

**OBJECT DETECTION AND DISTANCE
ESTIMATION USING YOLO ARCHITECTURE**

MAY THU AUNG

M.C.Tech.

DECEMBER 2022

**OBJECT DETECTION AND DISTANCE
ESTIMATION USING YOLO ARCHITECTURE**

BY

MAY THU AUNG

M.C.Tech.

DECEMBER 2022

**OBJECT DETECTION AND DISTANCE
ESTIMATION USING YOLO ARCHITECTURE**

BY

MAY THU AUNG

**A dissertation submitted in partial fulfillment of the
requirements for the degree of**

**Master of Computer Technology
(M.C.Tech.)**

UNIVERSITY OF COMPUTER STUDIES, YANGON

DECEMBER 2022

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to those who helped me with various aspects of conducting research and writing this thesis. Many things are needed to complete this thesis like suggestions from the teachers and supporting, appreciating of my family and friends.

First and foremost, I would like to express my deepest gratitude and my sincere thanks to **Dr. Mie Mie Khin**, Rector, University of Computer Studies, Yangon, for her kind permission to submit this thesis.

My sincere thanks and regards go to **Dr. Htar Htar Lwin**, Pro-rector, Head of Faculty of Computer Systems and Technologies, University of Computer Studies, Yangon, for her kind management throughout the completion of this thesis.

Specially thanks and regards go to **Dr. Yadanar Thein**, Pro-rector, University of Computer Studies, Yangon, for her kind suggestions throughout the completion of this thesis.

I would like to express my deeply thanks to **Dr. Amy Tun**, Professor, Course Coordinator of Master's (CT), Faculty of Computer Systems and Technologies, University of Computer Studies, Yangon, for her painstaking suggestion and encouragement throughout the development of the thesis.

My sincere thanks and regards go to my supervisor, **Dr. Khaing Khaing Wai**, Professor, Head of Department of Information Technology Support and Maintenance, University of Computer Studies, Yangon, for her support, guidance, supervision, patience and encouragement during the period of study towards completion of this thesis.

I also wish to express my deepest gratitude to **Daw Mya Hnin Mon**, Associate Professor, Department of English, University of Computer Studies, Yangon, for her editing this thesis from the language point of view.

Moreover, I would like to extend my thanks to all my teachers who taught me throughout the master's degree course and my friends for their cooperation.

I especially thank to my parents, all of my colleagues, and friends for their encouragement and help during my thesis.

ABSTRACT

Detecting various classes of objects and measuring the distance between camera and objects are implemented in this system. The input of the system is an image or video which are captured by the camera. YOLOv5 architecture is used to detect the objects of input image and calculated the distance between camera and detected objects. If the object is detected, the result will be shown with distance meter values. The major objective of this system is to find the instances of each object in digital photos or real-time videos predictions. In this system, the focal length value is employed to calculate the distance between the camera and the object. The focal length is calculated using the triangle formula utilizing the bounding box's width and height parameters. YOLOv5 object detector provided the height and width values of bounding box. Object detection is also useful in video surveillance and image retrieval systems. In the future, this system can be upgraded as a part of the autonomous vehicles to move automatically in real environment.

TABLE OF CONTENTS

	PAGES
ACKNOWLEDGEMENTS	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
LIST OF FIGURES	v
LIST OF EQUATIONS	vii
CHAPTER 1 INTRODUCTION	1
1.1 Objectives of the Thesis	2
1.2 Motivation of the System	2
1.3 Organization of the Thesis	2
1.4 Chapter Summary	2
CHAPTER 2 BACKGROUND THEORY	3
2.1 Convolutional Neural Network (CNN)	3
2.1.1 Working of CNN	3
2.2 You Only Look Once (YOLO)	11
2.2.1 Concept of YOLO Algorithm	11
2.2.2 Specifying Label Vector and Predict Object	12
2.3 Performances of YOLO Versions	13
2.4 Structure of YOLOv5 Architecture	15
2.4.1 Backbone: Focus structure and CSP network	15
2.4.2 Neck: Additional Block (SPP block)	17
2.4.3 Neck: Feature Aggregation (PANet)	19
2.4.4 Head: Output using GIoU-Loss	22
2.5 YOLOv5 Activation Function	22
2.6 YOLOv5 Loss Function	23
2.6.1 Balance Losses	23
2.6.2 Eliminate Grid Sensitivity	24
2.8 Calculate mAP Values for YOLOv5	25
2.8.1 Intersection over Union (IOU)	25

	2.8.2 Precision and Recall	26
	2.8.3 Precision Recall Curve	26
	2.8.4 Average Precision (AP)	27
	2.9 Chapter Summary	27
CHAPTER 3	THE PROPOSED METHODOLOGY	28
	3.1 Overview of The Proposed System	28
	3.2 COCO Dataset	28
	3.3 Object Detection Using YOLOv5 Architecture	29
	3.4 Distance Estimation	31
	3.5 Running Environment (PyTorch)	32
	3.6 Chapter Summary	33
CHAPTER 4	IMPLEMENTATION AND EXPERIMENTAL RESULTS	34
	4.1 Implementation of The System	34
	4.2 Experimental Results of The System	35
	4.3 Performance Evaluation for The System	39
	4.4 Chapter Summary	42
CHAPTER 5	CONCLUSION	43
	5.1 Advantages	43
	5.2 Limitation	43
	5.3 Related Works	44
	5.4 Further Extension	44
	REFERENCES	45
	LIST OF PUBLICATIONS	47

LIST OF FIGURES

FIGURE		PAGES
Figure 2.1	Comparing the Piece of Image by Section	4
Figure 2.2	Resulting Three Features or Filters	4
Figure 2.3	Multiply Corresponding Pixel Values	4
Figure 2.4	Adding and Diving by Total Number of Pixels	5
Figure 2.5	Create Map and Put an Amount of Filter	5
Figure 2.6	Perform Filtering in another Location	5
Figure 2.7	Convolution Layer Output	6
Figure 2.8	Convolution Layer Output For Each Filter	6
Figure 2.9	ReLU Layer	6
Figure 2.10	Sample Values of ReLU Layer	7
Figure 2.11	Remove Negative Values from Example Image	7
Figure 2.12	Output for One Feature	7
Figure 2.13	Output for All Features	8
Figure 2.14	Calculating the Maximum Value in Each Window	8
Figure 2.15	Moving Window across the Entire Image	9
Figure 2.16	Output After Passing Through Pooling Layer	9
Figure 2.17	Stacking Up the Convolution, ReLU and Pooling Layers	9
Figure 2.18	Reduce Image From 4×4 to 2×2 Matrix Using Double Stacking Layers	10
Figure 2.19	Final Result of Classification	10
Figure 2.20	YOLO Model With 7x7 Grid Cell Was Applied on Input Image	12
Figure 2.21	Specifying Label Vector y for 3x3 Grid Cells and Predict Object for 3 Classes	13
Figure 2.22	Darknet19 Architecture	14
Figure 2.23	Process of Input Processing in an Original Dense Block	16
Figure 2.24	Process of Input Processing in a CSP Dense Block	16
Figure 2.25	Dense Block Connection and Spatial Pyramid Pooling Based YOLOv2 Architecture	17
Figure 2.26	Classical SPP Block	18

Figure 2.27	New SPP Block Adapted to YOLO	19
Figure 2.28	Original FPN Architecture	19
Figure 2.29	Modified FPN Architecture Used in YOLOv3	20
Figure 2.30	PANet Architecture Including FPN Backbone	20
Figure 2.31	PANet Architecture Including Bottom-up Path Augmentation	21
Figure 2.32	PANet Architecture Including (a) FPN Backbone, (b) Bottom-up Path Augmentation, (c) Adaptive Feature Pooling	21
Figure 2.33	PAN Used ROI Align for Adaptive Pooling and Fully Connected Layers for Fusing Features from All Stages	22
Figure 2.34	SiLU Function Graph	23
Figure 2.35	Sigmoid Function Graph	23
Figure 2.36	Grid Sensitivity of YOLOv5 Model	24
Figure 2.37	Intersection over Union (IOU)	25
Figure 2.38	Precision Recall Curve Sample	26
Figure 3.1	Overview of The Proposed System	28
Figure 3.2	Architecture of YOLOv5 Model	30
Figure 4.1	Flowchart of the System	34
Figure 4.2	Run Window Command Prompt as Administrator	35
Figure 4.3	Detect and Save Detected Image File	36
Figure 4.4	Detected Image Result 1	36
Figure 4.5	Detected Image Result 2	37
Figure 4.6	Detected Video Result (a)	37
Figure 4.6	Detected Video Result (b)	38
Figure 4.6	Detected Video Result (c)	38
Figure 4.7	Precision Values of Test Set 1	39
Figure 4.8	Precision Values of Test Set 2	39
Figure 4.9	Precision Values of All Test Set	40
Figure 4.10	Precision Recall Curve of The Test Set 1	40
Figure 4.11	Precision Recall Curve of The Test Set 2	41
Figure 4.12	Precision Recall Curve of The All Test Set	41

LIST OF EQUATIONS

EQUATION	PAGES
Equation 2.1	11
Equation 2.2	23
Equation 2.3	23
Equation 2.4	24
Equation 2.5	24
Equation 2.6	24
Equation 2.7	24
Equation 2.8	26
Equation 2.9	26
Equation 3.1	31
Equation 3.2	31

CHAPTER 1

INTRODUCTION

AI based computer vision tasks are currently popular to improve smart city technology. Computer vision is an incorporative field that allows not only systems but also computers to derive capable information from videos, images, and other types of visual inputs. AI based computer vision helps computers can think, recognize and realize the real environment. In place of optic nerves, retinas and visual cortex, it mimics the human eye and is used to train models to carry out a variety of tasks using cameras, algorithms, and data.

One of the computer vision tasks that detects things of a specific type within an image is object detection. One-stage methods and two-stage methods are the two primary categories under which it can be divided. One-stage techniques like YOLO, SSD, and RetinaNet focus on inference speed. Two-stage approaches prioritize accurate detection, and Faster R-CNN, Mask R-CNN, and Cascade R-CNN are three examples of such models. The MSCOCO dataset is the most well-liked one. Usually, a Mean Average Precision metric is used to evaluate models.

The YOLO method, which stands for "You Only Look Once," was developed by researcher Joseph Redmon and colleagues in 2015. It is an object recognition system that executes all the necessary phases to recognize an object using a single neural network for the first time. Over the years, the YOLO algorithm has been improved upon, including the original version, with many of the most ground-breaking concepts coming from the computer vision research field.

YOLOv5x, YOLOv5l, YOLOv5m, YOLOv5s and YOLOv5n are five versions of YOLOv5 model. In this system, YOLOv5s is used for object detection, image classification and calculate the distance between camera and object. By going straight from image pixels to bounding box coordinates and class probabilities, it reframes the object detection problem as a single regression problem. This integrated model simultaneously forecasts various bounding boxes and class probabilities for items covered by boxes.

The camera provides the system with an image or video as input. The YOLOv5 model will recognize the object classes in the input image or video and calculate the distance between the camera and those recognized objects. Finally, the output result with class labels and distance meters is saved in the OS path.

1.1 Objectives of the Thesis

The following facts are described as the thesis's objective.

- ❖ To detect, classify and label the objects
- ❖ To find the distance of each object in digital photos or real-time videos
- ❖ To measure the distance between camera and detected objects
- ❖ To apply YOLOv5s architecture in OS as object detection and distance estimation model

1.2 Motivation of the Thesis

The motivation of the thesis is to detect various objects and measure the distance between camera and detected objects using YOLOv5 model and it can be applied in AI-based autonomous driving system. This system can be used in CCTV remote control for social distancing to prevent Covid-19. It can be also implemented in the drones for lane finding and landing.

1.3 Organization of the Thesis

Five chapters make up this thesis. One-stage object detector, the thesis's subject, its purpose, and its organization are all introduced in Chapter 1. The operation of CNN and the YOLO (You Only Look Once) algorithm are discussed in Chapter 2 as the system's underlying theory for object identification and distance estimation. The proposed methodology is explained in Chapter 3. The system's software implementation, system design and experiment findings are presented in Chapter 4. Chapter 5 presents the conclusion, benefits, limitations, and future extension.

1.4 Chapter Summary

This chapter includes the about object detection and distance estimation which are one of the computer vision task. This chapter introduce one-stage object detector YOLO algorithm and YOLOv5s model which is used in this thesis as object detection and distance estimation model. The objectives, motivation and organization of the thesis are also described in this chapter.

CHAPTER 2

BACKGROUND THEORY

Image Processing based object detection and distance measuring between camera and object of an image using one-stage detector is implemented with YOLO algorithm in this system.

YOLO is a real-time neural network-based object detection system. This algorithm is frequently used because of its efficiency and precision. In a number of applications, it has been used to detect traffic lights, pedestrians, parking meters, and animals. The YOLO technique uses convolutional neural networks (CNN) to detect objects in real-time and only needs one forward propagation via a neural network. In other words, the whole image is predicted using a single algorithm run. Multiple class probabilities and bounding boxes are simultaneously predicted using the CNN.

2.1 Convolutional Neural Network (CNN)

A sort of artificial feed-forward network called a Convolutional Neural Network (CNN) draws its connectivity pattern from the way that the visual brain of animals is organized. Specific regions of the visual field are responsive to a small subset of visual cortex cells. The geographical correlations in the input data are used by CNN. Each concurrent layer of the neural network connects a few input neurons. The area is referred to as the "local receptive field". The primary emphasis of the receptive field is local neurons. The hidden neuron analyzes the input data inside the specified field without being aware of changes occurring outside the boundary.

2.1.1 Working of CNN

Convolutional Neural Networks are composed of the following layers:

- ❖ Convolutional Layer
- ❖ ReLU Layer
- ❖ Pooling Layer
- ❖ Fully Connected Layer

Using CNN to extract small patches from an image, these pieces or patches are known as filters. CNN compares the image section by section. By finding rough matches in roughly the same position in two images, CNN outperforms whole-image

matching schemes in detecting similarity. Figure 2.1 shows the example of comparing the piece of image by section.

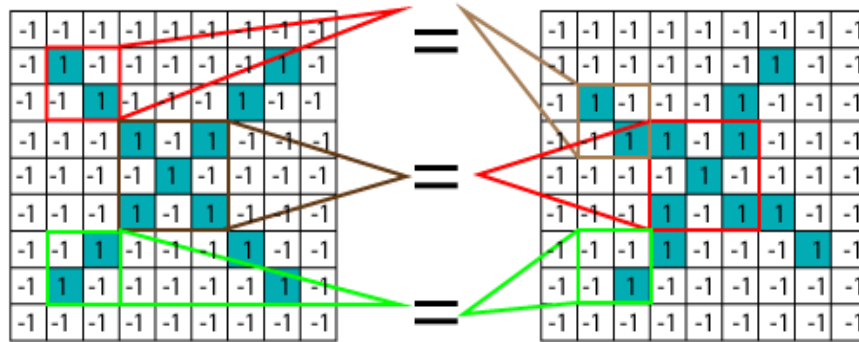


Figure 2.1 Comparing the Piece of Image by Section

After comparing a picture section by section, three features or filters are acquired. The results are shown in Figure 2.2.

