy the requried file to VM for training the model

After copying the required file, we start to train the model. This step takes time according to the number of images that the dataset contains. We will use the command line as shown in Figure 4.14. to train the model on our dataset.



```
!./darknet detector train data/dataset.data cfg/yolov3_custom.cfg darknet53.conv.74 -dont_show -map | tree output.log
```

Figure 4.14. start to train the model using YOLOv3 weights

With 73.49% average IoU and 62.43% mAP, as shown in Figure 4.15., our model has finished the training step.

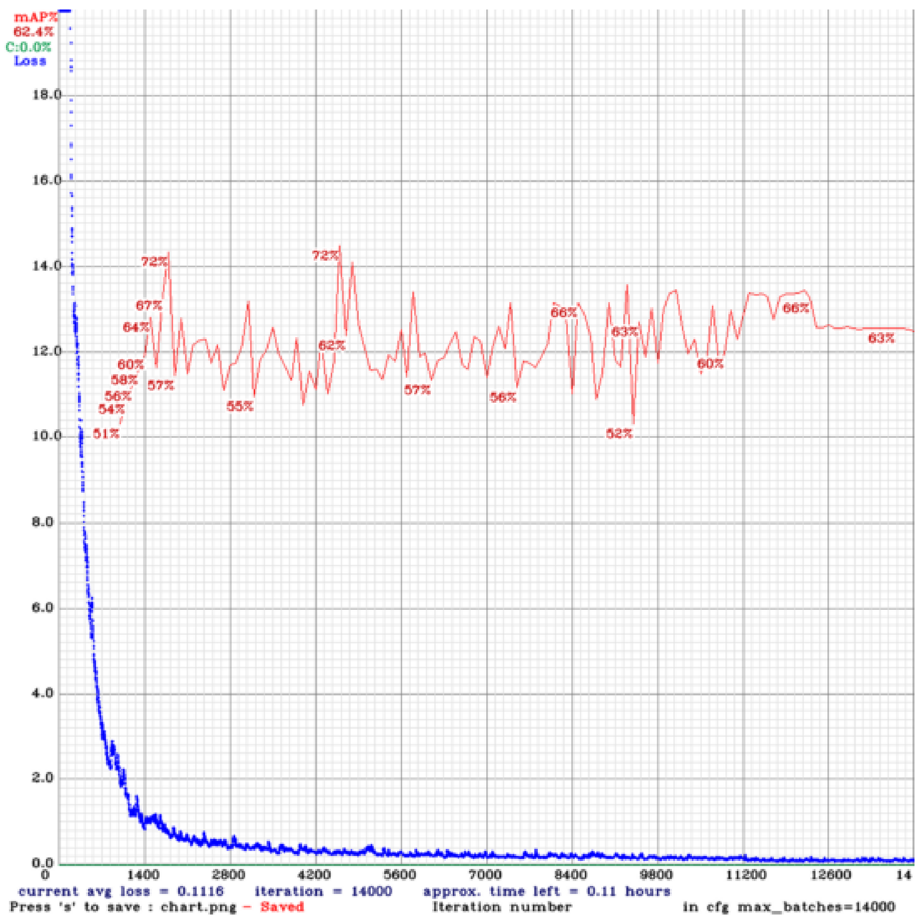
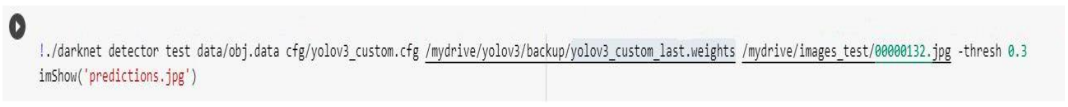


Figure 4.15. Model loss function and mAP result

Now we can start using our resulting weights that the Darknet saved on the Google drive folder with the name (yolov3_custom_last.weights) after finishing the training step. This weights file represents the weights file that contains our seven classes for prediction, and now we can use the weights to start to make protection to detect the object from an image our dataset doesn't contain. Figure 4.16. shows the command line we need to make a prediction.



```
!./darknet detector test data/obj.data cfg/yolov3_custom.cfg /mydrive/yolov3/backup/yolov3_custom_last.weights /mydrive/images_test/00000132.jpg -thresh 0.3
imShow('predictions.jpg')
```

Figure 4.16. Darknet prediction command line

After applying the prediction command line, we will get the result as outbound boxes around the predicted objects with an Annotation on each box representing the class name that the object belongs to (see Figure 4.17.).

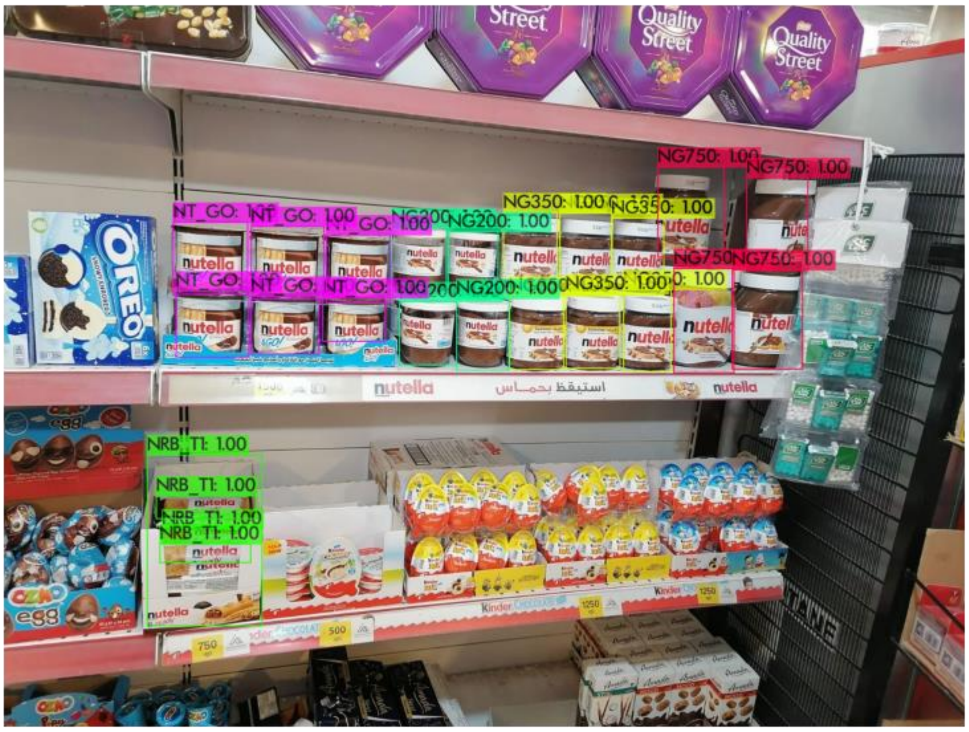
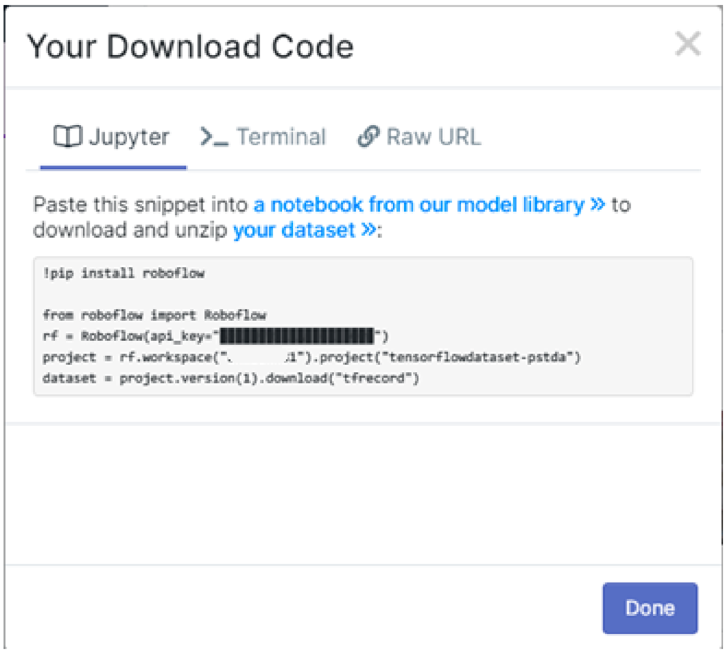


