REAL-TIME IMAGE PROCESSING FOR PRODUCTIVITY TRACKING

BY

TITH VONG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (ENGINEERING AND TECHNOLOGY)

SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

THAMMASAT UNIVERSITY

ACADEMIC YEAR 2022

THAMMASAT UNIVERSITY
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY

THESIS

BY

TITH VONG

ENTITLED

REAL-TIME IMAGE PROCESSING
FOR PRODUCTIVITY TRACKING

was approved as partial fulfillment of the requirements for
the degree of Master of Science (Engineering and Technology)

on November 17, 2022

Chairperson [illegible]

(Associate Professor Charnchai Pluempitiwiriyawej, Ph.D.)

Member and Advisor [illegible]

(Associate Professor Chawalit Jeenanunta, Ph.D.)

Member Warut P.

(Assistant Professor Warut Pannakkong, Ph.D.)

Director [illegible]

(Professor Pruettha Nanakorn, D.Eng.)

Thesis Title	REAL-TIME IMAGE PROCESSING FOR PRODUCTIVITY TRACKING
Author	Tith Vong
Degree	Master of Science (Engineering and Technology)
Faculty/University	Sirindhorn International Institute of Technology/ Thammasat University
Thesis Advisor	Associate Professor Chawalit Jeenanunta, Ph.D.
Academic Years	2022

ABSTRACT

The production planning manager could not keep track of real-time data in the production line. They could not identify the issue until a few days later. The problem in the production line can make products produce less or more than demand. It makes the company waste time and money. The worker counted the shoe without any timestamp and noted it on the paper note, which is unmanageable and easy to lose or damage. This proposal proposes shoe counting with a timestamp to help work count products automatically. We use python programming language and OpenCV2, object training by YOLOv4-tiny, to make the system recognize the object “shoe.” Then, the detected shoes from camera streaming will count by using a 2-point intersection of each center point algorithm. There are 106 images selected among 40 shoe models to train the approach, and then images will separate into 80 training images, 16 validation images, and 10 testing images. The 10-fold algorithms are applied to a 10-times shuffle of training, validation, and testing images. This application evaluates counting and timestamp accuracy by comparing manual and system counting reports. After evaluation, our system achieves 99 percent of shoe counting, then 80 percent of timestamp extraction, and it can correctly detect and count another untrained shoe model. This approach reveals that our proposed method is suitable for counting objects in real-time.

Keywords: Object counting with timestamps, 2-points intersection, Real-time object counting, 10-fold algorithms, *YOLOv4-tiny*.

ACKNOWLEDGEMENTS

We appreciate all CPL (Safety shoe industry Group) and Smart Sense Company’s staff and CEO for their corporation by providing proper camera streaming data from the industry’s floor plan. This approach is mainly supported by the *Centre of Excellence in Logistics and Supply Chain Systems Engineering and Technology* (*CoE LogEn*), *Sirindhorn International Institute of Technology, Thammasat University*. All of these will be hard to achieve without the involvement of all committee members.

Tith Vong

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	**Terms**
MQTT	Message Queue Telemetry Transport
VDO	Video
RSTP	Rapid Spinning Tree Protocol
YOLO	You Only Look Once
GPU	Graphics Processing Unit
CPU	Central Processing Unit
CSP	Cross-State-Partial
RPN	Region Proposal Network
FPS	Frame Per Second
OT	Over-Time
IP	Internet Protocol network
PN	Positive and Negative
API	Application Programming Interface
CNN	Convolution Neural Network
DNN	Deep Neural Network
ID	Identification

CHAPTER 1
INTRODUCTION

Image Processing is one of the leading marketing technologies used to help humans in many fields such as medicine, industrial surveillant, military aerospace, etc. Nowadays, many industries have applied Image Processing to support their work. It assists the firm in saving time and money, improves working more standardized, and improves the product's quality. Many industries in Thailand keep updating their company from industrial version 2.0 to automation with the help of an AI control system. As our shoes industrial case study, the production process still uses a paper note, and the planner gets the updated report the next day.

1.1 General Statement of Image Processing and its Benefit

Image processing plays an essential role in this automation world of object recognition. The speed and accuracy of real-time object detection are significantly improved. Technically, image processing has been applied for many purposes, such as military, medical study, robots, vision control, remote sensing, etc. In many fields, object detection frameworks have been practiced for different objectives. Related to the object counting concept, we notify that it has been applied to count various objects such as humans, cars, cells, rocks, and other products by using image processing. In this proposed application, YOLOv4-tiny will introduce to train our object "shoe."

Based on Bochkovskiy, Wang, and Liao (2020), the low-budget company could not afford a high computation power for a real-time object detector with a high frame rate. Real-time object counting becomes more critical to an embedded system for helping humans reduce their tasks. Ren, Wang, Fang, Song, and Djahel (2020) used a YOLO framework with a fire layer of Squee to measure humans in the subway. Doan and Truong (2020) applied *YOLOv4-tiny* to count vehicles on the countryside's roads. Even though many image-counting approaches include a human walkthrough, cars, and products, giving the exact timestamp when the object is detected using a low-computation device is still a hot topic to discuss. The model should be flexible and consistent enough to reduce the complexity and standardize the working behavior of the worker.