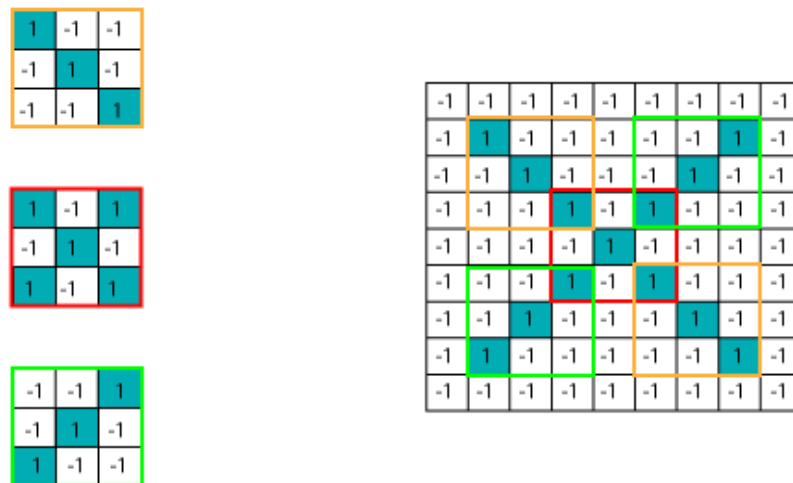


Figure 2.2 Resulting Three Features or Filters

Then, multiply the corresponding pixel values after obtaining the three filters.

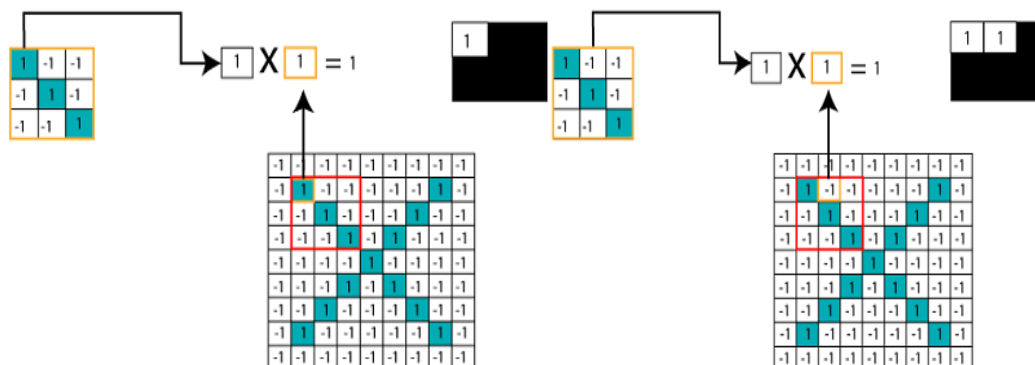


Figure 2.3 Multiply Corresponding Pixel Values

Add and divide the total number of pixels after multiplying the corresponding pixel values.

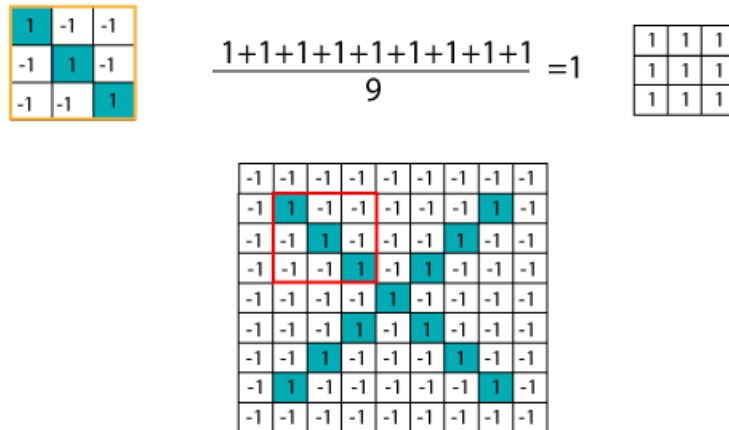


Figure 2.4 Adding and Dividing by Total Number of Pixels

Create a map with several filters to keep track of the features when the values are obtained.

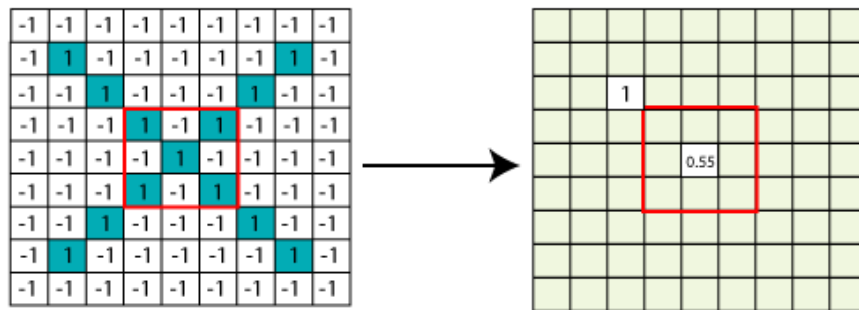


Figure 2.5 Create Map and Put an Amount of Filter

Then repeat the process by moving it to a different location and filtering it again.

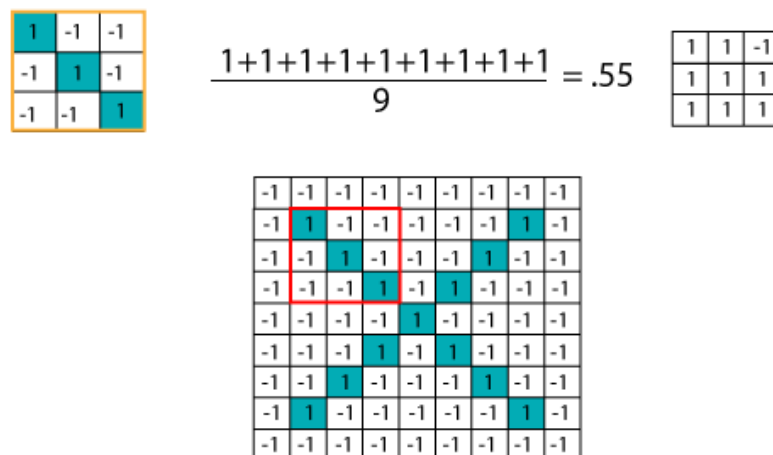


Figure 2.6 Perform Filtering in another Location

Transfer the features to each other location in the image to see well correspond to that area. The final result is shown in Figure 2.7.

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.0	-0.11	0.77	0.33	-0.11	-0.11
0.11	-0.11	1.0	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

Figure 2.7 Convolution Layer Output

Repeat the convolution process with each additional filter as the next step.

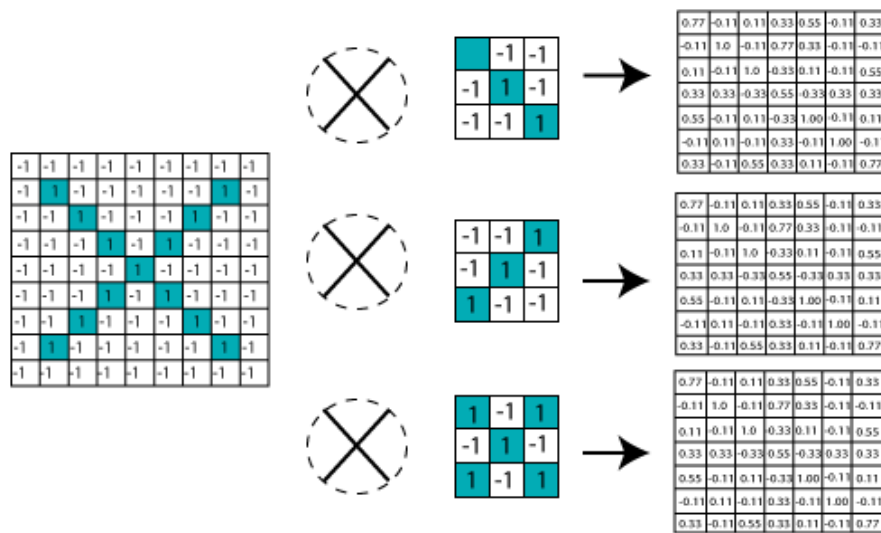


Figure 2.8 Convolution Layer Output for Each Filter

To prevent the values from accumulating up to zero, remove all negative values from the filtered pictures in the ReLU layer and replace them with zeros. A node is only activated by Rectified Linear Unit (ReLU) transform functions if the input value is over a specific threshold. Although the information climbs above a threshold, both the data and the output are zero. It and the dependent variable are related linearly.

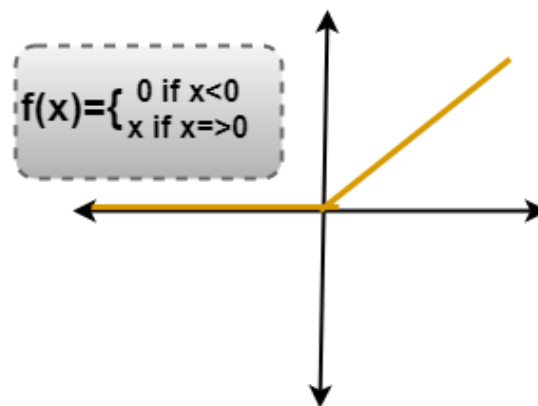


Figure 2.9 ReLU Layer

Consider any simple function with the value shown in the preceding example. As a result, the function only works if the dependent variable has that value. The values obtained, for example, are as follows.

x	F(x) = x	F(x)
-3	F(-3) = 0	0
-5	F(-5) = 0	0
3	F(3) = 3	3
5	F(5)=5	5

Figure 2.10 Sample Values of ReLU Layer

Remove the negative layers from Figure 2.7's convolution layer output.

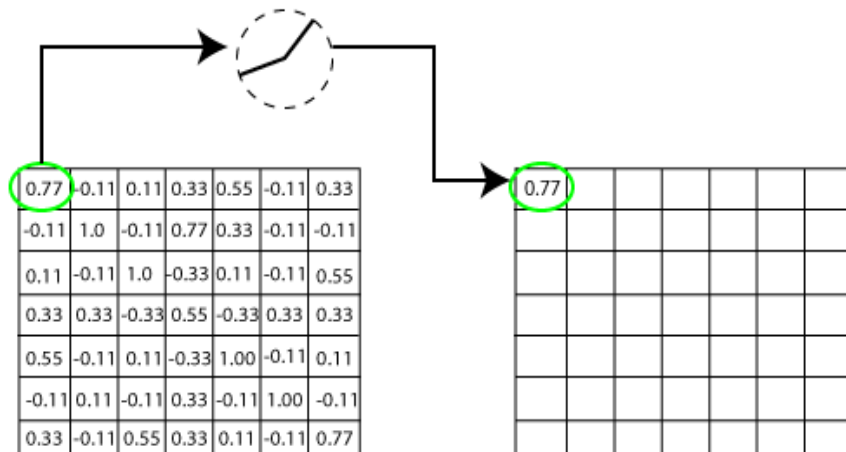


Figure 2.11 Remove Negative Values from Example Image

Then, the output for one feature is shown in Figure 2.12.

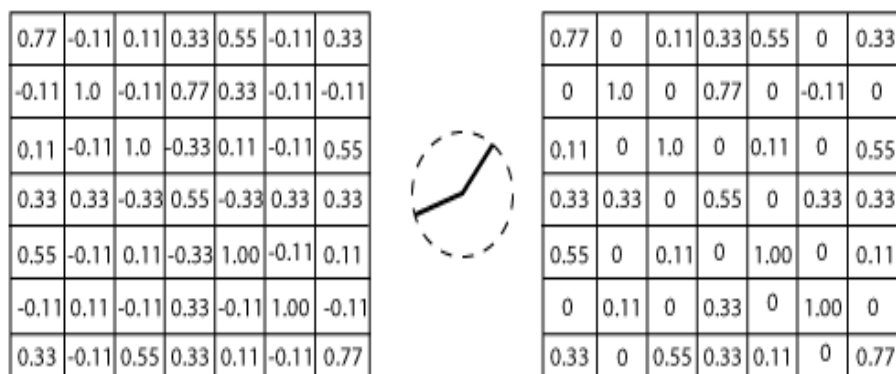


Figure 2.12 Output for One Feature

Removing the negative values for all features are done. The output is shown in Figure 2.13.

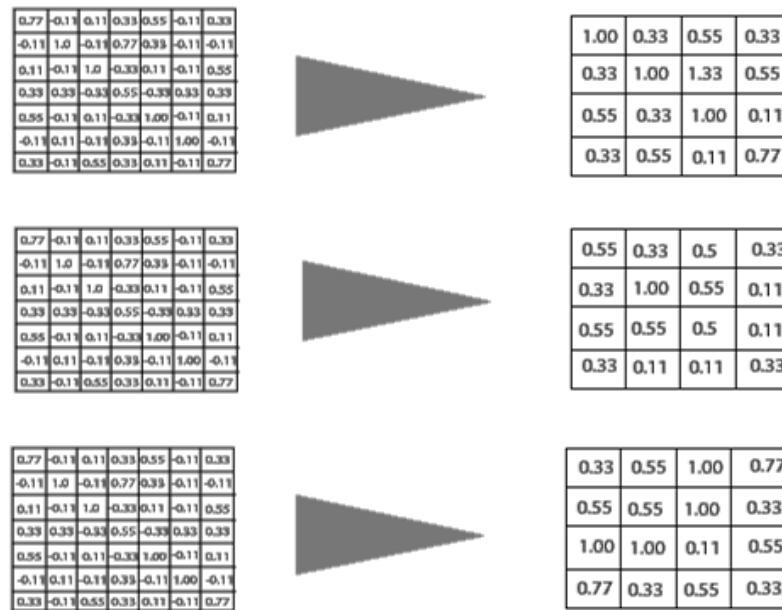


Figure 2.13 Output for All Features

Reduce the size of the image stack in the pooling layer. Pooling is performed after passing by the activation layer. To put pooling into practice, the following four stages are necessary:

- ❖ Choose a window size (commonly 2 or 3)
- ❖ A stride (often 2),
- ❖ And walk the window across the filtered photos.
- ❖ Finally, pick the maximum value from each window.

Consider executing pooling in this example with a window size of 2 and a stride of 2. The first Window's maximum or highest value for the first filtered image is 1. Track that and then advance the Window by two steps.

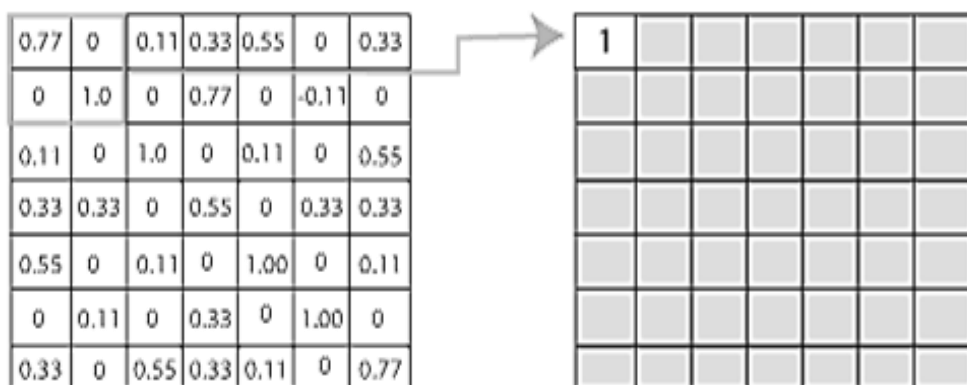


Figure 2.14 Calculating the Maximum Value in Each Window

Next, drag the Window over the entire picture.

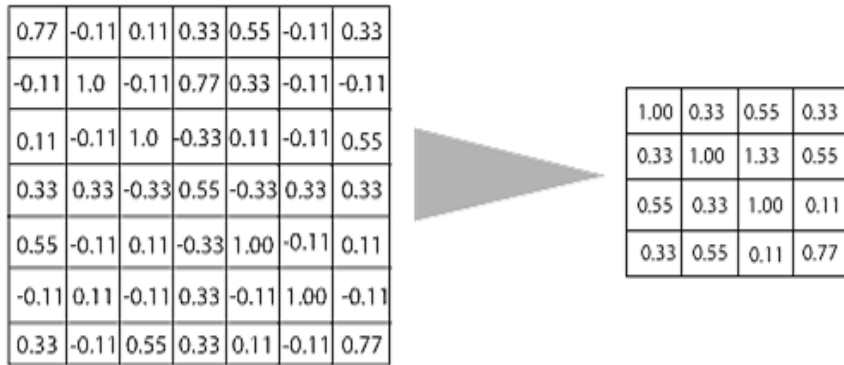


Figure 2.15 Moving Window across the Entire Image

The result of going through the pooling layer is shown in Figure 2.16.