Figure 4.17. Prediction result using YOLOv3

4.1.3.2. Using TF2 model zoo pre-trained model

Because of, YOLO suffers from a lack of object detecting capabilities. As there are only two anchor boxes for each kind of object in the grid, YOLO struggles to distinguish between small and close to each other objects [45]. Therefore, we need to use Faster-RCNN to solve this issue and to improve and enhance the prediction results for this kind of scenarios. As well as we did before in Darknet model we need to prepare the configuration file to be able to train our dataset on the Faster-RCNN model that we select from the TF2 Model Zoo framework. We need to export the dataset from RoboFlow framework as a TFRcord file. This file contains our dataset that preprocessed and prepared to be compatible with TF2 models by RoboFlow framework as appeared in Figure 4.18.



Paste this snippet into a notebook from our model library » to download and unzip your dataset »:

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="[redacted]")
project = rf.workspace(".[redacted]1").project("tensorflowdataset-pstda")
dataset = project.version(1).download("tfrecord")
```

Figure 4.18. RoboFlow Dataset Download Script

We will use this code to download our dataset to Google drive to start training the model. Now we need to download the pre-trained model (faster_rcnn_resnet50_v1_640x640) from the official website (https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md), then we need to prepare the configuration as shown in Table 4.2.

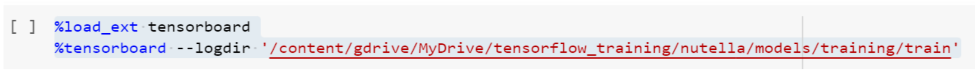
Table 4.2. TF2 Model Parameters

Parameter Name	Value	Description
train_input_reader_label_path	/content/gdrive/MyDrive/ TensorflowDataset/train/nutella_label_map.pbtxt	The path of group label name of the train dataset
tf_record_input_reader	/content/gdrive/MyDrive/ TensorflowDataset/train/nutella.tfrecord	The path of train dataset
metrics_set	coco_detection_metrics	The metrics we get after completing training the model
eval_input_reader	/content/gdrive/MyDrive/	The evaluation dataset lable

Table 4.2. (Continued)

	TensorflowDataset/train/nutella_label_map.pbtxt	path
tf_record_input_reader_eval	/content/gdrive/MyDrive/TensorflowDataset/valid/nutella.tfrecord	The evaluation dataset path to evaluate our model loss function

Using Google Colab, we start the training step after completing the model training using (tensorborad library) as shown in Figure 4.19.



```
[ ]  %load_ext tensorboard
     %tensorboard --logdir '/content/gdrive/MyDrive/tensorflow_training/nutella/models/training/train'
```

Figure 4.19. TensorBoard library command line

We get the loss function results that shows that the loss function percentage is bigger than the model that we get in YOLOv3 Darknet that cause weakness in our model to detect the object in all circumstances, as shown in Figure 4.20.

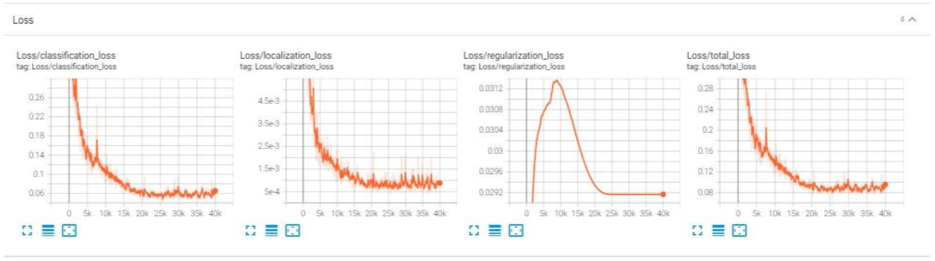
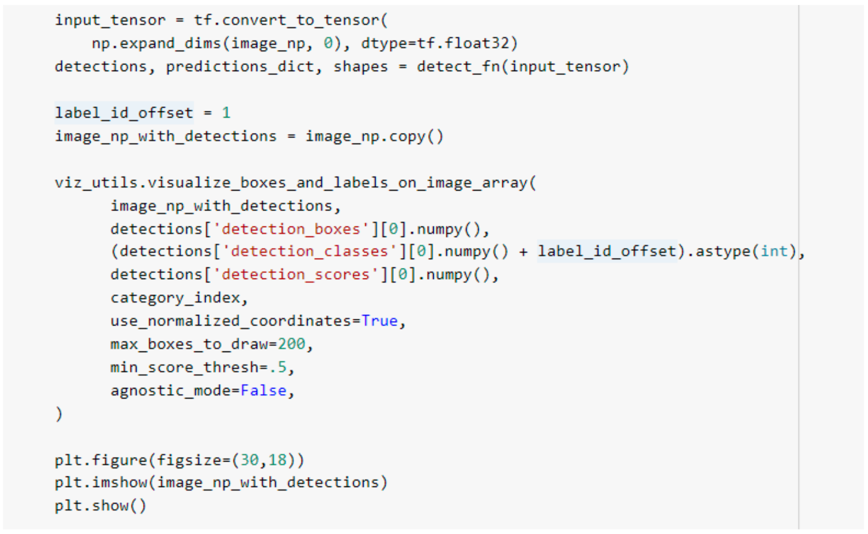


Figure 4.20. TF2 Result Graph

Then we used our model to predict by using the script as shown in Figure 4.21.