1.2 Existing Model Analyze

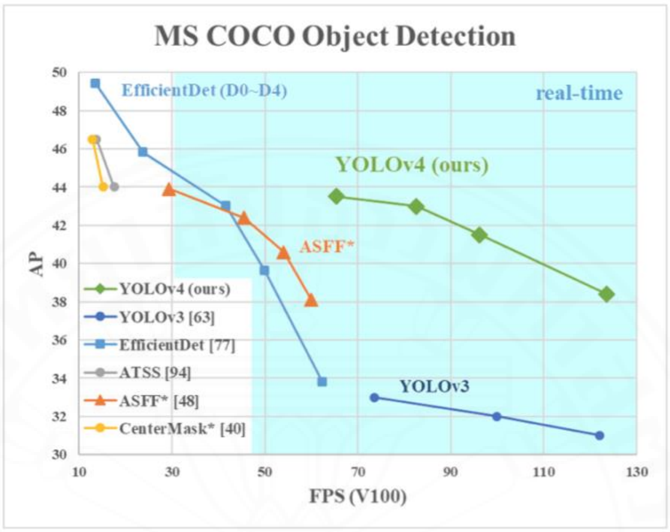


Figure 1.1 Machine Learning Model Comparison

To choose the best suitable and compatible technology with our application, we need to test every possible asset with a higher outcome during the neural network model comparison of ***EfficientDet***, ***RetinaNet***, ***ATSS***, ***ASFF,*** etc. We identified that each machine learning model has its weak and robust points depending on the application of its specialist.

➔ Below part is an explanation of each benefit of the applied method:

- ***EfficientDet*** needs more resources to scale up the accuracy. At the same time, ***Yolo-V4 Tiny*** was designed to produce higher accuracy, faster detection, and low resource consumption, which is also more appropriate for real-time detection.
- ***RetinaNet*** usually is applied with a single-stage detection which consumes fewer resources but low-accuracy detection level.
- ***ATSS*** stands for Adaptive Training Sample Selection, and it selects positive and negative sampling based on the characteristic of the object.
- ***ASFF*** comes from the word Adaptively Spatial Feature Fusion is used to separate the different scales of the input image as a pyramid structure, and it is also

applied to the single-shot detector, which is not the most suitable with our real-time application.

1.3 Purpose of Application

Our goal is to design a low computation real-time image counting system with a timestamp for a shoe manufacturer to help them enhance their production efficiency with fewer counting errors and visualize the real-time data update. The problem in the production line can make shoe production unstable to the production plan. The planner and project manager would like to know the detail of the production flow in the factory. Then they can identify the issue more clearly (Intalar, Chumnumporn, Jeenanuta, & Tunpan, 2021). Therefore, this proposal will propose an MQTT protocol to send real-time counting shoes with timestamps to the planner. The workflows are divided into four main steps: choose a suitable image covering 40 types of shoes and then legal theirs's data. We choose the best training dataset using the 10-folds technique to shuffle the total dataset into 10 different data sets. Then, the Convolution network model-29 and darknet-23 framework were applied for each collection of shuffle data. Next, we compared and chose the best result with the precision and recall value of each training—the outputs of configuration files and hyperparameter optimization with OpenCV2 and a python code. Our counting algorithms will convert to coding, and then we test it with manually recorded Video first. Next, we evaluate the proposed system in real-time with timestamps whenever the shoes pass the counting line. The system will publish shoe counts with a timestamp message using the MQTT protocol with a minicomputer called jetson nano for every detected shoe triggered whenever the shoe moves across the counting line.

This thesis report is divided into the following parts: Abstraction, Introduction, and understanding of the case study problem statement. Real-time object detection technique with a server's timestamp then illustrated Literature review for their method improvement and limitations. Next, the workflow of real-time object counting with low computation power. This report will also explain deeply related to the technique staff, such as flow chart, pseudo code, and general system physical architecture.

CHAPTER 2
LITERATURE REVIEW

Image processing technology has been significantly updated for a decade. Many open-source frameworks are used for object detection, such as *EfficientDet*, *YOLOv4, YOLOv5, ATSS, ASFF*, etc.