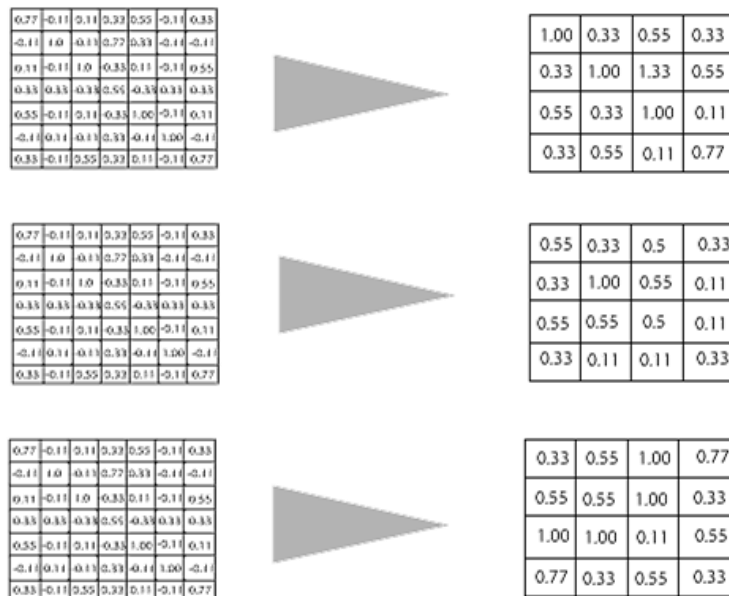


Figure 2.16 Output of After Passing through Pooling Layer

This produces a time-frame in one image with a 4x4 matrix from a 7x7 matrix after passing the input through three layers, Convolution, ReLU, and Pooling, as shown in Figure 2.17.

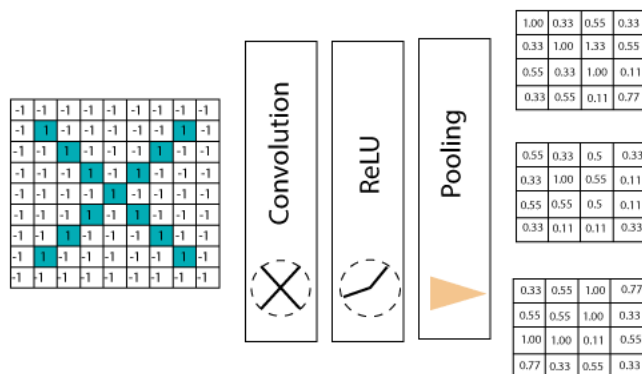


Figure 2.17 Stacking Up the Convolution, ReLU and Pooling Layers

To reduce the image size, double stacking layers are required. The image is reduced from 4×4 to smaller in the first pass. The second pass will then be reduced to a 2×2 matrix, as illustrated in Figure 2.18.

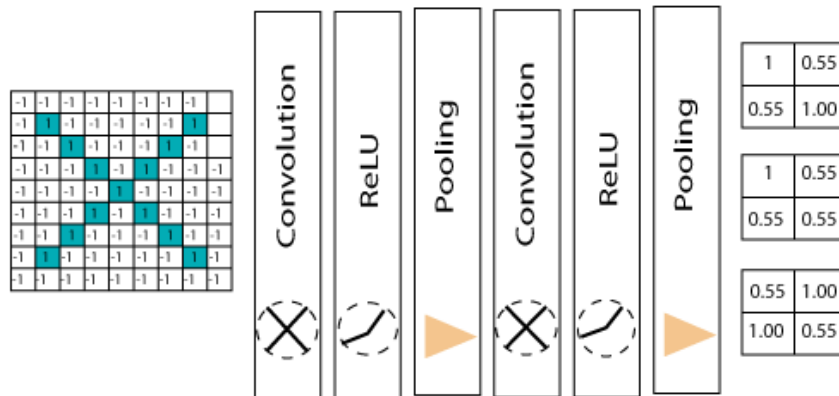


Figure 2.18 Reduce Image From 4×4 to 2×2 Matrix Using Double Stacking Layers

Every neuron in the network's final layer is connected to every other neuron in the layers before and after, which is referred to as the network being fully interconnected. This simulates higher-level reasoning, which takes into consideration all feasible paths from the input to the outcome. Take the reduced image and insert it in the single list after it has undergone two stages of convolution and pooling and been turned into a single file or a vector. Figure 2.19 illustrates the classification process' end outcome.

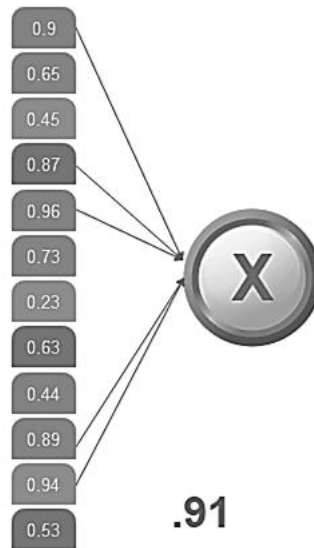


Figure 2.19 Final Result of Classification

2.2 You Only Look Once (YOLO)

The system generates prediction vectors corresponding to each object present in the input image after it has been passed through a single neural network of multiple convolutional networks. The YOLO system computes all the features of the image and produces predictions for all objects at once, as opposed to iterating the process of classifying various parts on the image [7].

The YOLO architecture comes in a number of variations. The YOLO architecture is tied to one iteration. YOLOv5 architecture is used in this system to detect objects and measure the distance between the camera and any detected objects in images or videos.

2.2.1 Concept of YOLO Architecture

The basic goal of YOLOv1 is to insert a grid cell with a default size of $S \times S$ (7×7). If an object's center falls within a grid cell, that grid cell is responsible for detecting the object. Thus, even the appearance of an object that was shown in a number of cells is disregarded by all other cells. To accomplish object detection, each grid cell projects B bounding boxes with their associated characteristics and confidence ratings [19]. These confidence score reflects the presence or absence of an object in bounding box. The confidence score is defined as:

$$\text{Confidence score} = p(\text{Object}) * IOU^{\text{truth\&pred}}$$

Equation 2.1

with $p(\text{Object})$ is the probability that there is an object inside the cell and $IOU^{\text{truth\&pred}}$ is intersection over union of prediction box and ground truth box. The $p(\text{Object})$ range has only between 0 and 1. If there is no object in that cell, the confidence score will nearly close to 0 or oppositely the score will be equal to $IOU^{\text{truth\&pred}}$. That process is shown in Figure 2.20.

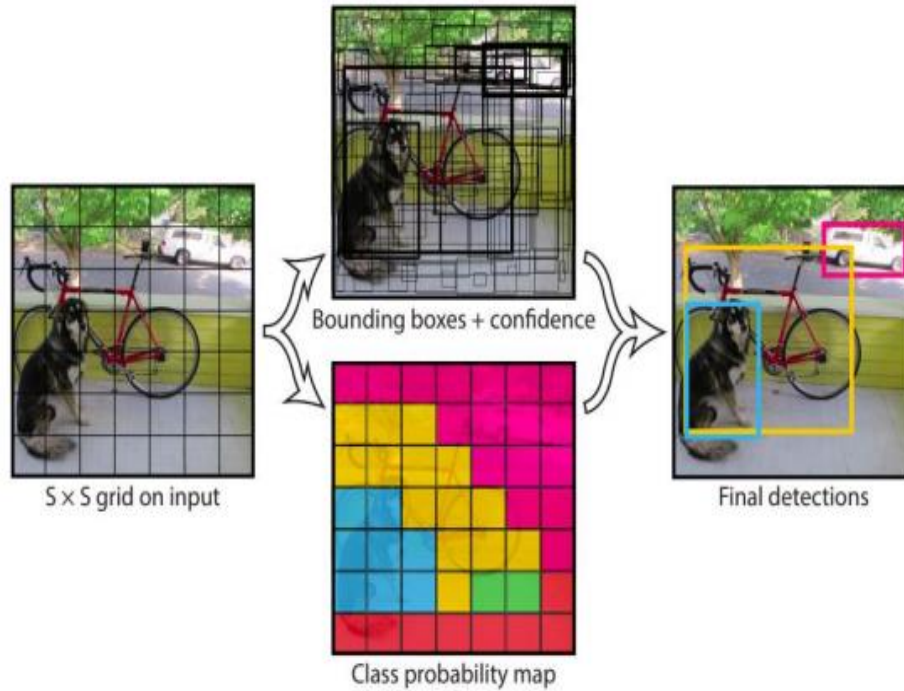


Figure 2.20 YOLO Model With 7x7 Grid Cell Was Applied on Input Image

Each bounding box also has a confidence score and a total of five other factors. $S \times S$ grid cells make up the model's picture space. Each cell forecasts B bounding boxes, which have five parameters and share C class forecast probability. $S \times S (5 * B + C)$ is the total YOLO output of the model parameters.

2.2.2 Specifying Label Vector and Predict Object

The goal of the YOLO architecture is to locate an object using the bounding box's coordinates in order to discover it by accurately anticipating its bounding box. As a result, ground truth bounding box vectors correspond to vector label y and anticipated bounding box vectors to output vector \hat{y} . Given that the purple cell in Figure 2.21 does not contain any objects, the confidence score of the bounding boxes in that cell is equal to 0, hence all other parameters will be disregarded.

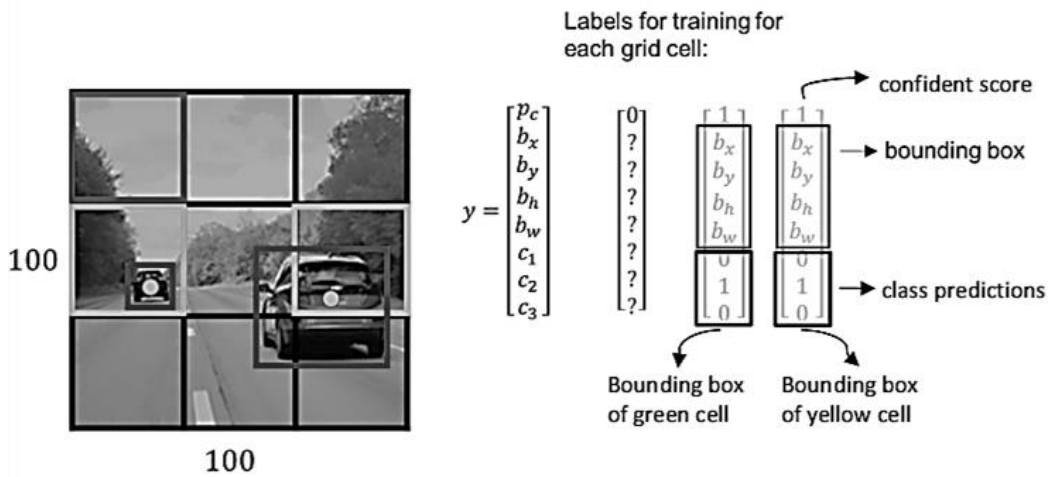


Figure 2.21 Specifying Label Vector y for 3x3 Grid Cells and Predict Object for 3 Classes

In order to clear up all bounding boxes that are empty or contain the same object as other bounding boxes, YOLO lastly uses Non-Maximum Suppression (NMS). NMS removes any overlap bounding boxes with intersection over union (IOU) values higher than the threshold value by selecting a threshold value.

The model evaluates the likelihood that an object will be found within each of the multiple grids it creates from the entering image. This is carried out for each of the image's grids. The program then collects adjacent high-value probability grids into a single object. Non-Max Suppression is a method for eliminating low-value forecasts (NMS).

2.3 Performances of YOLO Versions

YOLOv1 evaluates the likelihood that an object will be found within each of the multiple grids it creates from the entering image. This is carried out for each of the image's grids. Next, the algorithm creates a single item out of all nearby high-value probability grids. Low-value forecasts can be removed using Non-Max Suppression (NMS).

The YOLOv2 was the first version to use anchor boxes. Objects that need to be detected are placed at idealized locations in an image, which is represented by anchor boxes. Ratio of intersection over union (IoU) between the expected bounding box and the predetermined anchor box. The IoU value is used as a threshold to decide whether or not there is a good enough chance that an object has been identified to justify making a forecast. Anchor box calculations are not done at random. The training data is instead

analyzed and clustered using the YOLO technique (dimension clusters). The YOLOv2 model is dynamically scaled throughout the training process to adjust to various aspect ratios called multi-scale training. The COCO dataset and the ImageNet dataset were used to train the YOLOv2 model to ensure its robustness. The model calculates the detection and classification error as it analyses a picture with labels. While a label-less image causes the model to just backpropagate the categorization mistake. The WordTree is the name of this construction. Using a classification network architecture known as Darknet19 shown in Figure 2.22, inference speeds of up to 200 FPS and mAP of 75.3 were attained.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure 2.22 Darknet19 Architecture

Without using fully connected or pooling layers, YOLOv3 included 75 convolutional layers, significantly reducing the model's size and weight. Using residual models (from the ResNet model) for multiple feature learning using feature pyramid networks (FPN) while keeping short inference times gave the best performances. A feature extractor known as a feature pyramid network extracts several types, sizes, and shapes of features from a single image. To enable the model to learn both local and

broad features, it concatenates all the features. The accuracy of the class predictions for the YOLOv3 surpasses that of RetinaNet-50 and 101 thanks to the usage of logistic classifiers and activations. The Darknet53 architecture serves as the backbone for the YOLOv3 model.

Data augmentation methods, bounding box regression loss, regularization, normalization, spatial attention modules (SAM), non-max suppression (NMS), non-linear activation functions, and skip-connections like weighted residual connections (WRC) or cross-stage partial connections (CSP) are among the new features that were added in YOLOv4 [18].

2.4 Structure of YOLOv5 Architecture

YOLOv5 has fast detection speed, high accuracy, and is widely used in real-time object detection. The architecture of the YOLOv5 has three parts;

- ❖ Backbone: Focus structure and CSP network
- ❖ Neck: SPP block and PANet
- ❖ Head: Output using GIoU-loss

2.4.1 Backbone: Focus structure and CSP network

The vanishing gradient problem is solved by the deep network YOLO, which uses residual and dense blocks to allow information to travel to the deepest layers. CSP-Darknet53 serves as the foundation of YOLOv5. DenseNet and Darknet53 architectures are combined to create CSPNet, sometimes known as CSP-Darknet53. By truncating the gradient flow, CSPNet keeps the benefit of DenseNet's feature reuse properties and aids in decreasing the excessive amount of duplicate gradient information [4].

The CSP-core Darknet53's design, known as DenseNet, utilizes the prior input and concatenates it with the current input before proceeding into the dense layer (CSP stands for Cross Stage Partial). In order to address vanishing gradient issues, DenseNet was created to connect layers in a very deep neural network (as ResNet) [4].

Each step of the DenseNet architecture includes a dense block and a transition layer. There are k dense layers that made up each dense block. The transition layer is where the input goes after passing through the dense block to change size (downsample or upsample), perform convolution, and perform pooling. The input for the following layer $(i + 1)^{th}$ will be formed by concatenating the output of the i^{th} dense layer with its

own input. For instance, at the first dense layer, the input x_0 has generated the output x_1 after being passed forward through convolutional layers. The output x_1 is then concatenated with its own input x_0 , and the result of this concatenation becomes the input of the second dense layer. Figure 2.23 depicts the DenseNet architecture method [10].