```
input_tensor = tf.convert_to_tensor(
    np.expand_dims(image_np, 0), dtype=tf.float32)
detections, predictions_dict, shapes = detect_fn(input_tensor)

label_id_offset = 1
image_np_with_detections = image_np.copy()

viz_utils.visualize_boxes_and_labels_on_image_array(
      image_np_with_detections,
      detections['detection_boxes'][0].numpy(),
      (detections['detection_classes'][0].numpy() + label_id_offset).astype(int),
      detections['detection_scores'][0].numpy(),
      category_index,
      use_normalized_coordinates=True,
      max_boxes_to_draw=200,
      min_score_thresh=.5,
      agnostic_mode=False,
)

plt.figure(figsize=(30,18))
plt.imshow(image_np_with_detections)
plt.show()
```

Figure 4.21. Object Detection Script for TF2 Model

Then the result will be bound boxes drawn on the objects with the group annotation on each box, as shown in Figure 4.22.

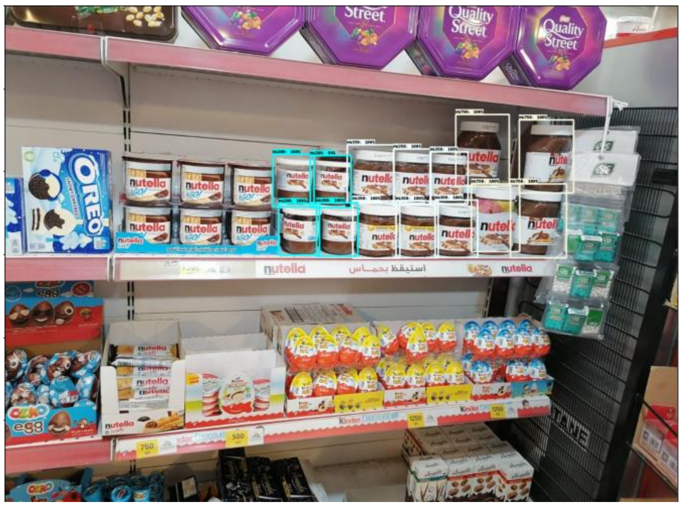


Figure 4.22. Prediction Result Using Faster-RCNN

4.1.4. Use the Resulting Model to Predict Live Video Objects

In this section, we will try to test our model to detect the objects in real-time. We used the YOLOv3 model for this task, as mentioned before that the YOLOv3 is very fast in detecting the objects in real-time because it generates classification and bounding box regression at the same time. As a result, we get a good result in detecting objects in a live camera. Figure 4.23. shows the real-time object detection using the YOLOv3 model.

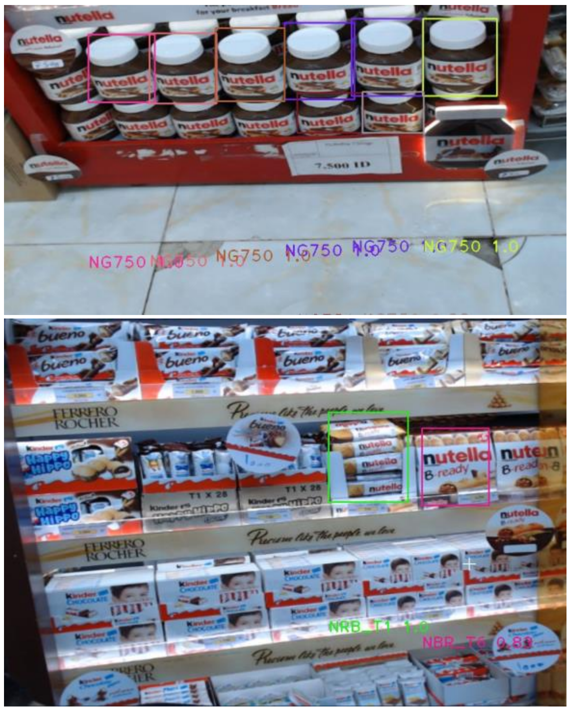


Figure 4.23. Using YOLO V3 Model to Detect

4.1.5. Darknet and TF2 model zoo performance comparison

Table 4.3. YOLOv3 Results vs Faster-RCNN Results

Criteria	YOLOv3	Faster-RCNN
Loss Function	0.1116	0.10392012
Mean Avarage Percent (mAP)	62.40%	33.60%
Detect all products at the same time	Yes	No
Speed detecting objects per image	17 milli seconds	15.8 seconds
Dataset Size	Good with small dataset	Need more data

Figure 4.24. shows the superiority of the YOLO model over the Faster-RCNN in detecting objects, while Faster-RCNN failed to predict all the products and was only able to predict two of the seven groups it had previously trained on, YOLOv3 model demonstrated that it was able to predict all the products that it had previously trained on with accuracy of up to 99%.

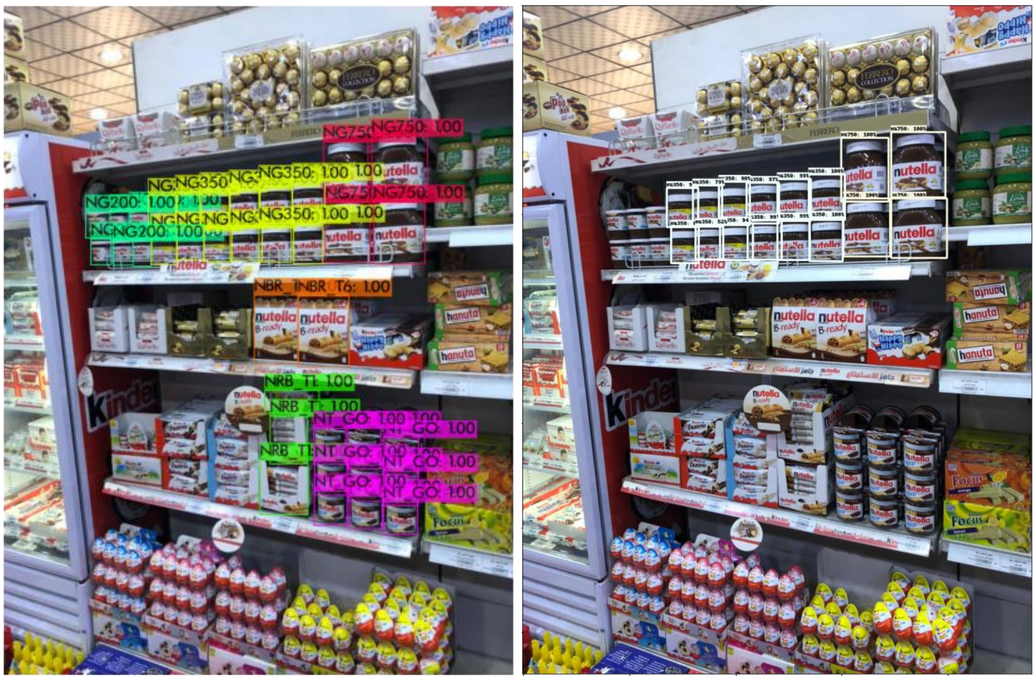


Figure 4.24. Yolo vs Faster-RCNN in Detecting Objects

As we mentioned before, the YOLO algorithm suffers from detecting the objects close to each other. We test this point as shown in Figure 4.25. and Figure 4.26., which shows the superiority of Faster-RCNN to recognize the objects that are adjacent to each other.

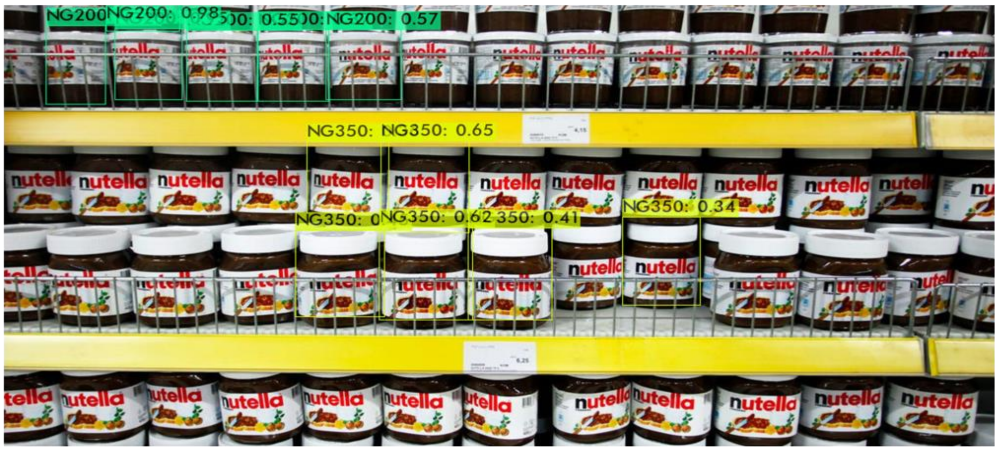


Figure 4.25. Yolo detecting objects

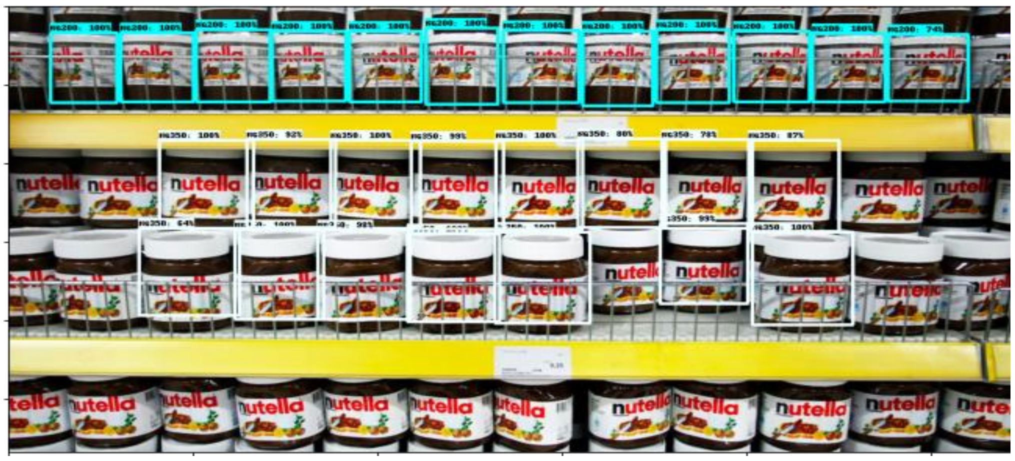


Figure 4.26. Faster-RCNN Detecting objects

Finally, the YOLOv3 is very suitable for our case study because the FMCG Company's goal was to get the products visibility in the outlets and our values with YOLO model show good results to fulfill this goal. Still, in case the FMCG Company needs to apply the Retail planogram, in this case, Faster-RCNN will be better to choose because it solves the problem of the contiguous object.

4.1.6. Using Google drive API

Google Drive API is a portal to access our resources on the Google Drive storage. It is a collection of methods to make users able to do the operations (Insert, Delete, and Update) on Google Drive. We used this API to read the results of the prediction of our models to send the result to a local database by using Python scripts. Figure 4.27. shows how Google Drive API works.

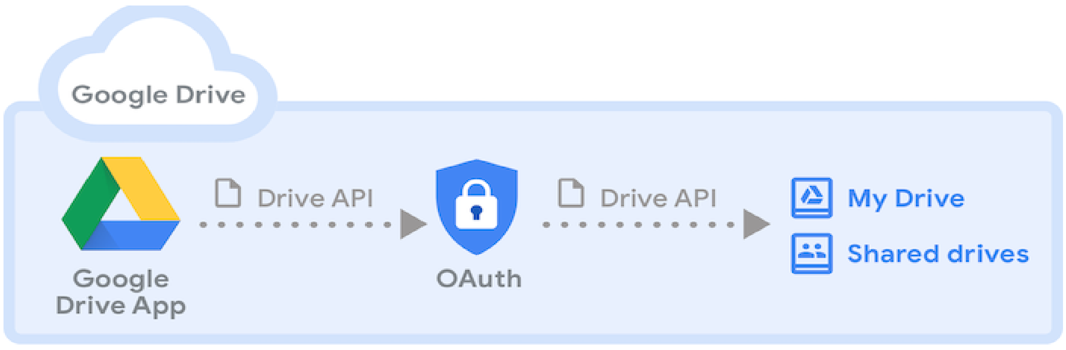
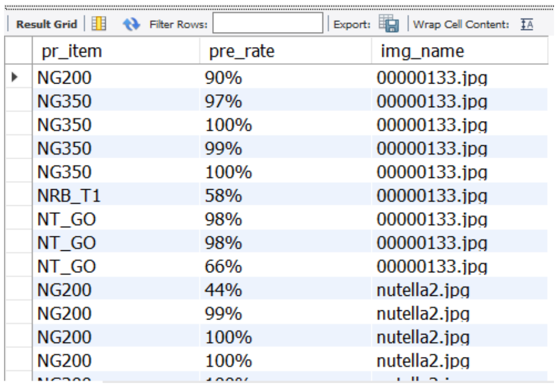


Figure 4.27. How Google Drive API is works

4.1.7. Storing prediction results into MySQL database

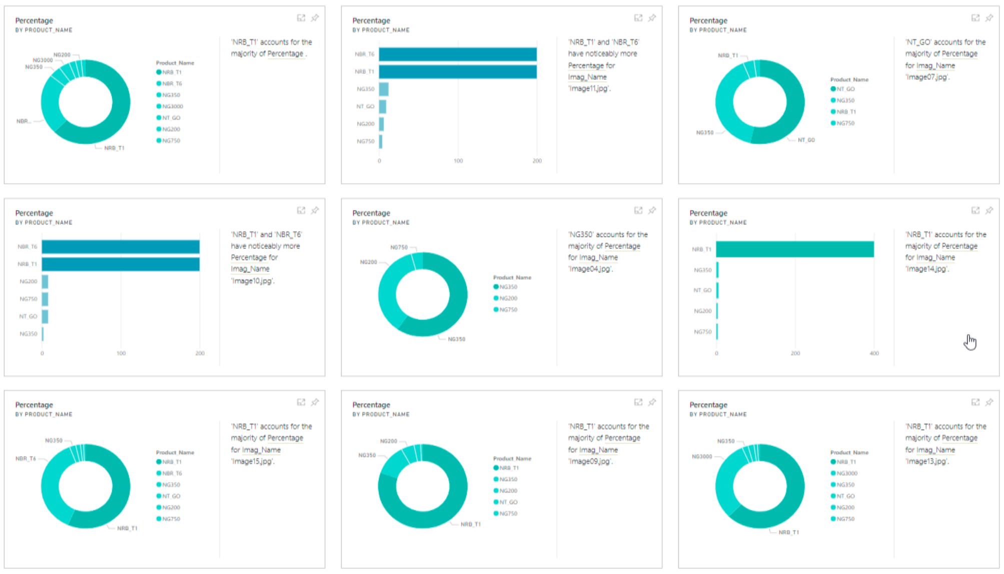
After we used Google Drive API, we got the prediction result for detecting the objects from the images, and we translated these values to be organized values in a MySQL table. MySQL is an open-source database. Figure 4.28. shows how the prediction results in a MySQL Table.