2.1 Introduction to YOLO

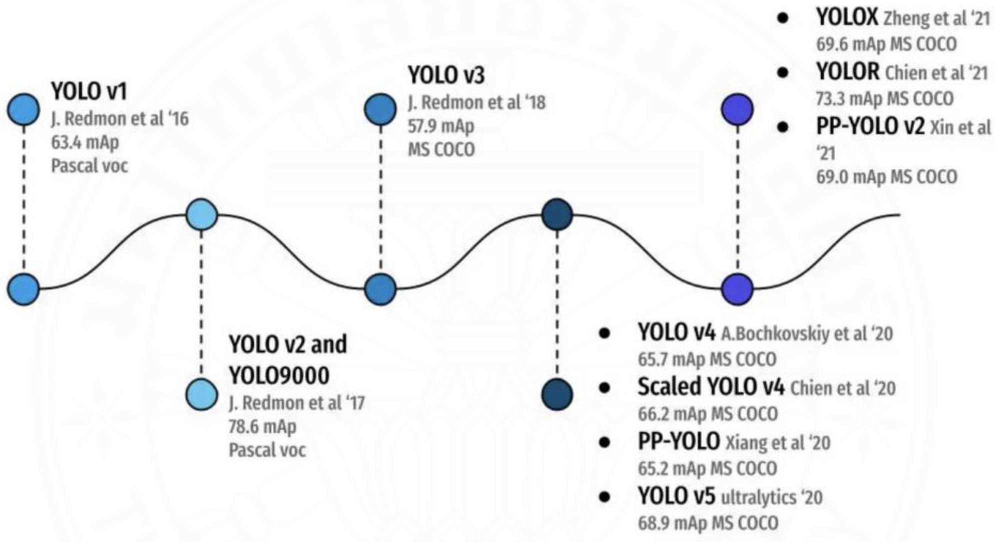


Figure 2.1 Yolo Family Timeline

YOLO (You only look once) was introduced in 2016 for real-time object detection with high speed and accuracy with high frame rate video. YOLO separate into 5 versions, and it is proposed from 2015 to 2020. The first version of YOLO was presented by Redmond et al. in 2015 to detect a real-time object. The authors used end-to-end neural networks to classify detection platforms in 2016. Redmon and Farhadi (2016) also proposed YOLO9000 to fix the main problem using Darknet-19 to detect smaller objects with more accurate localization. Redmon and Farhadi (2018) enhanced the model and convolutional network of YOLOv3 with an incremental improvement of 106 layers on CSPDarknet-53. A Year After, Jocher, Stoken, and Borove (2020) developed the YOLOv4 model to increase the speed and accuracy of real-time object detection with a higher frame rate of up to 65 FPS. YOLOv4-tiny was also introduced to detect real-time objects with a higher frame rate suitable for low computation power (Bochkovskiy, Wang, & Liao, 2020).

2.2 Related Work

Ren, Wang, Fang, Song, and Djahel (2020) used a fire layer of Squee to create more bounding boxes, and the counting technique moved left out of the frame to count people in the subway. Doan and Truong (2020) proposed their approach with YOLOv4-tiny to count vehicles on countryside roads using DeepSort algorithms to define new objects with the new Identity tracking that appeared in the frame. Covavisaruch and Saengpanit (2004) applied Coarse Detection to detect timestamps with a bounding box and the Fine Detection technique to locate each digit of a timestamp in that frame. Covavisaruch and Saengpanit (2004) delete background portions in the video clip for timestamp extraction.

2.3 Yolov4 Object Detection and Its Framework with Timestamp

This proposed model obtained a good result for the loss function. Meanwhile, Alsanabani, Saeed, Al-Mkhlafi, and Albishari (2021) mentioned that the convolutional layer’s deduction in the backbone and replaced residual-block to CSP block layer to increase detection accuracy. Zhe Li et al. used time annotation to train a segmentation model to predict action labels for every frame in the Video. Li, Abu Farha, and Gall (2021) also proposed frame-wise labels to solve the limitation of frame annotation. The multi-stage temporal convolutional network is a segmentation model that applies parallel stages for the first stage with 5 kernel-size and 3 kernel-size for other stages (Li, Abu Farha, & Gall, 2021). Yang, Xie, and Qu (2021) designed a model improvement of YOLOv4 called Aircraft-YOLOv4. The authors used a large-scale network and RPN (Region Proposal Network) with UCAS-AOD for Aircraft real-time object detection (Yang, Xie, & Qu, 2021). The authors attempted to change the residual structure in the backbone. Mathematic equations are applied with deep separatable convolution to improve the performance of real-time detection accuracy (Yang, Xie, & Qu, 2021). Jiang, Zhao, Li, and Jia (2020) used the ResBlock-D module in its network instead of CSPBlock modules for computation complexity deduction. Alsanabani, Saeed, Al-Mkhlafi, and Albishari (2021) illustrated that the CSPBlock module could provide more detection accuracy; it may reduce detection speed due to the complex network. The author increases accuracy by adding “residual network blocks” into ResBlock-D. Biradar, Gupta, Mandal, and Vipparthi (2019) applied a two-stage technique for timestamp anomaly detection. The author used this proposed system to detect a vehicle with a timestamp in a traffic light from a video clip (Biradar, Gupta, Mandal, & Vipparthi, 2019).