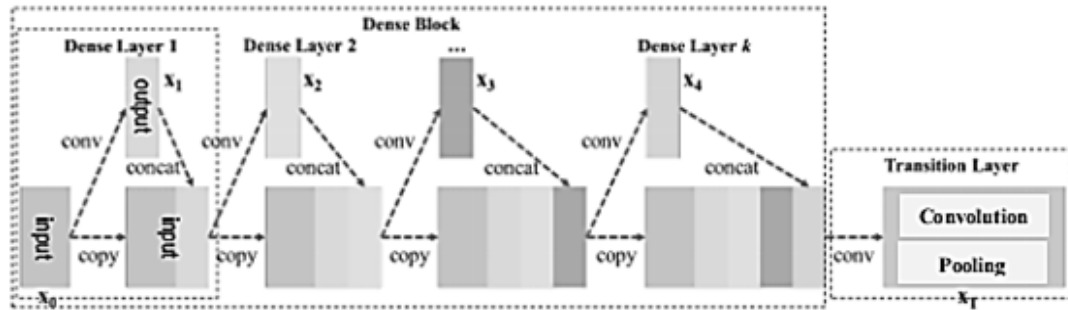


Figure 2.23 Process of Input Processing in an Original Dense Block

Cross Stage Partial (CSP) is built on the same theory as the aforementioned DenseNet, with the exception that the input will be divided into 2 portions rather than using the full-size input feature map of the foundation layers. As shown in Figure 2.24, some of the data will proceed normally through the dense block while other portions will be transferred directly to the next stage without being processed. As a result, several dense layers will continuously learn copied gradient information [10].

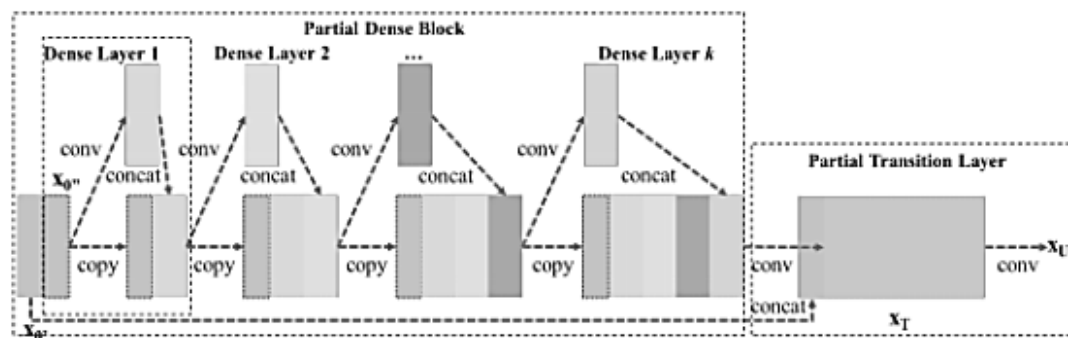


Figure 2.24 Process of Input Processing in a CSP Dense Block

Convolutional neural network Darknet53 serves as the YOLOv5 object identification model's structural foundation. The remaining blocks were swapped out for the dense blocks using the concepts from Darknet53 architecture. CSP promotes the network to reuse features, decreases the amount of network parameters, and preserves fine-grained features to enable more effective forwarding to deeper layers. The Darknet53 backbone network's final convolutional block, which can extract the richer semantic characteristics, is upgraded to be a dense block because an excessive increase

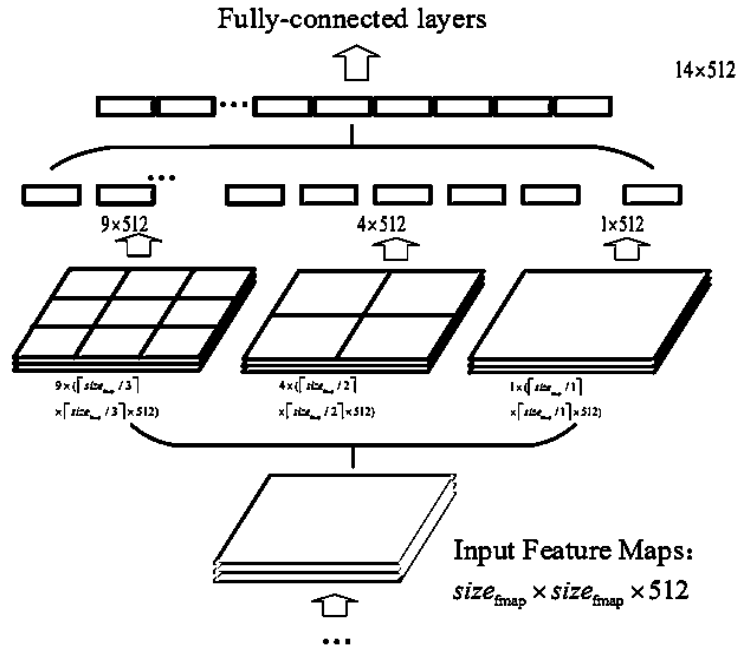


Figure 2.26 Classical SPP Block

Due to the removal of completely linked layers, which enables the input of images in various dimensions, the YOLO algorithm has evolved into an FCN-based (fully convolution network) model. Additionally, using the $S \times S$ grid cell drawn on the image, YOLO must generate predictions and localizations about the positions of the enclosing boxes. Therefore, it may not always be preferable to convert two-dimensional feature maps into a fixed-size one-dimensional vector.

For those reasons, SPP block shown in Figure 2.27 has been altered to maintain the output spatial dimension. A 1×1 convolution is employed between the backbone and the new SPP block to reduce the amount of input feature maps supplied to the SPP block (from 1024 to 512). The new SPP block was placed next to the backbone. The input feature maps are then copied and pooled in various scales using the same methodology as the original SPP block, with the exception that padding is employed to maintain a constant size for the output feature maps, leaving 3 feature maps with the dimensions $size_{fmap} \times size_{fmap} \times 512$.

The new SPP block concatenates these 3 feature maps pooled with the sizes of $size_{fmap} \times size_{fmap} \times 512$ and 30 including the input feature maps to avoid loss of significant features in the case that 3-scale max-pooling is insufficient. This is in contrast to the classical SPP block where the feature maps were converted to a one-dimensional vector after conducting multi-scale max-pooling. As a result, the input retained the spatial dimension in addition to extracting the key aspects that facilitated training.

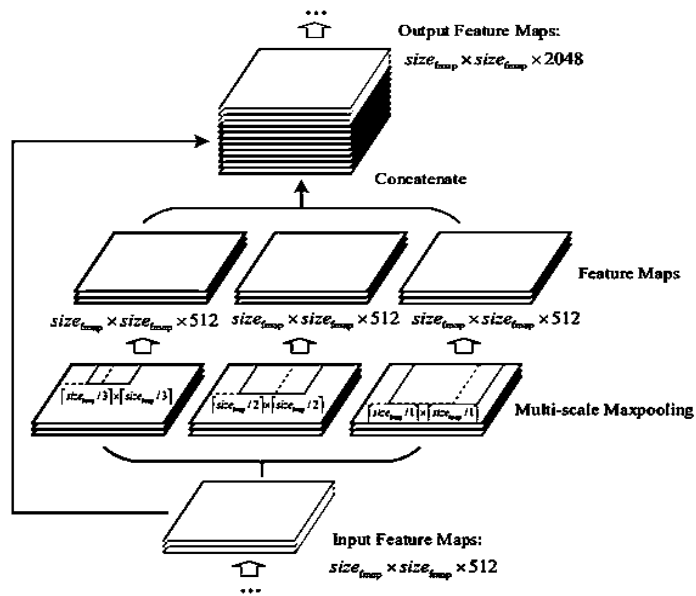


Figure 2.27 New SPP Block Adapted to YOLO

2.4.3 Neck: Feature Aggregation (PANet)

The input picture features are converted into semantical features after being forwarded over the backbone (or learned features). Particularly, as the input image is processed through lower-level layers, the complexity of semantic features will increase while downsampling causes feature maps' spatial resolution to decline significantly. Due to this, fine-grained characteristics and spatial information are lost. The neck of YOLOv5 uses the Feature Pyramid Network (FPN) architecture to preserve these fine-grained features.

Figures 2.28 and 2.29 illustrate the top-down method that the FPN architecture implemented to transmit semantical information (from the high-level layer) and concatenate them to fine-grained features (from the low-level layer in the backbone) for predicting small objects in the large-scale detector [5].

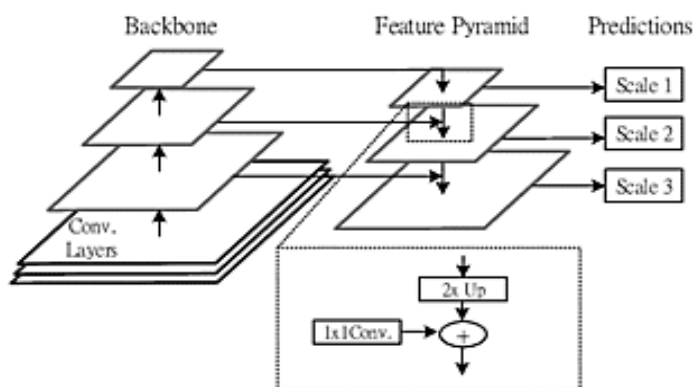


Figure 2.28 Original FPN Architecture

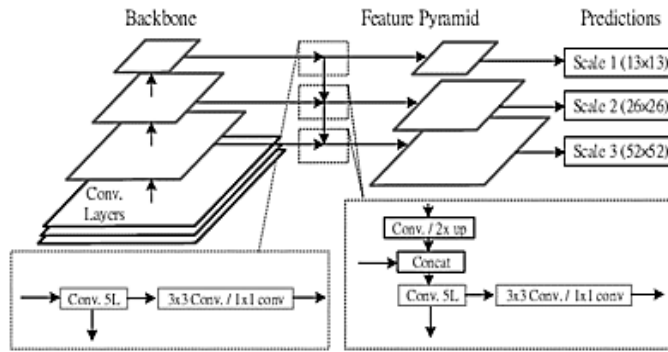


Figure 2.29 Modified FPN Architecture Used in YOLOv3

Due to the top-down flow in the FPN architecture, Path Aggregation Network (PAN) is a more sophisticated variant of FPN. Therefore, in the lateral backbone depicted in Figure 2.30, only the large-scale detector from low-level layers in FPN is able to concurrently receive the semantic data from high-level layers and fine-grained information from low-level layers.

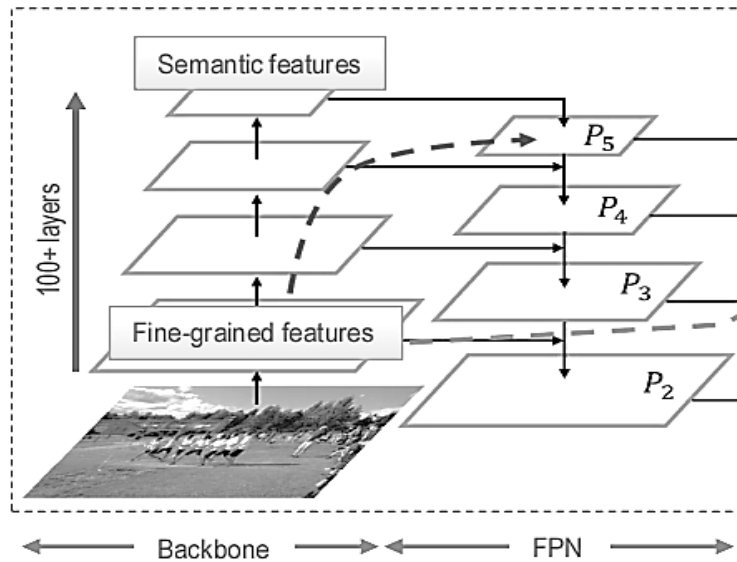


Figure 2.30 PANet Architecture Including FPN Backbone

Recent developments in FPN's small-scale detector limit the application of object detection to semantic features. The idea of concatenating semantic features and fine-grained features at high-level layers was taken into consideration to enhance the performance for the small and medium-scale detector.

There are many layers, possibly over 100, in the deep neural network's backbone. As a result, the fine-grained features in FPN must travel a lengthy distance to get from low-level to high-level layers. In addition to the top-down augmentation method utilized in FPN and depicted in Figure 2.31, a bottom-up augmentation path is proposed for PAN architecture.

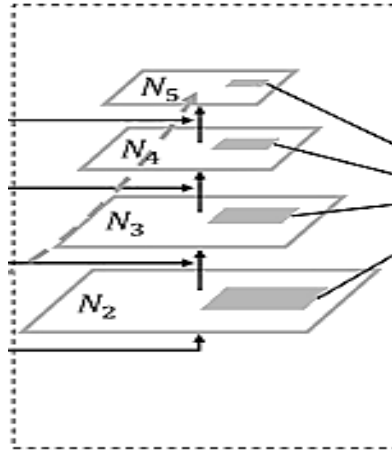


Figure 2.31 PANet Architecture Including Bottom-up Path Augmentation

The direct connection of fine-grained features from lower-level layers to the top ones was thus made possible. This shortcut has fewer than ten layers, which facilitates easy information flow. Figure 2.32 depicts the PAN architecture's overall process.

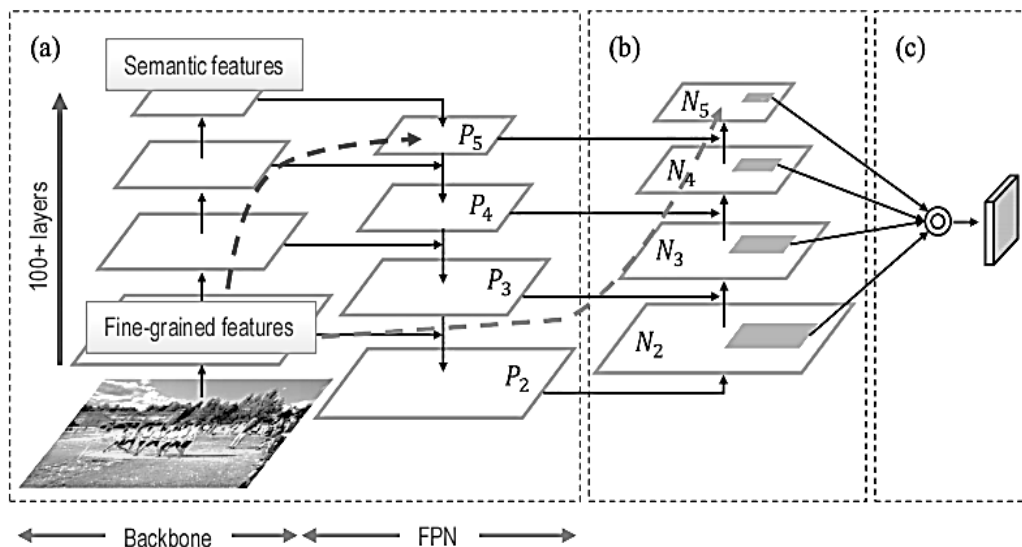


Figure 2.32 PANet Architecture Including (a) FPN Backbone, (b) Bottom-up Path Augmentation, (c) Adaptive Feature Pooling

With layers that yield feature maps with the same spatial sizes at each stage, the bottom-up augmentation path can be seen as a copy of the FPN top-down path. These feature maps are coupled to the lateral architecture via the element-wise addition operation, as opposed to the concatenation operation used in the modified PAN architecture for YOLOv5. This improves the information flow as neither the bottom-up augmentation path features nor the FPN features are lacking. This process is illustrated in Figure 2.33 [18].