pr_item	pre_rate	img_name
NG200	90%	00000133.jpg
NG350	97%	00000133.jpg
NG350	100%	00000133.jpg
NG350	99%	00000133.jpg
NG350	100%	00000133.jpg
NRB_T1	58%	00000133.jpg
NT_GO	98%	00000133.jpg
NT_GO	98%	00000133.jpg
NT_GO	66%	00000133.jpg
NG200	44%	nutella2.jpg
NG200	99%	nutella2.jpg
NG200	100%	nutella2.jpg
NG200	100%	nutella2.jpg

Figure 4.28. Prediction result into MySQL Table

In the next step, we had to make this data readable by sales managers by using presentation and analysis tools such as business intelligence tools, which enable the user to display the information clearly and smoothly, as well as help to give suggestions for decision-making. We used the Microsoft Power BI tool, and as shown in Figure 4.29., we made a dashboard that helps the sales manager to read which the data that we get from applying object detection algorithms on the images.


Percentage
BY PRODUCT_NAME
'NRB_T1' accounts for the majority of Percentage .

Percentage
BY PRODUCT_NAME
'NRB_T1' and 'NBR_T6' have noticeably more Percentage for Imag_Name 'Image11.jpg'.

Percentage
BY PRODUCT_NAME
'NT_GO' accounts for the majority of Percentage for Imag_Name 'Image07.jpg'.

Percentage
BY PRODUCT_NAME
'NRB_T1' and 'NBR_T6' have noticeably more Percentage for Imag_Name 'Image10.jpg'.

Percentage
BY PRODUCT_NAME
'NG350' accounts for the majority of Percentage for Imag_Name 'Image04.jpg'.

Percentage
BY PRODUCT_NAME
'NRB_T1' accounts for the majority of Percentage for Imag_Name 'Image14.jpg'.

Percentage
BY PRODUCT_NAME
'NRB_T1' accounts for the majority of Percentage for Imag_Name 'Image15.jpg'.

Percentage
BY PRODUCT_NAME
'NRB_T1' accounts for the majority of Percentage for Imag_Name 'Image09.jpg'.

Percentage
BY PRODUCT_NAME
'NRB_T1' accounts for the majority of Percentage for Imag_Name 'Image13.jpg'.

Figure 4.29. Microsoft Power BI Tool Dashboard

CHAPTER 5. RESULTS AND DISCUSSION

In this chapter, the results obtained through our study will be presented, and those results will be discussed.

5.1. Results

As we mentioned earlier that the goal of our study focuses on reducing expenses (costs) and working to increase the efficiency of the performance of the sales representative. We studied the comparison between the results that obtained from use of deep learning techniques with the current results of relying on the human element to manage the task entrusted to him. The following is a measure of comparison between the performance of deep learning algorithms in managing expenses compared to the use of the human factor. These expenses are

1- Salaries,

2- Computers,

3- Mobile devices,

4- Offices, and

5- Annual Cloud fees.

We found that deep learning algorithms reduced the annual costs to (99%) and the total Expense by (99.99%) as shown in Figure 5.1.

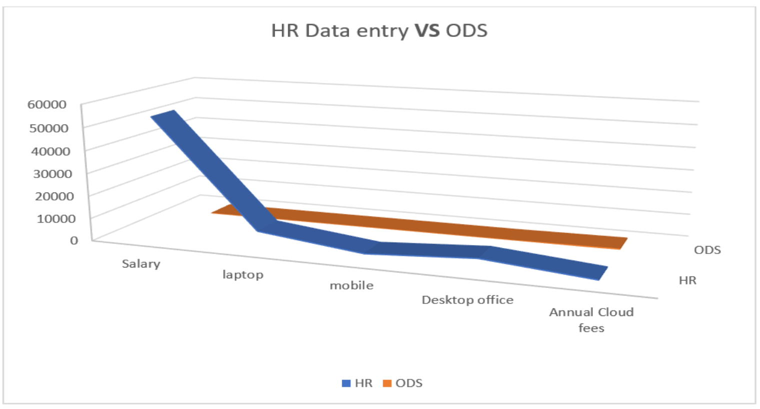
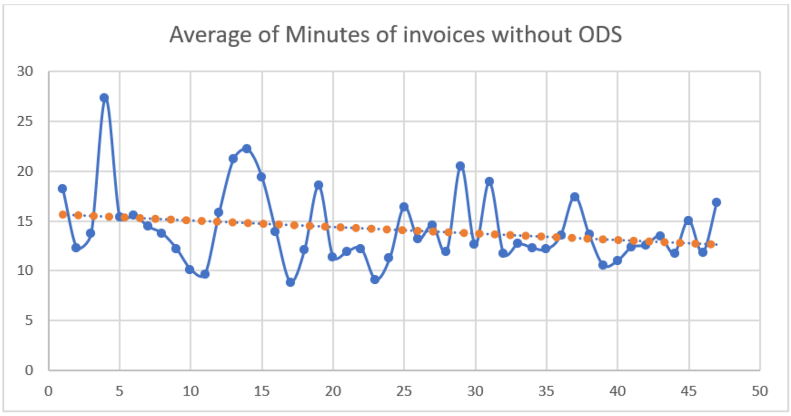
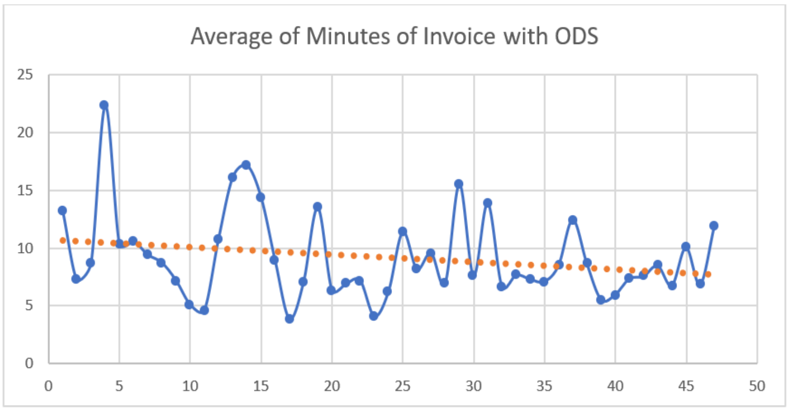


Figure 5.1. Human Resource VS ODS Expenses Comparison

On the other hand, the scale (time) has been adopted as a critical factor to improve the performance of the salesmen, as time is a crucial factor in the performance of daily tasks. This provide adding new jobs by the sales department to the delegates a drain on the delegate’s time incorrectly if the representative is not supported with the necessary tools. Accordingly, we conducted a field study with the delegates to follow up on the part of collecting images from the ports, arranging them, and sending them to the cloud storage so that the data would be ready for analysis by the human element in the back office. Specific samples were taken from the delegates’ data in different geographical locations. The time spent by the sales representative in organizing and preparing the data to be uploaded to the cloud storage ranges from 7 to 10 minutes. Therefore, the total time for making sales invoices will range from 8 to 27 minutes, as shown in Figure 5.2.



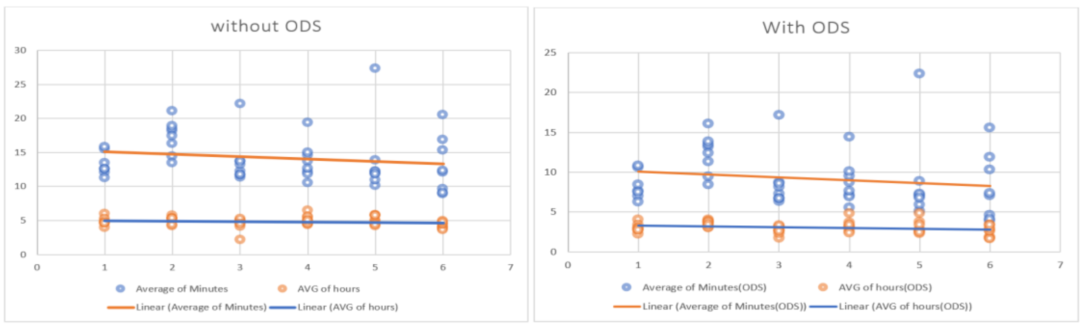
(a)



(b)

Figure 5.2. Average of minutes of invoices before and after ODS