CHAPTER 3
PROPOSE SOLUTIONS & ALGORITHMS

3.1 Company's Floor Plan Study

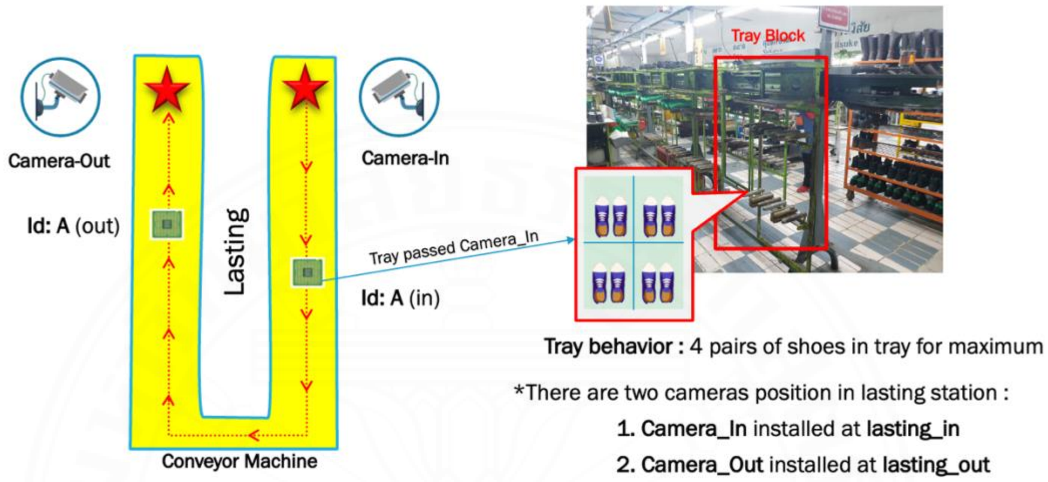

Camera-Out
Id: A (out)
Lasting
Camera-In
Tray passed Camera_In
Id: A (in)
Conveyor Machine
Tray Block
Tray behavior : 4 pairs of shoes in tray for maximum
*There are two cameras position in lasting station :
1. Camera_In installed at lasting_in
2. Camera_Out installed at lasting_out

Figure 3.1 Station Case-study Process Explanation

During the data collection stage, we need to understand the correct part of the process workflow inside the company. The Figure above explains how the shoes are produced in that station; the machine's behavior is passing the shoe on each tray through each process from the input to the output. The input point is *lasting_in*, and the output point is *lasting_out*. In the **Lasting station**, we decided to install two cameras at the position in and out, as shown as *camera_in* and *camera_out*. We can place a maximum of 4 pairs of shoes on each tray. Our purpose is to count the number of shoes that get in and get out with the timestamp. This counting method with timestamp can help us visualize the product flow of the cycle and flow time of each shoe pair.

3.2 Problem Statement

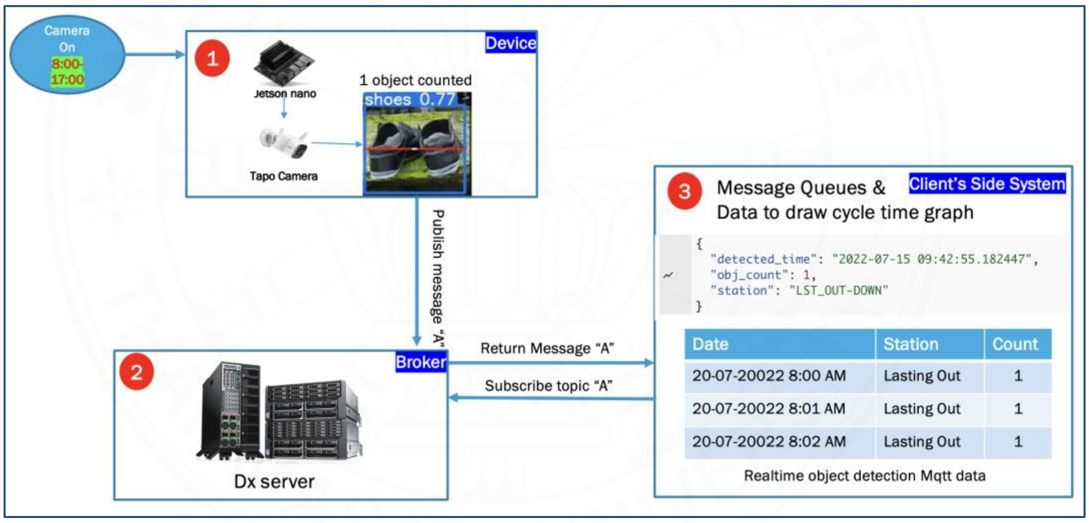
Inside the case study's factory, workers typically count shoes by noticing down on paper from one process to another and submitting reports later to the plan manager. Sometimes, workers miss counting caused of the complexity of the task, and it requires at least two staff; one staff work on the shoe process, and the other one count and note down the shoe. Using a paper form, sometimes the report is easy to lose or damage.

The plant manager cannot visualize the production flow in real time because the information hasn’t been updated on time, and there is no timestamp record. It takes more time to calculate that report. In these cases, the production will be more delayed, making customers unsatisfied.

3.3 System Architecture and Algorithms

A system architecture can help us understand the whole process of the designed system. We can also identify the required flow of each use case by preventing the cost of complexity in the organization. A transparent system design needs a good architecture design.

3.3.1 General Physical Architecture



```
{
  "detected_time": "2022-07-15 09:42:55.182447",
  "obj_count": 1,
  "station": "LST_OUT-DOWN"
}
```

Date	Station	Count
20-07-20022 8:00 AM	Lasting Out	1
20-07-20022 8:01 AM	Lasting Out	1
20-07-20022 8:02 AM	Lasting Out	1

Figure 3.2 Shoe Detection Event Trigger

- (1) is used to connect to the broker and notify (2) when shoes are count
- (2) is a host broker; it publishes messages of counted shoes to the client
- (3) is the application platform that shows data as a graph or table to the user

**Time interval of the case study station starts from 8:00 AM until 5:00 PM*

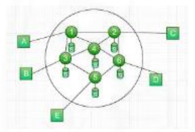
The camera has been set to work and recording at 8:00 AM daily. After receiving the data, our proposed program will execute the startup service of our counting application, which will count every shoe that passes the counting line. Then it will publish a message to the broker, and next, the broker needs to host the broadcast message separately by topic name. Whenever the client subscribes correctly to follow the topic published by the broker, the client will receive a kind of data as shown in (3). This process will repeatedly send and receive until the system is offline at 5:00 PM.