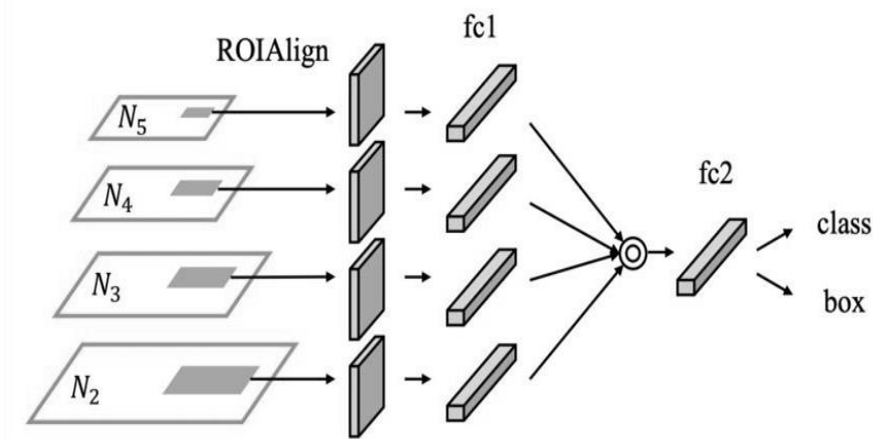


Figure 2.33 PAN Used ROI Align for Adaptive Pooling and Fully Connected Layers for Fusing Features from All Stages

2.4.4 Head: Output using GIoU-Loss

Three output branches in the head anticipate the bounding boxes and groups of objects in various sizes. An $S \times S$ grid is used to be divided into the input image. YOLOv5 predicts B boundary boxes for each grid cell. Each bounding box has three categories of parameters: object confidence C , prediction probabilities P of N classes, and location (x, y, w, h) corresponding to (central coordinate (x, y) , width, and height) of a bounding box. Thus, the bounding box position loss, object confidence loss, and class probability loss make up the loss function. The GIoU loss suggested in the literature is the bounding box position loss used here. The cross-entropy loss function computes the object confidence loss and the class probability loss [12].

2.5 YOLOv5 Activation Function

YOLOv5 activation functions include SiLU and Sigmoid activation functions. Sigmoid Linear Unit, or SiLU, is also known as the swish activation function. Convolutional operations have been employed with it in the buried layers. Convolution procedures in the output layer have been combined with the Sigmoid activation function. Figures 2.34 and 2.35 show graphs of the activation functions used in YOLOv5.

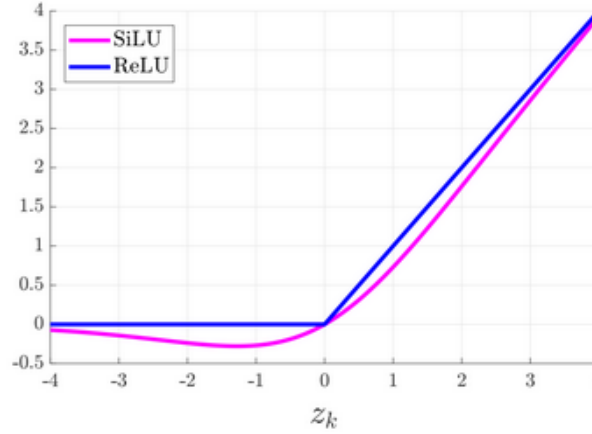


Figure 2.34 SiLU Function Graph

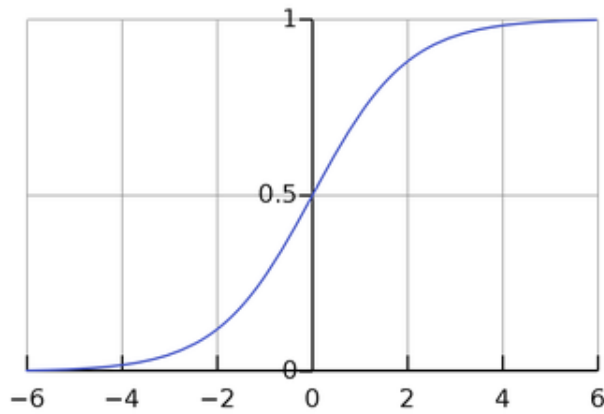


Figure 2.35 Sigmoid Function Graph

2.6 YOLOv5 Loss Function

YOLOv5 returns three outputs. They are the classes of the detected objects, their bounding boxes and the objectness scores. Therefore, it uses Binary Cross Entropy (BCE) to compute the classes' loss and the objectness loss. To compute the location loss used Complete Intersection over Union (CIoU) loss. The final loss formula is given by the following equation.

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}$$

Equation 2.2

2.6.1 Balance Losses

Different weights are applied to the objectness losses of the three prediction layers (P3, P4, and P5). The balancing weights are, correspondingly, [4.0, 1.0, 0.4].

$$L_{obj} = 4.0 \cdot L_{obj}^{small} + 1.0 \cdot L_{obj}^{medium} + 0.4 \cdot L_{obj}^{large}$$

Equation 2.3

2.6.2 Eliminate Grid Sensitivity

To detect bounding boxes on image corners due to the equations used to estimate the bounding boxes, increase the range of the center point offset from (0-1) to (-0.5, 1.5). As a result, the height and width scaling ratios were decreased when training instabilities, and the offset may be easily set to 1 or 0 (coordinates can be in the edge of the image). The method used by YOLOv5 to determine the projected target information is [14]

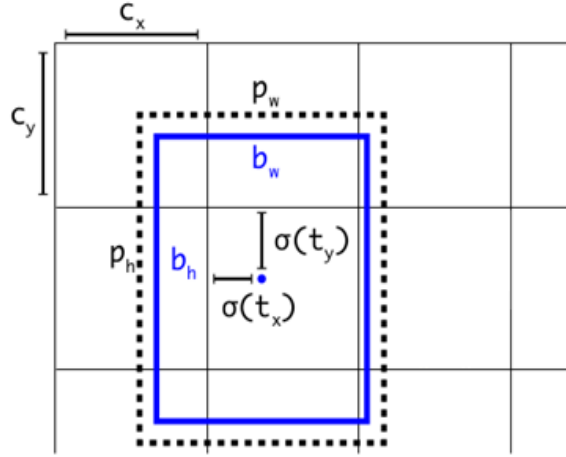


Figure 2.36 Grid Sensitivity of YOLOv5 Model

$$b_x = (2 \cdot \sigma(t_x) - 0.5) + c_x$$

Equation 2.4

$$b_y = (2 \cdot \sigma(t_y) - 0.5) + c_y$$

Equation 2.5

$$b_w = p_w \cdot (2 \cdot \sigma(t_w))^2$$

Equation 2.6

$$b_h = p_h \cdot (2 \cdot \sigma(t_h))^2$$

Equation 2.7

where, b_x = x-axis of final predicted bounding box,

b_y = y-axis of final predicted bounding box,

c_x = x-axis of corner image,

c_y = y-axis of corner image,

p_w = predefined anchor box's width,

p_h = predefined anchor box's height,

t_x = x-axis of bounding box,

t_y = y-axis of bounding box,
 t_w = bounding box's width,
 t_h = bounding box's height,
 b_w = final predicted bounding box's width,
 b_h = final predicted bounding box's height and
 σ = sigmoid function

2.8 Calculate mAP Values for YOLOv5

YOLOv5 employs mAP numbers rather than accuracy to determine performance rate. Object detection models like YOLO use the assessment metric known as Mean Average Precision (mAP). IOU, Precision, Recall, Precision Recall Curve, and AP are needed to calculate mAP.

2.8.1 Intersection over Union (IOU)

It is possible to tell whether the bounding box was accurately predicted using intersection over Union (IOU). How much the bounding boxes overlap is shown by the IOU. The ratio of overlap between the regions of two bounding boxes becomes 1.0 in the case of a precise match, and it becomes 0.0 in the absence of any overlap.

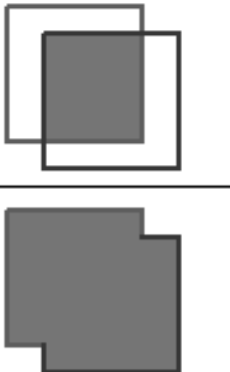
$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{Diagram 1}}{\text{Diagram 2}}$$


Figure 2.37 Intersection over Union (IOU)

It's important to specify how much bounding box overlap with respect to the ground truth data should be regarded as successful recognition when evaluating object detection models. IOUs are utilized for this, and the accuracy at IOU=50 is mAP50. The detection is deemed successful if there is an overlap of greater than 50%. The detection of the bounding box becomes more challenging and requires greater accuracy for larger IOUs. As an illustration, mAP75 has a lower value than mAP50.

2.8.2 Precision and Recall

Precision is a model's capacity to recognize only the pertinent objects. A model with a precision of 1.0 produces no false positives. However, even if there are bounding boxes that should be identified but aren't, the value will still be 1.0.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\text{all detections}}$$

Equation 2.8

where, TP = True positive and

FP = False positive

Recall is a model's capacity to locate every ground truth bounding box. A model with a recall of 1.0 results in no undetected bounding boxes that should be detected. The recall will still be 1.0, even if there is an "overdetection" and the incorrect bounding box is discovered.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\text{all ground truths}}$$

Equation 2.9

where, TP = true positive and

FN = false negative

2.8.3 Precision Recall Curve

Plotting Precision on the vertical axis and Recall on the horizontal axis results in the Precision Recall Curve.

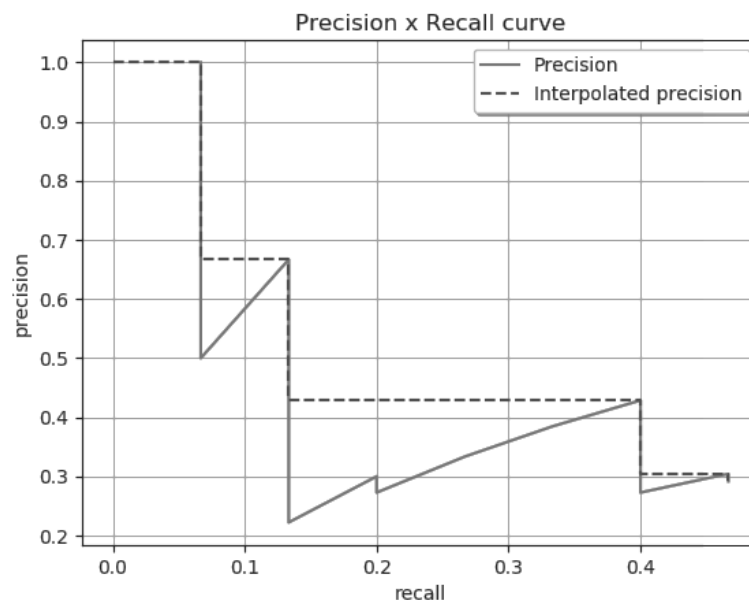


Figure 2.38 Precision Recall Curve Sample

The detection of objects has a threshold. The chance of over-detecting objects is decreased when the threshold is raised, while the risk of missed detections is increased. For instance, if the threshold is set to 1, no item will be found, the Precision is set to 1, and the Recall is set to 0. In contrast, an infinite number of objects will be discovered, Precision will be 0.0, and Recall will be 1.0 if the threshold is set to 0.0. In contrast, an infinite number of objects will be discovered, Precision will be 0.0, and Recall will be 1.0 if the threshold is set to 0. Consequently, the machine learning model is better the further up the curve to the right in the graph [19].

2.8.4 Average Precision (AP)

The greater the Precision Recall Curve, when comparing the performance of two machine learning models, the better the performance. The Average Precision (AP), which reflects the area under the curve (AUC) Precision Recall Curve, is a more logical way to assess models. The greater the area, the higher the AP, and the better the machine learning model are, the steeper the curve is in the upper right corner [19].

2.9 Chapter Summary

This chapter goes into detail on how the CNN algorithm functions, as well as how the various YOLO iterations perform. Additionally, it explains the YOLOv5 model's grid sensitivity, balance losses, and loss function. This chapter also explains how precision and recall are calculated. This chapter includes a description of Average Precision (AP).

CHAPTER 3

THE PROPOSED METHODOLOGY

In this chapter describes the proposed methodology of the system which can identify the input objects and measure the distance from the camera to the objects using YOLOv5 model.

3.1 Overview of The Proposed System

Firstly, camera takes the image or video as the input of the system. YOLOv5 model will detect the object classes of the input image or video and calculate the distance between camera and those detected objects. Finally, the output result is saved in OS path with class labels and distance meters. The summary of the system diagram is shown in Figure 3.1.

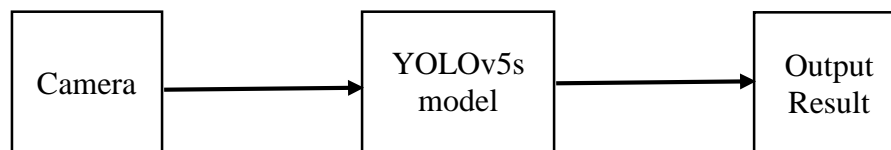


Figure 3.1 Overview of the Proposed System

3.2 COCO Dataset

In this system, YOLOv5 is used for object detection and distance estimation. The COCO dataset is the default dataset of the YOLOv5 architecture and 128 images were used for testing and validation in this system. COCO means Common Objects in Context is a large-scale image dataset containing 328000 images of objects. This contains annotations to train machine learning models to recognize, label and describe objects. The dataset has 80 classes and its image ratio is 640 x 480.

COCO dataset consists of 80 classes. They are person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, traffic light, fire hydrant, stop sign, parking meter, bench, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra, giraffe, backpack, umbrella, handbag, tie, suitcase, frisbee, skis, snowboard, sports ball, kite, baseball bat, baseball glove, skateboard, surfboard, tennis racket, bottle, wine glass, cup, fork, knife, spoon, bowl, banana, apple, sandwich, orange, broccoli, carrot, hot dog, pizza, donut, cake, chair, couch, potted plant, bed, dining table, toilet, tv, laptop, mouse, remote, keyboard, cell phone, microwave, oven, toaster, sink, refrigerator, book, clock, vase, scissors, teddy bear, hair drier and toothbrush.

The COCO dataset offers the following annotations:

- ❖ **Object detection** – bounding box coordinates and complete segmentation masks for 80 different item categories.
- ❖ **Captioning** – each image is described in natural language.
- ❖ **Keypoints** – the dataset includes more than 200,000 photos of more than 250,000 people that are annotated with important details like the right eye, nose, and left hip.
- ❖ **Stuff image segmentation** – pixel maps of 91 different types of objects, including amorphous background areas like the sky, grass, or walls.
- ❖ **Panoptic** – entire picture segmentation, identifying objects in the image in accordance with 91 "stuff" categories (road, sky, water, etc.) and 80 categories of "things" (cat, pen, fridge, etc).
- ❖ **Dense pose** – every tagged individual in the dataset is annotated with an instance id and a mapping between pixels indicating that person's body and a template 3D model. The dataset comprises more than 39,000 photos with more than 56,000 humans in them.

3.3 Object Detection Using YOLOv5 Architecture

In this system, object detection and distance estimates between the camera and identified objects are done using YOLOv5s. One of the YOLOv5 models is the small size object detection model (YOLOv5s). YOLOv5 is frequently used in real-time object identification and has a quick detection speed and good accuracy. Figure 3.2 depicts the YOLOv5 architecture.

First, the focus structure is used to separate the input image into layers. It is used to speed up forward and backward passes while decreasing the number of parameters, FLOPS, and CUDA memory, with only small effects on mean Average Precision (mAP). Backbone is a pre-trained network that uses repeated down-sampling with CSP-Darknet53 to efficiently extract feature information from the input image [12]. This aids in lowering the image's spatial resolution and raising its feature (channel) resolution. The neck is in charge of aggregating the picture features retrieved by the backbone using the bottom-up PANet and FPN's cascade structure [12]. It is employed for accurate generalization to objects of various scales and sizes. The head model

performs the last stage actions, applying anchor boxes to feature maps and rendering the finished product that includes classes, objectness scores, and bounding boxes [17].