After applying the (ODS) system, we found that the rate of reducing the time for a single bill range between 4.5 to 5 minutes, which is 35% of the rate of one invoice, which represents 14.18 minutes. Therefore, the time required for the representative to visit the customer will range between 3.8 to 22.31 minutes, as shown in Figure 5.2., this work led us to provide a daily average of time for the sales representative by 1 hour and 38 minutes, at a rate of (28%) of the daily time spent in the daily representative's route. Figure 5.3. shows the difference in time before and after using the ODS system.



(a) (b)

Figure 5.3. Average of hours of invoices before and after ODS

The percentage 28% means to decrease the average time that the sales representative spent on each invoice to 9.18 minutes, and that will give the sales representative to add 5 new customers to his daily route to with percentage 21% as an increasing percentage then this will increase the revenue of sales, thus accelerating the achievement of the monthly sales target.

5.2. Discussion

Through our study, we highlighted how to remove the additional tasks on the daily tasks of sales representatives and transform the process of following up the items on the shelves in the outlets into an automated procedure. Deep learning techniques perform these tasks instead of the sales representatives themselves. In previous studies, as in [1] and [3], the representatives themselves were involved in the visibility of goods in stores divided with the help of artificial intelligence techniques, which did not fully contribute to the participation of the entire sales team, including the sales management, in evaluating the visibility of the products in the market. Drive through the artificial intelligence techniques present on the company's server, will transfer the data resulting from the prediction of artificial intelligence techniques to the database on the company's local server. This transfer will give a complete reading of the sales management such as supervisors and sales managers for the visibility of items in the market instead of using mobile applications whose results are visible only to sales representatives. One of the most important points addressed by this study is to improve the efficiency of the sales representative by removing any task to verify the visibility of the products by the sales representative himself and only that the images are sent to the cloud by the salesman within a period of no more than seconds. YOLO algorithms have achieved the goal of deep learning, where the salesman can send videos to be converted into readable data within the database from. A 99% success rate was achieved as a product protection rate whether in pictures or videos, which resulted in the dispensation of any human cadre in the company's work branches to record the data extracted from the images. This led to a reduction in monthly and annual expenses to (99.99%). Comparing the results after applying the ODS system with the human resource team (HR), we obtained this percentage and as shown in Figure 5.1., and therefore when the tasks were canceled from the daily tasks of the sales representative, the time for each representative was reduced from the required visit time to any sales outlet from an average of 5 minutes less from the visit without using the ODS system. As Figure 5.2. shows the time average of invoices issued by the sales representatives during an entire month, where the average billing time ranged between 8 to 27 minutes, with a rate of 14.8 as an average billing time one after using the ODS system. The billing time was reduced to a period ranging from 3.8 to 22.31 with an average of 9.18 minutes for the invoice. Let us clarify this change in time according to hours also in Figures 5.3., which shows the total invoices in weeks during one month issued by sales representatives, where the entire time in hours that were provided to the sales representative is 1 hour and 38 minutes per day from the total visits that he makes by the delegate on a daily basis. This led to an increase in coverage for five new agents for each sale representative.

CHAPTER 6. CONCLUSION AND FUTURE WORKS