3.3.2 System Model design

1. Technology Usage definition:

At this point, we will get to know the technology part in the context of why we chose that type of that platform to apply in our proposed system.

Table 3.1 List of Model Technology Chosen

Technology Icon	Name	Definition	Apply For
	RoboFlow	It is a tool used for computer vision tasks	Object labeling platform
	10-Folds Algorithms	It is a cross-validation process of best data selection	Data shuffling of 10 different set
	Yolo	You Only Look Once	Object Training Technique
	Darknet Framework	It is an open-source neural network framework	Fast object training computation with GPU
OpenCV	Python, OpenCV	Is an interpreter, OOP programming language	DNN network for object detection and counting program
Tesseract OCR	Tesseract OCR	It is a character recognition engine under the Apache license	Video clip's timestamp extraction
	Jetson nano	It is a single mini-board computer	Onside object counting computation

NVIDIA CUDA	CUDA GPU	Graphic Processing Unit	Helping computation faster

2. Detail Workflow:

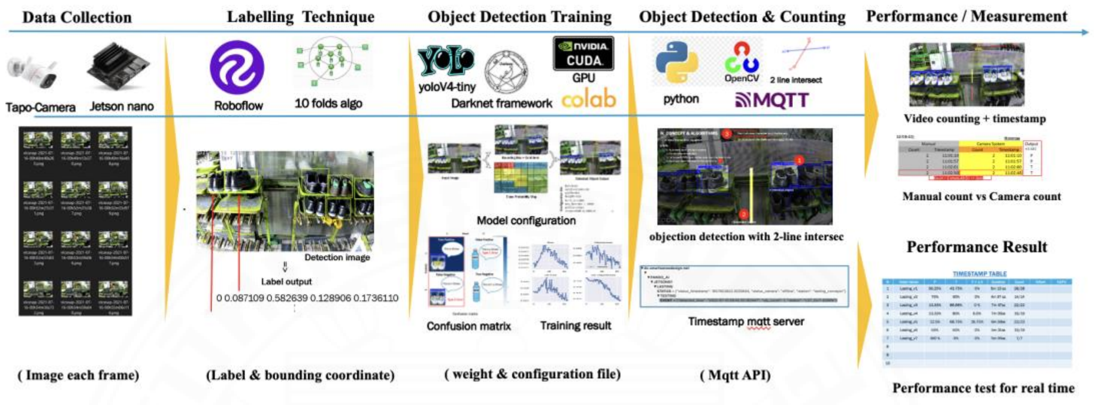


Figure 3.3 Model Design Workflow

The architecture model workflow of a system played an essential role in giving a specific purpose of the research with a road map on the technical study and starting from collecting data from the company's floor plan with the help of *openAPI* and *rstp* after we installed the camera directly onside. Then, we can access the streaming data. Initially, we need to record some vital video that summarizes all types of shoes. After that, we use the *roboflow* web platform to label our image with the specific class coordinate. Additionally, we use 10-fold algorithms to shuffle and divide 1 set of data into ten groups. Next, we applied each collection of the shuffle data to train by using a *yolov4-tiny* technique with *darknet framework* to train it on the GPU of *collab-engine*. At the end of the training, we will select one of the best training sets by comparing the loss graph function and the confusion matrix of its recall and precise result. With the help of *OpenCV*, we input this training neural to the proposed system, and we also applied coding to make the system able to detect shoes and generate the center point of each shoe detected. As a detailed block, the 2-point intersection equation was used for object counting with a time server from RSTP data. All the trigger events when the shoes are counted will store in a .csv file for daily reports and send to the client side who subscribes to the same topic using *MQTT*. The last process, the system counting report, needs to compare with the manual report to evaluate the reliability of our proposed application at the end of regular working hours.

Moreover, it will explain how our proposed methods are essential to help the shoe industry measure their products within a specific timestamp with low-computation resource consumption.

3.4 Object Detection with YOLO

Yolo is an image processing algorithm that uses a convolution neural network to detect real-time objects with high speed and accuracy. It is proposed to apply with the small-scale and low-computation device. There are three crucial components in the Yolo framework (1) residual blocks, (2) bounding box regression, and (3) intersection over the union known as *IOU*. Below parts is the definition of each component in detail:

(1) Residual blocks: divided image into the grid (SxS)
(2) Bounding box regression: box outline that specific object detection
(3) Intersection over the Union (IOU): describe the overlapping object and make a perfect object detection boundary.

Starting from loading an input image, the model will divide the whole picture into grid blocks and then run over pre-processing training data of the "shoe" object to generate a sample bounding box regression. Next, the model will find the highest bounding box overlap on each other with the best probability detection. Finally, the output image of the detection process will generate the latest bounding box of each object detected with confidential rate detection, as shown in Figure below.