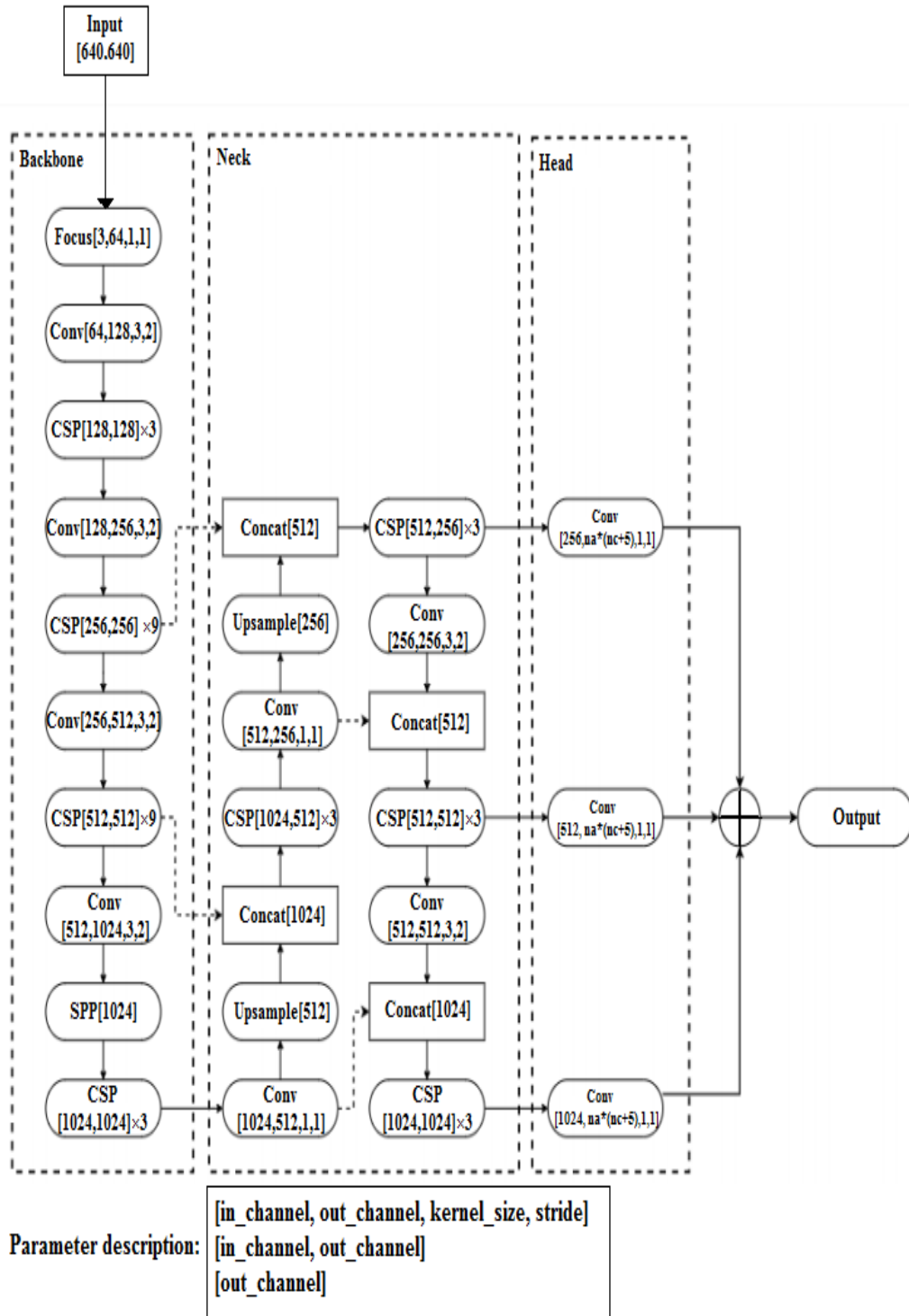


Figure 3.2 Architecture of YOLOv5 Model

An $S \times S$ grid is used to divide up the input image. YOLOv5s predicts B boundary boxes for each grid cell. Each bounding box has three categories of parameters: object confidence C , prediction probabilities P of N classes, and the center

coordinate (x, y), width, and height of the box. Thus, the bounding box position loss, object confidence loss, and class probability loss make up the loss function. Bounding box position loss is implemented using GIoU-loss. The cross-entropy loss function computes the object confidence loss and the class probability loss [12].

$$\begin{aligned}
Loss = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} L_{\text{GIoU}} - \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} [\hat{C}_j^i \log(C_j^i) + (1 - \hat{C}_j^i) \log(1 - C_j^i)] \\
& - \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{noobj}} [\hat{C}_j^i \log(C_j^i) + (1 - \hat{C}_j^i) \log(1 - C_j^i)] \\
& - \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{\text{obj}} \sum_{C \in \text{classes}} [\hat{P}_j^i \log(P_j^i) + (1 - \hat{P}_j^i) \log(1 - P_j^i)]
\end{aligned}$$

Equation 3.1

where I_{ij}^{obj} is defined as 1 if object presents inside j-th predicted bounding box in i-th cell, and 0 for otherwise. I_{ij}^{noobj} is the opposite. λ_{coord} and λ_{noobj} are the loss weights [12].

3.4 Distance Estimation

At first, find the focal length of the camera to estimate the detected object distance from bounding box's width and height which are get from YOLOv5 object detector using triangle formula.

$$Focal\ Length = \sqrt{w^2 + h^2}$$

Equation 3.2

where, w = bounding box's width and

h = bounding box's height

Then, insert into torch library to get the distance between camera and detected objects layer by layer.

3.5 Running Environment (PyTorch)

YOLOv5 is implemented in PyTorch which can give more flexibility to control the encoded operations. Python and the Torch library are the foundation of PyTorch, an open source machine learning (ML) framework. Torch is a Lua scripting language-based open source machine learning library that is used to build deep neural networks. One of the most popular platforms for deep learning research is this one. The following are some of PyTorch's important features:

- ❖ **Tensor computation** - Similar to NumPy array which is an open source library of Python that adds support for large, multidimensional arrays. Tensors are generic n-dimensional arrays used for arbitrary numeric computation and are accelerated by graphics processing units. These multidimensional structures can be operated on and manipulated with application program interfaces (APIs).
- ❖ **TorchScript** - This is PyTorch's production environment, which allows users to switch between modes with ease. TorchScript improves functionality, speed, usability, and flexibility.
- ❖ **Dynamic graph computation** - By using this feature, users can alter network behavior immediately, without having to wait for all the code to run.
- ❖ **Automatic differentiation** - Neural networks are built and trained using this method. By using neural network backtracking, it quantitatively calculates the derivative of a function.
- ❖ **Python support** - PyTorch may be used with well-known libraries and packages like NumPy, SciPy, Numba, and Cython because it is based on the Python programming language.
- ❖ **Variable** - The variable is enclosed outside the tensor to hold the gradient. It represents a node in a computational graph.
- ❖ **Parameter** - Parameters are wrapped around a variable. They're used when a parameter needs to be used as a tensor, which isn't possible when using a variable.
- ❖ **Module** - Modules represent neural networks and are the building blocks of stateful computation. A module can contain other modules and parameters.
- ❖ **Functions** - These are the relationships between two variables. Functions don't have memory to store any state or buffer and have no memory of their own [22].

3.6 Chapter Summary

This chapter provides an overview of the proposed system as well as information about the COCO dataset. This chapter includes the architecture of YOLOv5s model which is used as object detection and calculate the distance between camera and detected objects in this system and also involves the about of PyTorch which is apply as a running environment in this system.

CHAPTER 4

IMPLEMENTATION AND EXPERIMENTAL RESULTS

Software implementation is required to perform object detection and distance estimation tasks. Intel Celeron N3060 processor running at 1.60 GHz; 4.00 GB of RAM; 64-bit operating system is used.

4.1 Implementation of the System

The flowchart of the system is given in Figure 4.1.

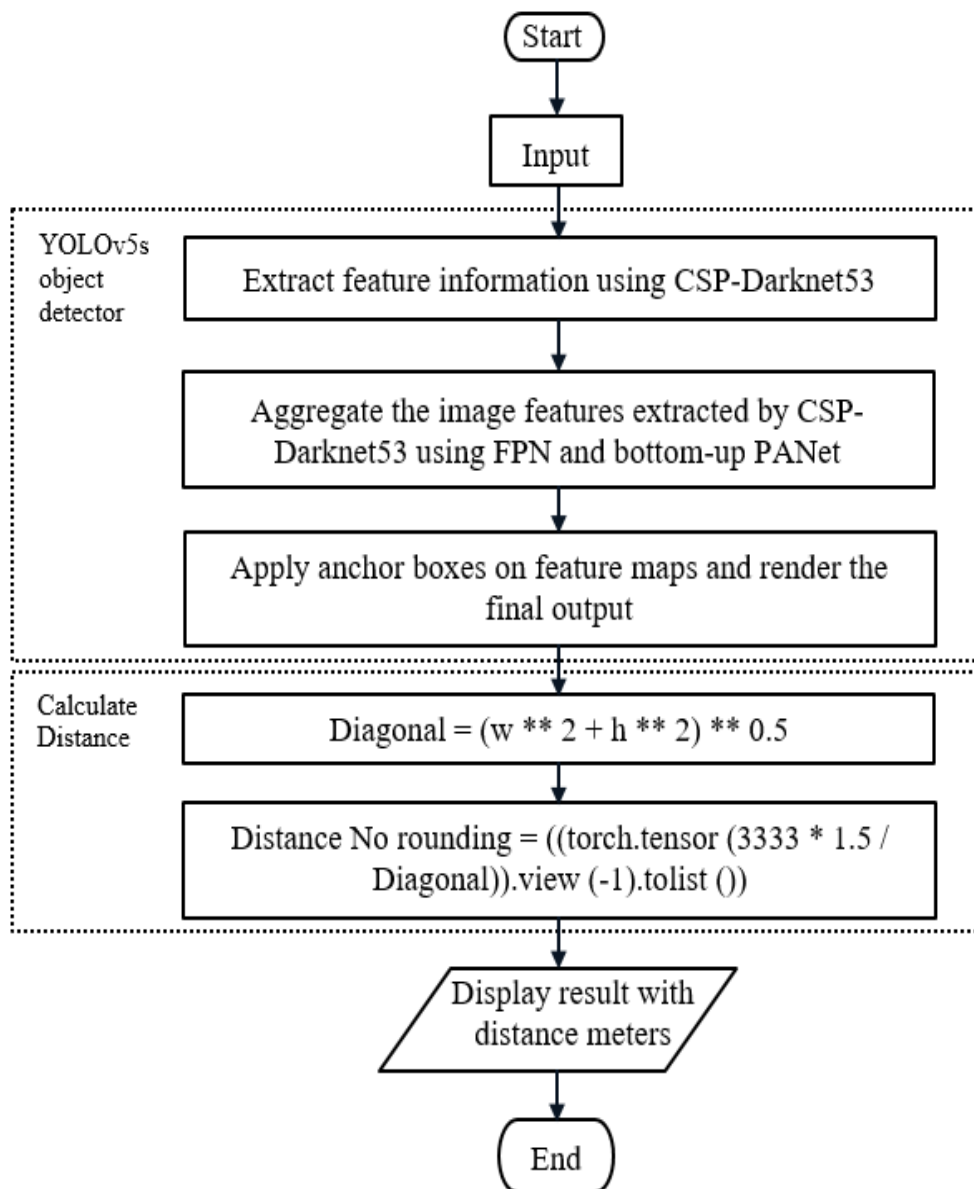


Figure 4.1 Flowchart of the System

To perform object detection, the camera captures the image or video, focus structure of the YOLOv5 model divides the input into the layers and then extracts the feature information using CSP-Darknet53 in the backbone layer. In the neck layer, aggregates the image features which are coming from the backbone layer using FPN and bottom-up PANet. Then, apply anchor boxes on feature maps and combine the outputs as the final result in the head layer. For calculate distance, uses the triangle formula to get the focal length. The values of bounding box width (w) and height (h) are get from YOLOv5s model. After that compute the distance meter between camera and detected object using focal length value in Torch.tensor metrics. Then, the final detected result will be with class labels and distance meters.

4.2 Experimental Results of the System

To detect image in OS, firstly, open the Window Command Prompt and need to run as administrator.

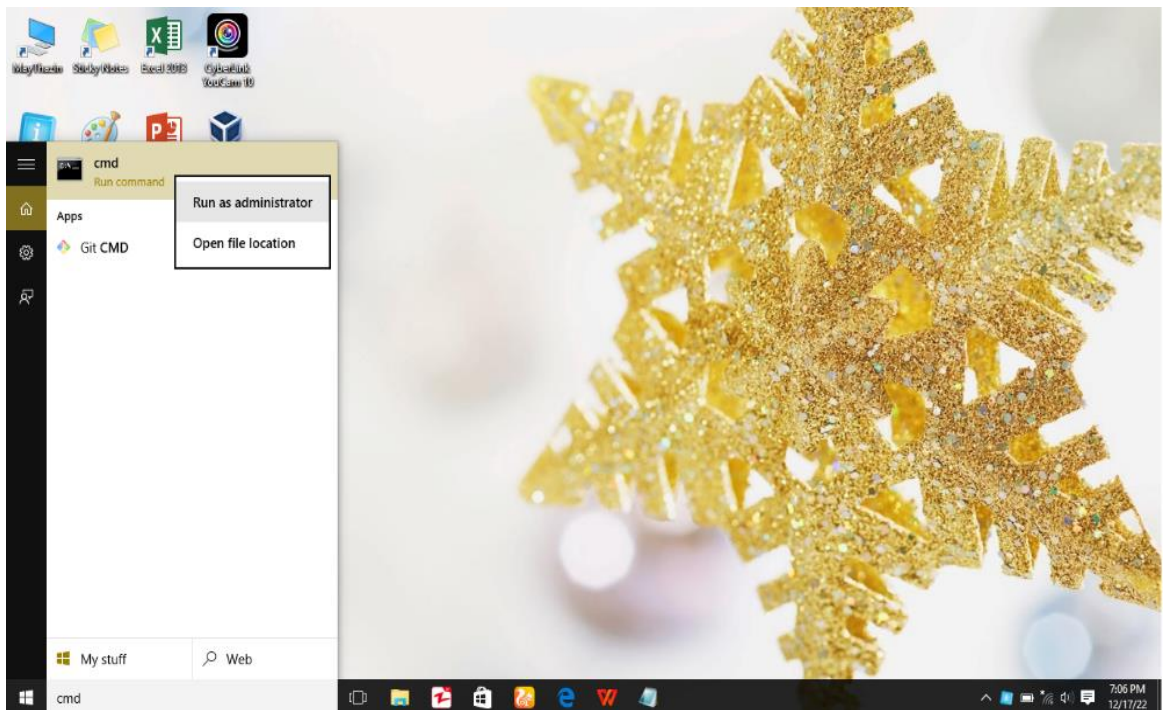


Figure 4.2 Run Window Command Prompt as Administrator

The following Figure 4.3 shown the testing image from the test file and the result is shown in Figure 4.4.

```

Administrator: Command Prompt
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>cd..

C:\Windows>cd..

C:\>cd oddey5

C:\ODDEY5>python detect.py --source data/images/test005.jpg
Detect: weights=yolov5s.pt, source=data/images/test005.jpg, data=data\coco128.yaml, imgsz=[640, 640], conf=0.25, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
YOLOv5 2022-4-24 torch 1.12.1+cpu CPU

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
C:\ODDEY5\detect.py:167: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.view(-1) rather than torch.tensor(sourceTensor).
  distance_No_rounding = ((torch.tensor(3333 * 1.5 / Diagonal)).view(-1).tolist()) #<class 'float'>
image 1/1 C:\ODDEY5\data\images\test005.jpg: 480x640 4 cars, 2 motorcycles, 1 truck, Done. (0.007s)
image 1/1 C:\ODDEY5\data\images\test005.jpg: 480x640 4 cars, 2 motorcycles, 1 truck, Done. (0.007s)
image 1/1 C:\ODDEY5\data\images\test005.jpg: 480x640 4 cars, 2 motorcycles, 1 truck, Done. (0.007s)
image 1/1 C:\ODDEY5\data\images\test005.jpg: 480x640 4 cars, 2 motorcycles, 1 truck, Done. (0.007s)
image 1/1 C:\ODDEY5\data\images\test005.jpg: 480x640 4 cars, 2 motorcycles, 1 truck, Done. (0.007s)
image 1/1 C:\ODDEY5\data\images\test005.jpg: 480x640 4 cars, 2 motorcycles, 1 truck, Done. (0.007s)
image 1/1 C:\ODDEY5\data\images\test005.jpg: 480x640 4 cars, 2 motorcycles, 1 truck, Done. (0.007s)
image 1/1 C:\ODDEY5\data\images\test005.jpg: 480x640 4 cars, 2 motorcycles, 1 truck, Done. (1.883s)
Speed: 7.0ms pre-process, 1883.1ms inference, 66.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs\detect\exp3

```

Figure 4.3 Detect and Save Detected Image File

In Figure 4.4, four types of object classes are detected and the distance meter is calculated. A bicycle, a bench, a person, and a potted plant have all been detected. The placement of the camera affects distance estimation measurement. For example, one bicycle's distance is 24.6m, while a person's distance is 40.5m.