Costs and expenses in commercial companies are the main factors for managing the company’s financial resources and choosing plans to expand the work of companies. FMCG companies are considered among the most affected by the factors of leakage in costs and expenses for projects that occur on the work plan without choosing the right tools. These lead to the influencing the performance of the work of cadres by increasing the working time or increasing the tasks assigned to the work team at the same time, which generates great pressure on the employees. The thesis adopted a solution to this problem using deep learning techniques. The study deals with a general explanation of the convolution algorithms within the scope of deep learning and an accurate description of the object detection algorithms using the YOLOv3 and Faster-RCNN algorithms to replace the human element with these algorithms to identify products within the images and thus convert them into organized, analytical values. Readable by the sales team, this thesis contributed to choosing the appropriate algorithm to solve a problem. Products visibility and invest the efforts of Deep learning algorithms in executing more than one task at the same time may, in turn, need significant human elements to implement these tasks. The YOLOv3 algorithm was proposed to solve the problem of the visibility of products where Faster-RCNN algorithm was proposed to solve the shelf arrangement KPIs and planogram problem. A comparison was made between algorithms of YOLOv3 and Faster-RCNN. It was found that the YOLOv3 algorithm suffers from a weak problem of convergent object detection, while Faster-RCNN has made progress in solving this problem. While the YOLOv3 algorithm excelled in being satisfied with a small dataset and a faster time in discovering things, Faster-RCNN algorithm is lacked from these features. The data was converted to digital data stored in an MYSQL database by using the Google Drive API cloud technologies, and therefore the results were displayed using the business intelligence tool to design a dashboard. The Microsoft Power BI tool was used to achieve this goal.

As a future study, we plan to solve the problem of discovering competing products on the rented shelves at outlets and also to measure the space of shelves to match the terms of contracts between food distribution companies (FMCG) with outlets using deep learning techniques.

REFERENCES

[1] Sykes, Edward R. "A deep learning computer vision iPad application for Sales Rep optimization in the field." The Visual Computer: 1-20, 2021.

[2] Tarai, Rahul Karan. "The Implications of Ai In Enhancing Fmcg Industries." PalArch's Journal of Archaeology of Egypt/Egyptology 17.9: 6505-6522, 2020.

[3] Picareta, Gilberto, Eugenie Weissheim, and Martin Klöhn. "Intelligent Applications in the Modern Sales Organization." The Machine Age of Customer Insight. Emerald Publishing Limited, 2021.

[4] A. Nichol, J. Batten, M. Halley, J. Axelrod, P. Sankar, M. Cho. "A Typology of Existing Machine Learning–Based Predictive Analytic Tools Focused on Reducing Costs and Improving Quality in Health Care: Systematic Search and Content Analysis." Journal of medical Internet research 23.6, 2021.

[5] Ilan, Yaron. "Improving global healthcare and reducing costs using second-generation artificial intelligence-based digital pills: a market disruptor." International Journal of Environmental Research and Public Health 18.2: 811, 2021.

[6] Golding, Lauren Parks, and Gregory N. Nicola. "A business case for artificial intelligence tools: the currency of improved quality and reduced cost." Journal of the American College of Radiology 16.9: 1357-1361, 2019.

[7] A. Sahoo, CH. Pradhan, R. Barik, H. Dubey. "DeepReco: deep learning-based health recommender system using collaborative filtering." Computation 7.2: 25, 2019.

[8] Sharma, Neerav, and Rahul Dev Garg. "Cost reduction for advanced driver assistance systems through hardware downscaling and deep learning." Systems Engineering, 25: 105-188, 2021.

[9] Dinler, Ali. "Reducing balancing cost of a wind power plant by deep learning in market data: A case study for Turkey." Applied Energy 289, 2021.