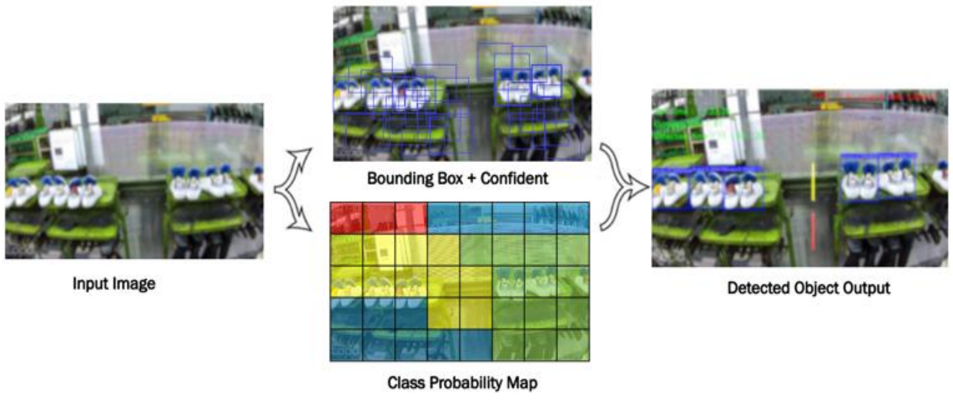


Figure 3.4 YOLO Object Detection Model

The accuracy of the counting system is the critical value of our proposed technique, so object detection with a high confidential rate must achieve. The training image will import into the training system, and then the system needs to train ten times with shuffle data of 10-folds. The system reads streaming Video from the industry's floor plan, and then the system needs to perform specific computations on GPU. Next, training outputs with configuration files are imported to the OpenCV project along with DNN-network to detect objects. If a shoe object is detected with a confidential threshold more significant than 0.7, the system will draw the bounding box of each object seen in the frame. With the proposed technique of 10-folds algorithms, our system can detect unseen shoes.

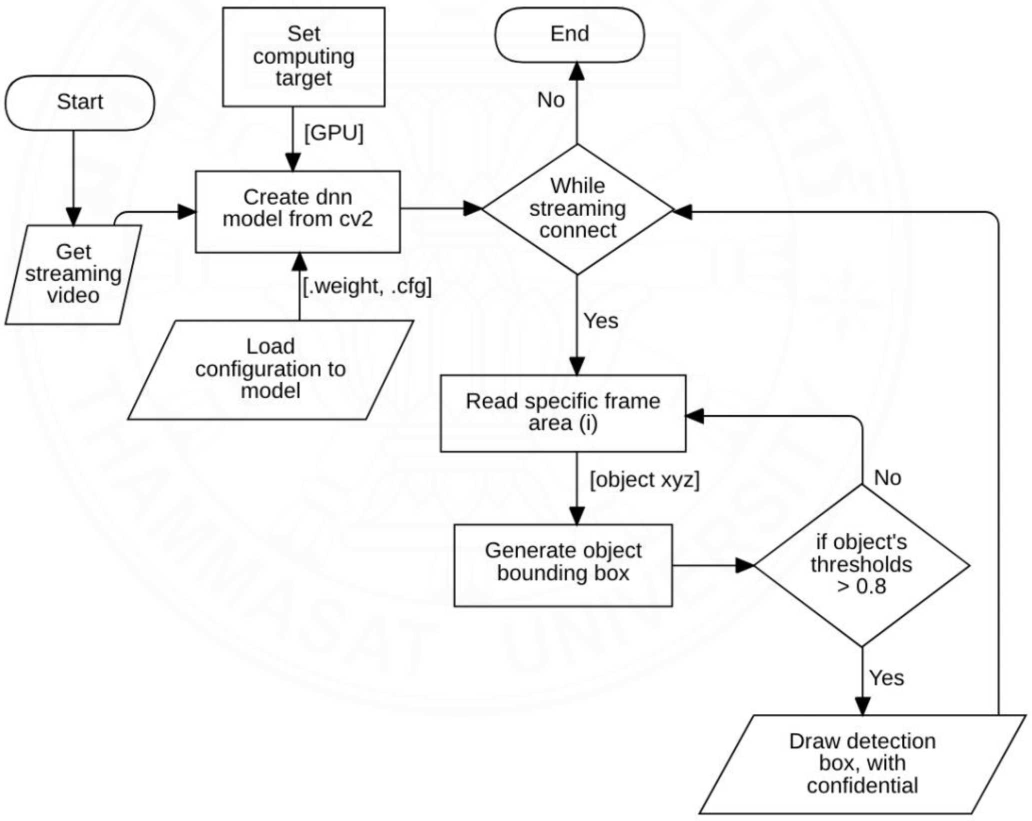

Set computing target
End
Start
No
[GPU]
Create dnn model from cv2
While streaming connect
Get streaming video
[.weight, .cfg]
Yes
Load configuration to model
Read specific frame area (i)
No
[object xyz]
Generate object bounding box
if object's thresholds > 0.8
Yes
Draw detection box, with confidential

Figure 3.5 Real-Time Object Detection Flow Chart

YoloV4 uses the backbone components CPSDarknet53 and other features of a YoloV3 extension update. YoloV4-tiny is a compressed version of YoloV4. It aims to simplify the network structure and cut down unusual parameters to make it flexible and suitable with the embedded device as a minicomputer (raspberry pi, jetson).