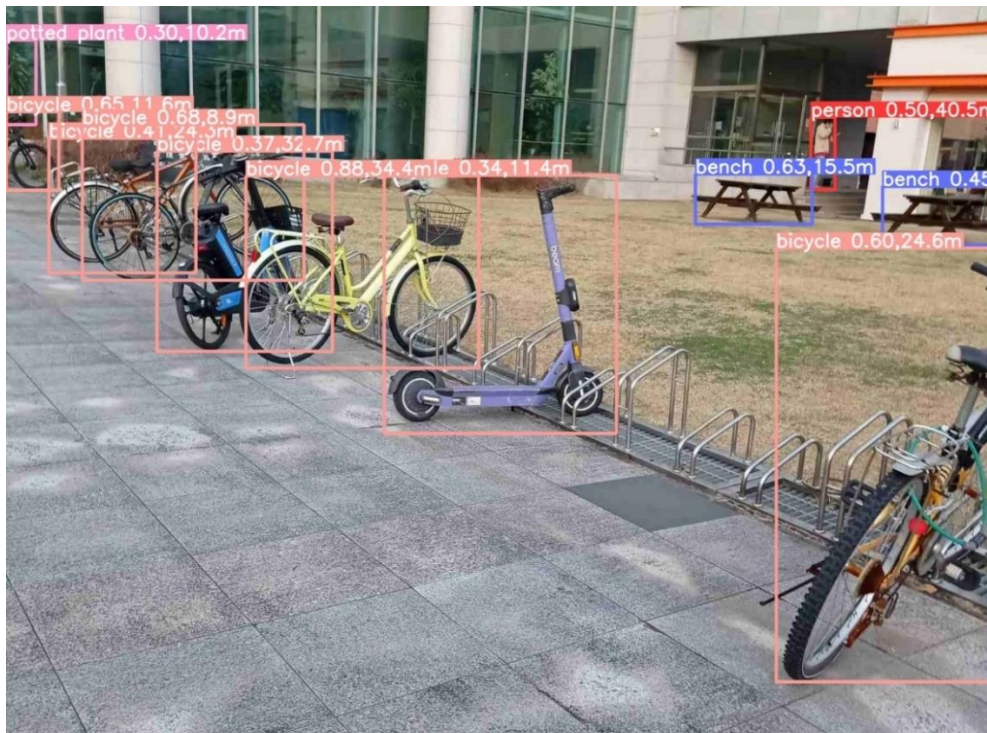


Figure 4.4 Detected Image Result 1

Five classes of objects are found in Figure 4.5. Person, chair, laptop, cup, and dining table are among them. The largest person distance, as seen in figure 4.5, is 33.8 m.

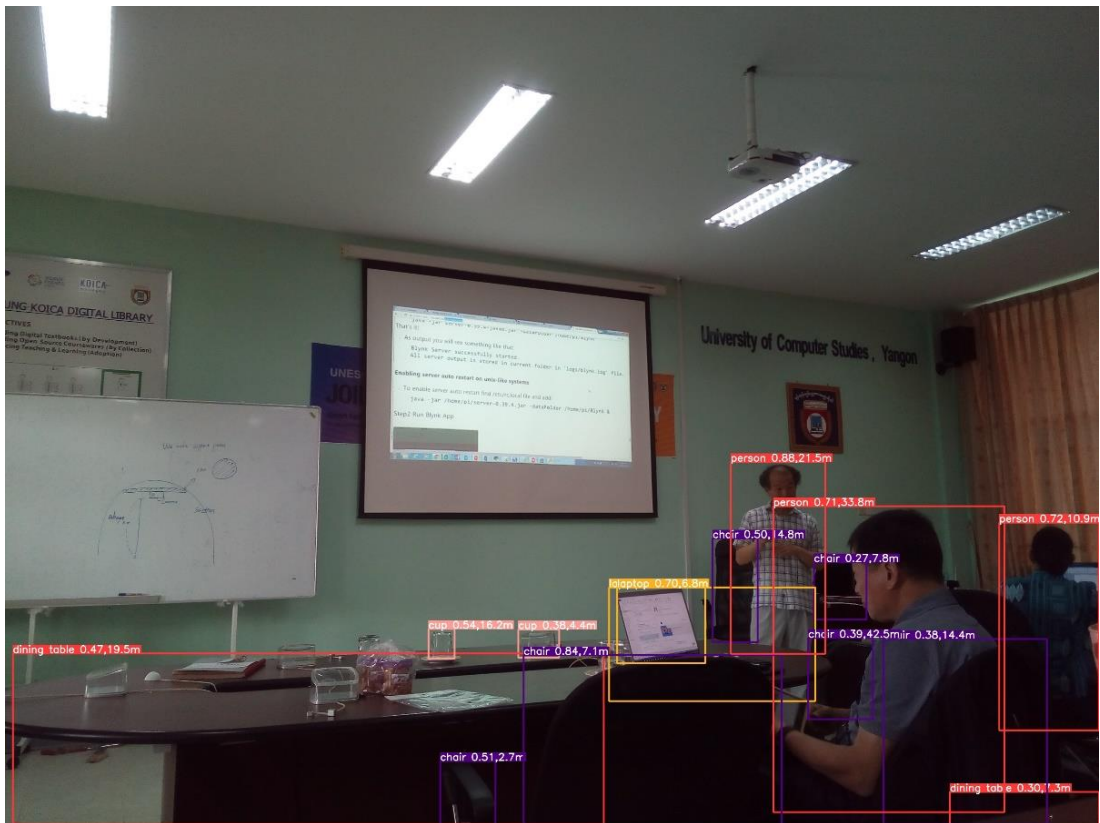


Figure 4.5 Detected Image Result 2

Figure 4.6 (a), (b) and (c) are the sample detected results of the video. Car and truck are detected in result (a) and (b). In the result (a), the truck's closest distance is 12.3 meters, whereas the car's distance is 47.6 meters.

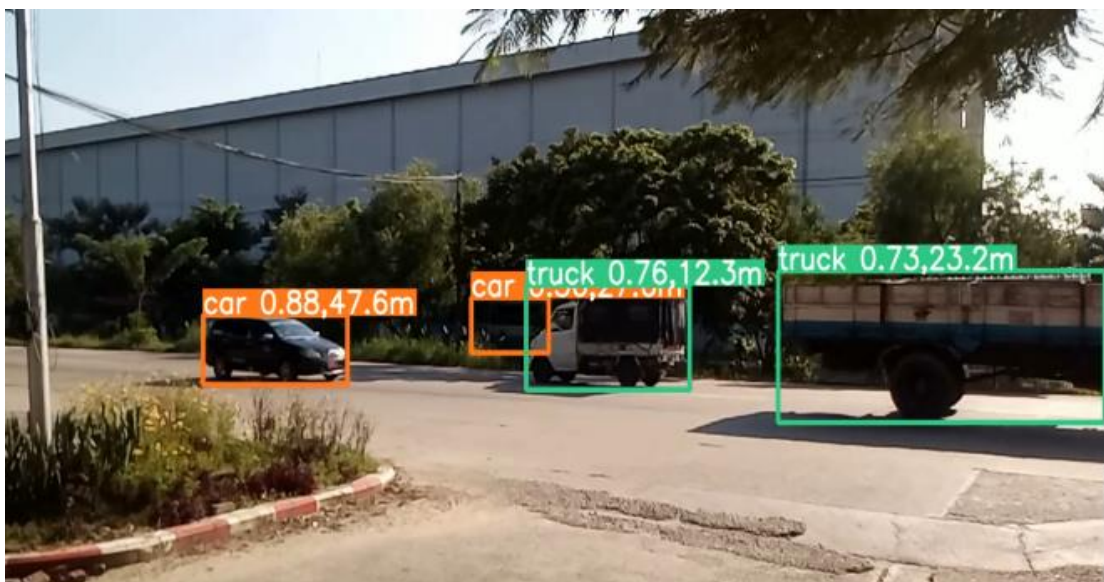


Figure 4.6 Detected Video Result (a)

The truck's closest distance is 7.7 meters, whereas the car's distance is 48.3 meters in the result (b).



Figure 4.6 Detected Video Result (b)

In Figure (c), a car, a truck, and an umbrella are detected. The distances between a car, an umbrella, and a truck are 58.3 meters, 21.6 meters, and 27.0 meters, respectively.

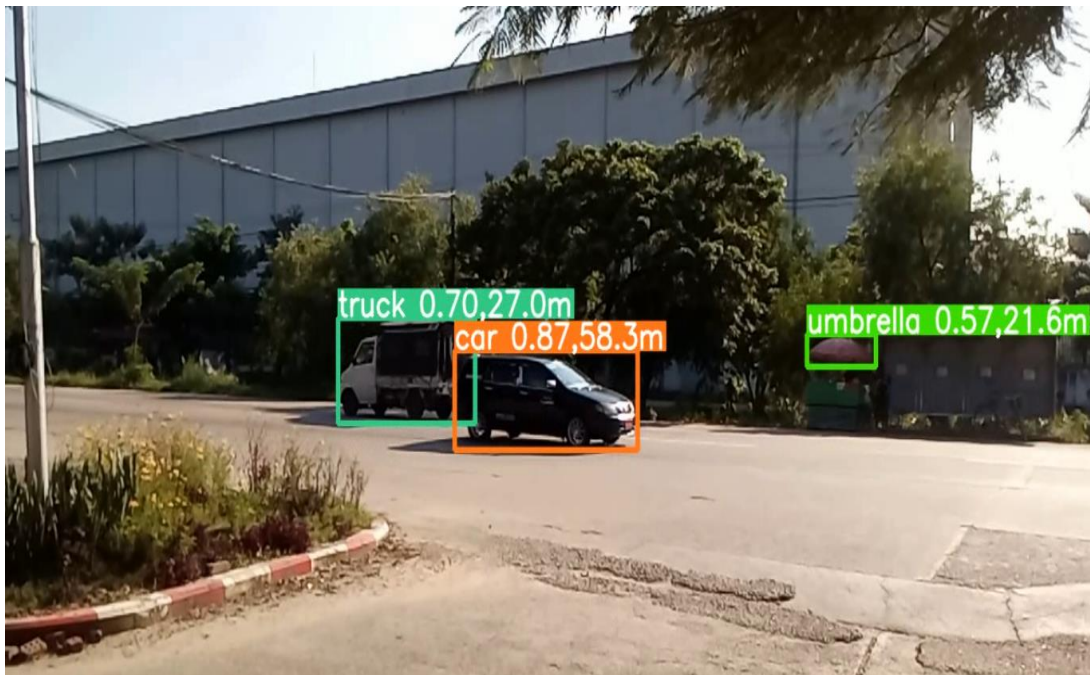


Figure 4.6 Detected Video Result (c)

4.3 Performance Evaluation for the System

Determine the precision values of the test set to learn about its performance. 800 images are used to calculate the system's precision recall values for the test set. The precision values of the test set are displayed in the figures below. A test set of 10 object classes was generated in order to better understand the precision values but only 9 object classes are shown in the report of Figure 4.7 and 4.8. When the confidence score is 0.915 for test set 1, the precision values for all classes will be 1 in Figure 4.7. Because there are no such objects in test set 1, the precision values for person, car, bus, train, truck, and traffic Light in test 1 are 0.000. The motorcycle's precision value is 0.028, the airplane's is 0.022, and the boat's is 0.035.

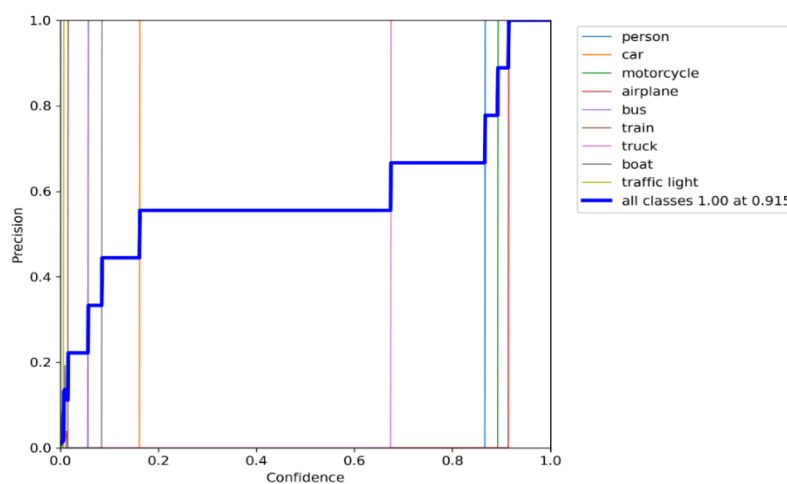


Figure 4.7 Precision Values of Test Set 1

In Figure 4.8, the precision values for all classes are 1 and the confidence score is 0.696 for the test set 2. Figure 4.10 depicts the detail precision values for each object in the test set 2.

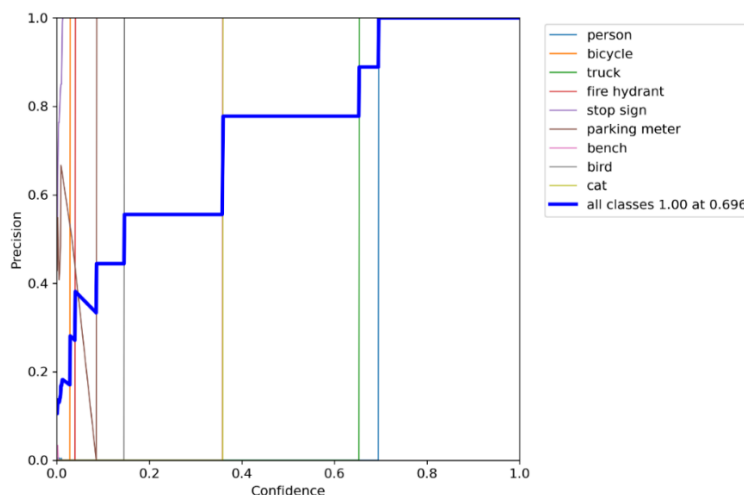


Figure 4.8 Precision Values of Test Set 2

When all precision values from all classes are added together, the result is illustrated in Figure 4.9. Figure 4.9 shows that when the confidence value is 0.962, all classes are 1.