[10] S. Birim, I. Kazancoglu, S. Mangla, A. Kahraman, Y. Kazancoglu. "The derived demand for advertising expenses and implications on sustainability: a comparative study using deep learning and traditional machine learning methods." Annals of Operations Research, 1-31, 2022.

[11] Chintagunta, Pradeep K., and Harikesh S. Nair. "Structural workshop paper—discrete-choice models of consumer demand in marketing." Marketing Science 30.6: 977-996, 2011.

[12] X. Du, Y. Cai, S. Wang, L. Zhang. "Overview of deep learning." 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC). IEEE, 2016.

[13] M. Alizadeh, J. Fernández-Marqués, N. Lane, Y. Gal. "An empirical study of binary neural networks' optimisation." International Conference on Learning Representations, 2018.

[14] Kim, Phil. "Convolutional neural network." MATLAB deep learning. Apress, Berkeley, CA, 121-147, 2017.

[15] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." 2017 International Conference on Engineering and Technology (ICET). Ieee, 2017.

[16] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." International conference on machine learning. PMLR, 2015.

[17] Arc. “Convolutional Neural Network. In This Article, We Will See What Are… by Arc Towards Data Science.” Medium, towardsdatascience.com, 26 Nov. 2018, https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05., [Access Date: 08.05.2022].

[18] Jain, Vandit. “Everything You Need To Know About “Activation Functions” In Deep Learning Models | By Vandit Jain | Towards Data Science.” Medium, Towardsdatascience.com, 30 December. 2019, https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253., [Access Date: 08.05.2022].

[19] Brownlee, Jason. “A Gentle Introduction To the Rectified Linear Unit (ReLU) - Machine Learning Mastery.” Machine Learning Mastery, Machinelearningmastery.com, 8 January. 2019, https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/., [Access Date: 08.05.2022].

[20] Yang, Jing, et al. "Real-time tiny part defect detection system in manufacturing using deep learning." IEEE Access 7 (2019): 89278-89291.

[21] Dickson, Ben. “An Introduction To Object Detection With Deep Learning – TechTalks.” TechTalks, Bdtechtalks.com, 21 June. 2021, https://bdtechtalks.com/2021/06/21/object-detection-deep-learning/., [Access Date: 08.05.2022].

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

[23] Oinar, Chingis. “Object Detection Explained: R-CNN | By Chingis Oinar | Towards Data Science.” Medium, Towardsdatascience.com, 22 March. 2021, https://towardsdatascience.com/object-detection-explained-r-cnn-a6c813937a76., [Access Date: 08.05.2022].

[24] Guan, Tongfan, and Hao Zhu. "Atrous faster R-CNN for small scale object detection." 2017 2nd International Conference on Multimedia and Image Processing (ICMIP). IEEE, 2017.

[25] Guan, Tongfan, and Hao Zhu. "Atrous faster R-CNN for small scale object detection." 2017 2nd International Conference on Multimedia and Image Processing (ICMIP). IEEE, 2017.

[26] Gao, Hao. “Faster R-CNN Explained.” Medium, Medium.com, 5 October. 2017, https://medium.com/@smallfishbigsea/faster-r-cnn-explained-864d4fb7e3f8., [Access Date: 08.05.2022].

[27] Świeżewski, Jędrzej. “Introduction To YOLO Algorithm And YOLO Object Detection - Appsilon | Enterprise R Shiny Dashboards.” Appsilon | Enterprise R Shiny Dashboards, Appsilon.com, 22 May. 2020, https://appsilon.com/object-detection-yolo-algorithm/., [Access Date: 08.05.2022].

[28] Du, Juan. "Understanding of object detection based on CNN family and YOLO." Journal of Physics: Conference Series. Vol. 1004. No. 1. IOP Publishing, 2018.

[29] H. Rashed, E. Mohamed, G. Sistu, V. Kumar, C. Eising, A. El-Sallab, S. Yogamani. "Generalized object detection on fisheye cameras for autonomous driving: Dataset, representations and baseline." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2021.

[30] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[31] Almog, Uri. “YOLO V3 Explained. In This Post We’ll Discuss the YOLO… By Uri Almog Towards Data Science.” Medium, Towardsdatascience.com, 13 October. 2020, https://towardsdatascience.com/yolo-v3-explained-ff5b850390f., [Access Date: 08.05.2022].