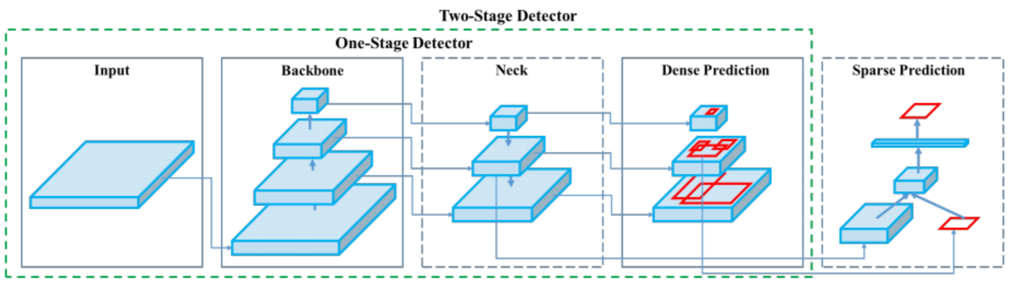


Figure 3.6 YOLOv4-Tiny.Conv29 Darknet Framework

- Backbone: **CSPDarknet53** contains 29 convolutional layers 3 × 3

- Neck: **SPP, PAN**

- Head: **Yolov3 (**Dese Prediction, Sparse Prediction**)**

Bochkovskiy, Wang, and Liao (2020) **Model parameter:**

- Image processing [N x C x H x]

3.4.1 10-Fold Best Training Data Selection

K-folds is a cross-validation method used in many fields of machine learning techniques. 10-Folds is recognized as one of the best sequences of data shuffling technique by comparing the bias of each model of the predictive modeling. There are three different types of data (1) training, (2) validation, and (3) testing.

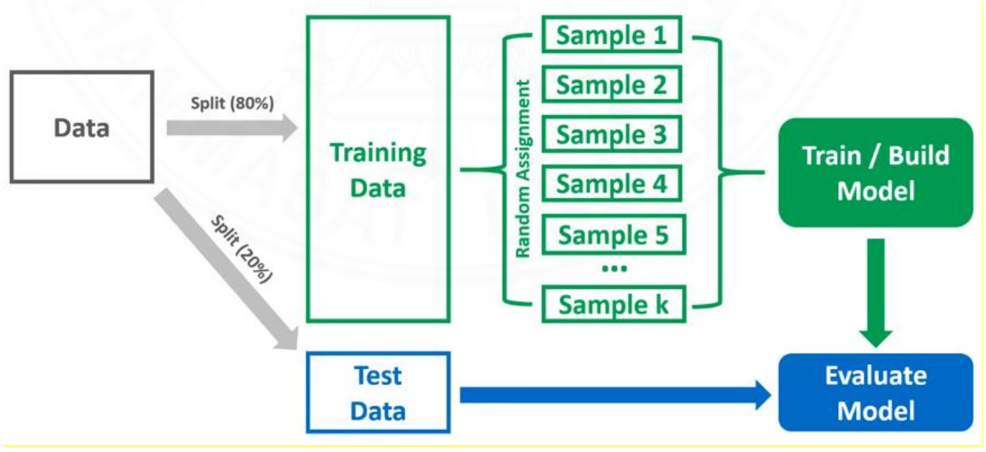


Figure 3.7 10-fold Data Cross-Validation Processing

In the process of the 10-fold model, data between training, testing, and validation will shuffle without duplication, and it estimates the model on each bundle set. This proposed technique can help design a training model with unseen data. Each bundle of 10 sets of the output will simply apply to the same training system as shown in *3.5.2 Object Training Algorithms.*

3.4.2 The 2-points Intersection for Counting Algorithms

After the shoes are detected, the system must count every shoe that passes the counting line. There are two critical components for this proposed counting system, generate the center of shoes detected and draw two counting lines at the suitable area, as illustrated in Figure 3.8.

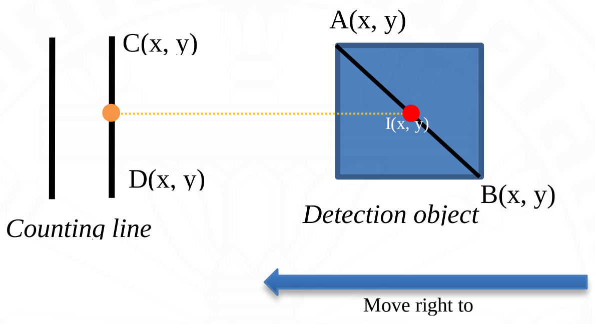


Figure 3.8 Object Counting Technique for 2-Point Intersection Solution

*System behavior: *Shoe Object* move left to counting line

Definition:

● : Center point of object shoe detected

● : Counting line intersected point

I (x, y): object shoe CenterPoint

$y = ax + b$ *(linear equation)* (3.1)

$Ix = \frac{xa+xb}{2}, Iy = \frac{ya+yb}{2}$ => *I (x, y) (object center point)* (3.2)

$Ax + By = C$ (3.3)

The below pseudo code parts are detailed technical explanations of each process step by step. The answer is based on the code concept structure of an application with a python programming project. We also implemented some technical work with the *OpenCV* framework.