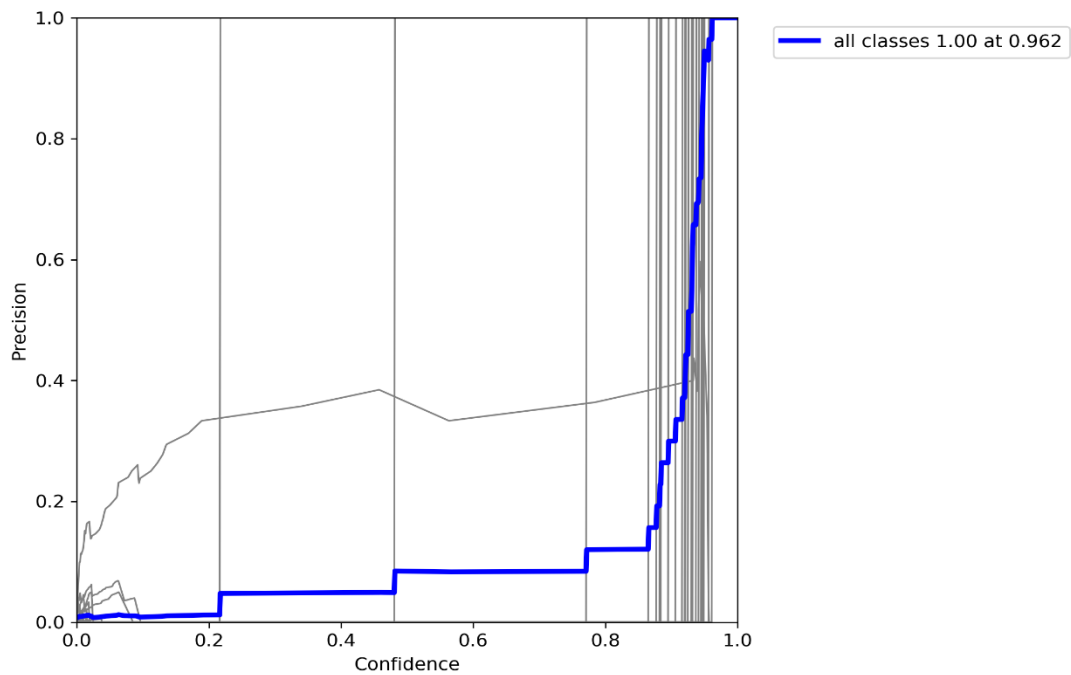


Figure 4.9 Precision Values of All Test Set

Precision Recall curves for the test sets are shown in Figure 4.10, 4.11 and 4.12 respectively.

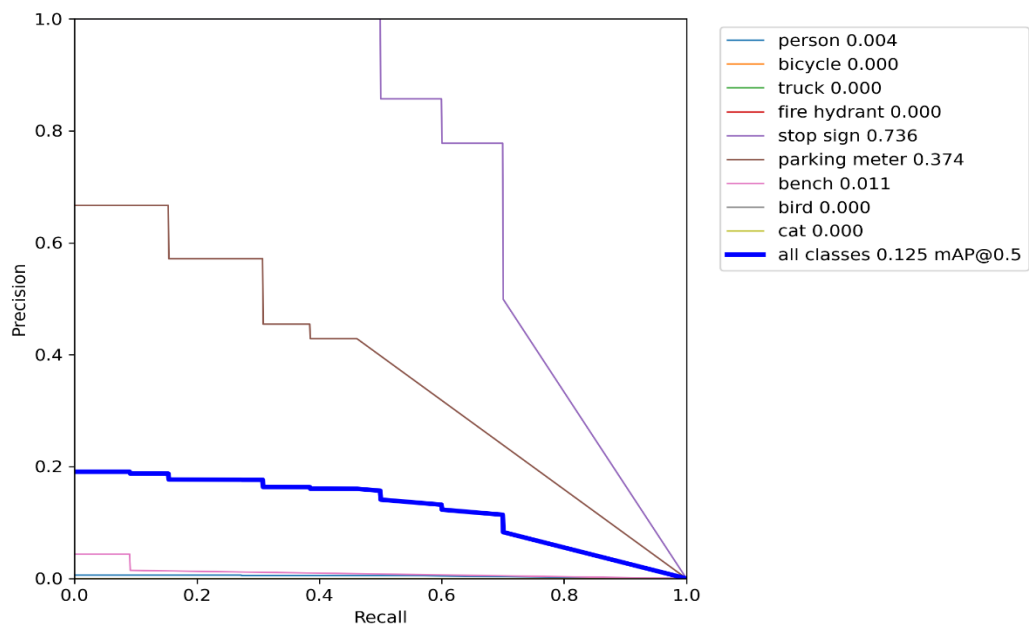


Figure 4.10 Precision Recall Curve of the Test Set 1

Figure 4.11 shows that the precision recall values of bicycle, airplane, truck, and boat are all 0.000, while the other five classes are 0.030, 0.161, 0.019, 0.378, and 0.028, respectively.

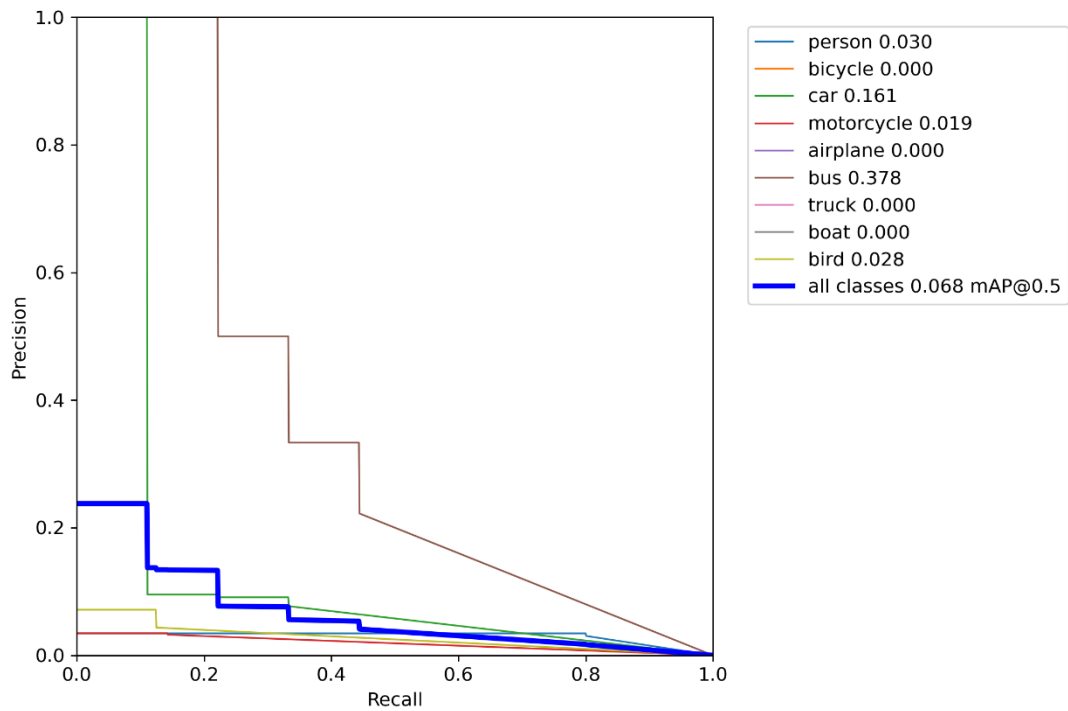


Figure 4.11 Precision Recall Curve of the Test Set 2

However, when all precision values from all classes are added together, the result is not clear and will be shown in Figure 4.12.

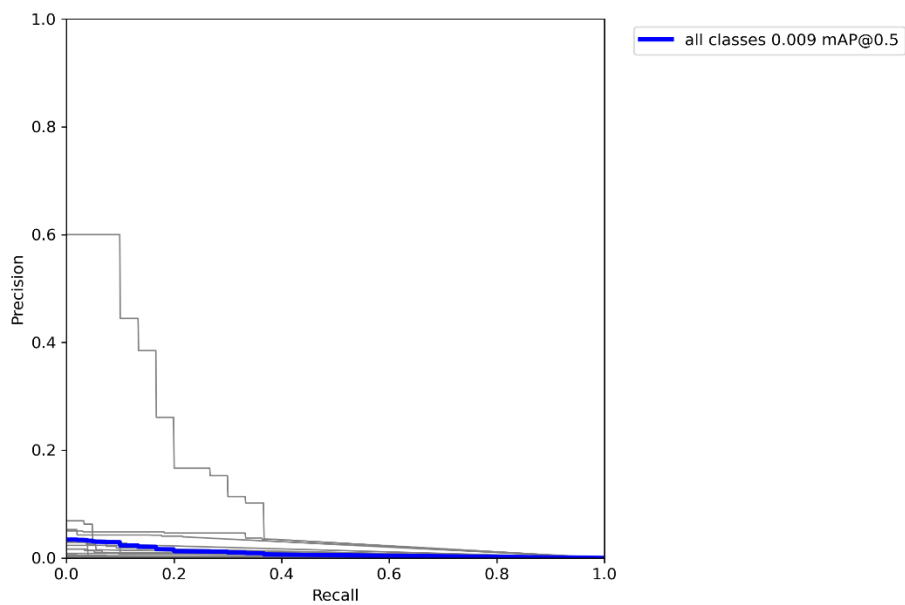


Figure 4.12 Precision Recall Curve of All Test Set

4.4 Chapter Summary

This chapter describes the system's flowchart and experimental results when two or more classes are detected in an image or video. For the test sets, it also includes precision values and precision recall curves.

CHAPTER 5

CONCLUSION

In summary, AI based computer vision tasks are become popular in nowadays. Object detection is one of the famous computer vision task to apply in various applications such as healthcare, security surveillance and self-driving cars. Distance estimation is combined together with object detection so that to improve driving system of AI based autonomous vehicles and remote controls. YOLOv5s is most suitable for real-time detecting objects and measure the distance between camera and its detected objects because of its higher performance and accuracy.

The object detector YOLOv5 is used to calculate the distance between the camera and the detected objects. Backbone, Neck, and Head are the three layers of YOLOv5. The backbone layer extracts feature information using CSP-Darknet53, and the neck layer aggregates the image features extracted by CSP-Darknet53 using FPN and bottom-up PANet. Then, on feature maps, apply anchor boxes and render the final output. To calculate the distance meter of detected objects between cameras, use focal length values obtained from the triangle formula and combine them with the width and height of the bounding box obtained from the object detector and then sort into the torch. Tensor metric.

5.1 Advantages

This system is only base on the single camera of the device. Therefore, it can easily be used in operation system and no need to install other external devices such as Light Detection and Ranging (LiDAR) sensor or other cameras. This system can be detected various classes of objects.

5.2 Limitation

Distance meter values can vary according to the focal length because distance calculating formula is based on the device camera's focal length. The more cameras lens are better distance meter values are more accurate.

5.3 Related Works

In the paper [2] uses the YOLOv3 to predict the absolute distance of objects using only information from a monocular camera and design the two ways of measuring the distance, class-agnostic and class-aware. Class-agnostic creates smaller prediction vectors than class-aware and achieves better results. In this paper, KITTI dataset is used and show the distance range within [0, 150] m. Uses cameras instead of LIDARs to present the possibility for distance estimation in the paper [21]. This paper is based on the YOLOv3 deep neural network and principles of stereoscopy. In this paper, uses two slightly moved cameras to get two pictures which goes through algorithm for stereoscopy-based measurement and estimate distance to detected objects. In the paper [1] uses the YOLOv5 model to measure the distance between objects for processing real-time images with OpenCV in order to restrict the distance between several people in the same space and also add Euclidean distance calculation method in DeepSORT and OpenCV to minimize occlusion. Detecting the distance between people and using the open-source COCO dataset for learning in this paper.

5.4 Further Extension

The system functions can be added distance accuracy correction. When the distance is too close, a warning message will be issued. Another extension is not only can add emoticons but also vehicle speed per hour and target object speed per hour. If one of the speed per hour of the vehicle and the speed of the target exceeds the upper limit, an early warning will be given.

REFERENCES

- [1] A study on object distance measurement using OpenCV-based YOLOv5, International Journal of Advanced Culture Technology, Vol.9 No.3 298-304 (2021), DOI.
- [2] Dist-YOLO: Fast Object Detection with Distance Estimation, Appl. Sci. 2022, 12, 1354.<https://doi.org/10.3390/app12031354> by Marek Vajgl, Petr Hurtik and Tomáš Nejezchleba
- [3] Gochoo, M. (2020). ReseachGate. Search date 03.12.2020. researchgate.net: https://www.researchgate.net/figure/a-Feature-pyramid-network-FPN-b-YOLO3-c-Proposed-concatenated-feature-pyramid_fig2_335538302
- [4] Huang, G., Liu, Z., & Maaten, L. v. (2018). Densely Connected Convolutional Networks. arXiv. Seach date 28.11.2020. <https://arxiv.org/pdf/1608.06993.pdf>
- [5] Hui, J. (2020). YOLOv4. Medium. Seach date 27.11.2020. <https://jonathan-hui.medium.com/yolov4-c9901eaa8e61>
- [6] Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path Aggregation Network for Instance Segmentation. arXiv. Seach date 03.12.2020. <https://arxiv.org/pdf/1803.01534.pdf>.
- [7] Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. arXiv. Seach date 17.11.2020. <https://arxiv.org/pdf/1612.08242.pdf>
- [8] Solawetz, J. (2020). Breaking Down YOLOv4. Roboflow. Seach date 27.11.2020. <https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/>
- [9] V Thatte, A. (2020). Evolution of YOLO — YOLO version 1. Medium. Seach date 14.11.2020. <https://towardsdatascience.com/evolution-of-yolo-yolo-version-1-afb8af302bd2>

- [10] Wang, C.-Y., Mark Liao, H.-Y., Yeh, I.-H., Wu, Y.-H., Chen, P.-Y., & Hsieh, J.-W. (2019). CSPNET: A new backbone that can enhance learning capability of CNN. arXiv. Search date 30.11.2020. <https://arxiv.org/pdf/1911.11929.pdf>
- [11] YOLO Multi-Camera Object Detection and Distance Estimation, DOI: 10.1109/ZINC50678.2020.9161805 by Bojan Strbac, Marko Gostovic, Zeljko Lukac and Dragan Samardzija
- [12] Zero-Shot Pipeline Detection for Sub-Bottom Profiler Data Based on Imaging Principles
- [13] <https://blog.superannotate.com/introduction-to-computer-vision/>
- [14] <https://docs.ultralytics.com/tutorials/architecture-summary/>
- [15] <https://github.com/o920130130/YangSongbo/pulls>
- [16] <https://github.com/rafaelpadilla/Object-Detection-Metrics>
- [17] <https://iq.opengenus.org/yolov5/>
- [18] <https://machinelearningknowledge.ai/a-brief-history-of-yolo-object-detection-models/>
- [19] <https://medium.com/axinc-ai/map-evaluation-metric-of-object-detection-model-dd20e2dc2472>
- [20] <https://paperwithcode.com/task/object-detection>
- [21] <https://www.javatpoint.com/working-of-convolutional-neural-network-tensorflow>
- [22] <https://www.techtarget.com/searchenterpriseai/definition/PyTorch>

LIST OF PUBLICATIONS

- [1] May Thu Aung, Khaing Khaing Wai, “Object Detection Along With Distance Estimation Using YOLOv5 Model”, University of Computer Studies, Yangon, Myanmar, 2022.