[32] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[33] Adarsh, Pranav, Pratibha Rathi, and Manoj Kumar. "YOLO v3-Tiny: Object Detection and Recognition using one stage improved model." 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS). IEEE, 2020.

[34] Gupta, Manish. “YOLO—You Only Look Once.” Medium, Towardsdatascience.com, 30 May. 2020, https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4., [Access Date: 08.05.2022].

[35] Bandyopadhyay, Hmrishav . “YOLO: Real-Time Object Detection Explained.” YOLO: Real-Time Object Detection Explained, Www.v7labs.com, 31 January. 2022, https://www.v7labs.com/blog/yolo-object-detection#yolo-versions., [Access Date: 08.05.2022].

[36] Rosebrock, Adrian. "Intersection over Union (IoU) for object detection." Diambil kembali dari PYImageSearch https//www. pyimagesearch. com/2016/11/07/intersection-over-union-iou-for-object-detection., [Access Date: 08.05.2022].

[37] Tan, Ren Jie. “Breaking Down Mean Average Precision (mAP) | by Ren Jie Tan | Towards Data Science.” Medium, towardsdatascience.com, 2 Mar. 2022, https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52., [Access Date: 08.05.2022].

[38] Yohanandan, Shivy. “MAP (mean Average Precision) Might Confuse You! | By Shivy Yohanandan | Towards Data Science.” Medium, Towardsdatascience.com, 9 June. 2020, https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2., [Access Date: 08.05.2022].

[39] Redmon, Joseph. “Darknet: Open Source Neural Networks In C.” Darknet: Open Source Neural Networks In C, Pjreddie.com, 2018, https://pjreddie.com/darknet/., [Access Date: 08.05.2022].

[40] Akkas, Selahattin, Sahaj Singh Maini, and Judy Qiu. "A fast video image detection using tensorflow mobile networks for racing cars." 2019 IEEE International Conference on Big Data (Big Data). IEEE, 2019.

[41] Kuzmic, Jurij, and Günter Rudolph. "Object Detection with TensorFlow on Hardware with Limited Resources for Low-power IoT Devices.", 2021.

[42] Potter, Rayan. "What Is Data Annotation And What Are Its Advantages? | By Rayan Potter | ANOLYTICS | Medium." Medium, Medium.com, 19 February. 2021, https://medium.com/anolytics/what-is-data-annotation-and-what-are-its-advantages-95766213351e., [Access Date: 08.05.2022].

[43] Nelson , Joseph. "Getting Started With LabelImg for Labeling Object Detection Data." Roboflow Blog, Blog.roboflow.com, 16 March. 2020, https://blog.roboflow.com/labelimg., [Access Date: 08.05.2022].

[44] J. Bhattacharyya. "Step By Step Guide To Object Detection Using Roboflow." Analytics India Magazine, Analyticsindiamag.com, 11 October. 2020, https://analyticsindiamag.com/step-by-step-guide-to-object-detection-using-roboflow/., [Access Date: 08.05.2022].

[45] Blog, Everitt's. "YOLO Vs Faster RCNN | Everitt's Blog." Everitt's Blog, Everitt257.github.io, 10 August. 2018, https://everitt257.github.io/post/2018/08/10/object_detection.html., [Access Date: 08.05.2022].

[46] "Google Colaboratory." Google Colab, colab.research.google.com, https://colab.research.google.com/., [Access Date: 08.05.2022].

[47] Team, Spyder. "Home — Spyder IDE." Home — Spyder IDE, www.spyder-ide.org, https://www.spyder-ide.org/., [Access Date: 08.05.2022].

[48] "Space to Store Data in the Cloud for Use at Work and at Home with Google Drive." Space to Store Data in the Cloud for Use at Work and at Home with Google Drive, www.google.com, https://www.google.com/intl/ar/drive/., [Access Date: 08.05.2022].

RESUME

Name Surname	**:** Ahmad Sabah Abdullah KUBAJI

EDUCATION

Degree	**School**	**Graduation Year**
Master	T.C. Sakarya University/ Department of Computer and Information Engineering	Ongoing
Undergraduate	AlHadba'a University - Mosul / Department of Computer Science	2004
High School	Alressalah High School, Mosul	2000

JOB EXPERIENCE

Year	**Place**	**Position**
2017-Still	ALBA Group/Ferrero	IT Director
2011-2017	ALMAHA Company/Nestle	Iraq IT Manager
2009-2011	ALMAHA Company/Nestle	Branch Manager
2006-2009	ALMAHA Company/Nestle	Oracle Developer

FOREIGN LANGUAGE

English

HOBBIES

Swmming , Boxing, Shooting