PYTHON PSEUDO CODE: *Object Counting algorithm1*

```
-   We assume that we have AB, CD and A(1, 1), B(4, 5), C(1,7), D(3, 5)
-   Draw CD as in frame frame[150:300, 690:700, 1:]
-   Create Blank Array 1 dimension from video frame using NumPy.zero(frame.
    shape, dtype=numpy.uint8)
-   Create a Center of AB for each object detection by cv2.circle(Blank Array, (Ix,
    Iy), thickness, color, -1)
-   While Video is reading:
        o   Sum the Total blank of the detection line position in the video frame as
            NumPy.sum (blank[line_coordinate])
        o   If Total blank > 0 => object touches the line
        o   Then count one object at that time
        o   Else => object does not touch the line
        o   Then re-test another object until the object is detected and counted or
            the program ends.
```

3.4.3 Timestamp Extraction Technique from Previous Task

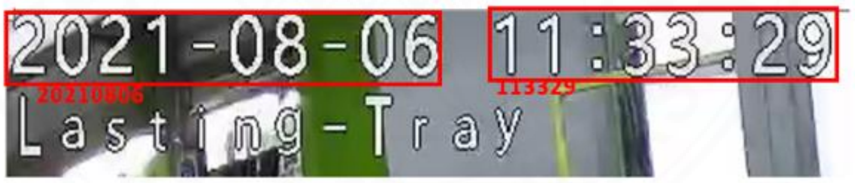


Figure 3.9 Time Extraction in Video Frame

The timestamp for each shoe object detection was extracted from the Video clip. The system will provide counting data when the shoe object passes through the line at the exact timestamp from each testing video clip in the specific frame. For detail, it will present in Figure 3.10. A broker time server replaced this timestamp extraction from the Video. The below flow chart will introduce the concept flow of how the system extracts the timestamp from the recorded input video.

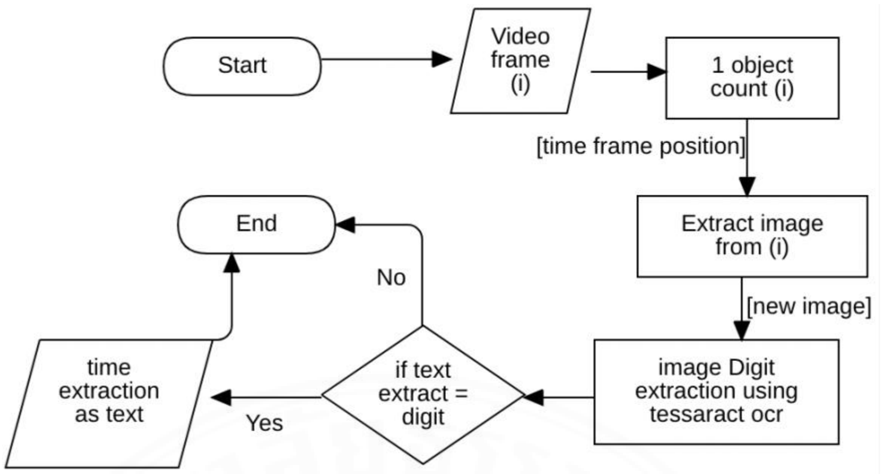


Figure 3.10 Video Time Extraction Flowchart

In the beginning, the system will read the input video. When the object 'shoe' moves across the counting line, the system will capture that frame and call the *extract image function.* In that function, first, we set the position of the timestamp in the structure at the up-left corner to minimize the computation cost. We applied the *TessaractOCR* engine to separate text and numbers. When we get the digit number, we need to split every two digits as the time format, and finally, the system will return that extracted time text to the request calling function. The below part is a detailed explanation using pseudo code step by step.

PYTHON PSEUDO CODE: *Time extraction algorithm2*

```
-   We assume that the Video is loaded and gets its frame
-   Link tesseract cmd by using by tesseract package
-   While the center of the shoe object is intersected line:
      o Resize the frame for specific time location
        img2 = imutils.resize(img)
      o Read image to data for img2 with only digits configuration cmd,
      ⇨ pytes.image_to_data(img2, config)
      o segmentation and split text to data
-   return extraction text result
```

***In the case of Real-Time application**:* when we applied the system to be real-time by using the MQTT framework, a timestamp was provided by the time server whenever the shoes had passed the detection line. It may also delay sometime due to the connection latency is not stable.

3.5 Workflow and Propose Technique

The problem statement clearly shows that the shoe industry needs a production tracking system to help them keep tracking their production line in real-time and reduce counting problems generally made by workers. This research is proposed to solve the problem with the detection line to count the object "shoes" with a timestamp. The following part will explain in more detail.

3.5.1 Label Technique

After selecting all shoes in each photo, we need to export that dataset as a darknet framework for training purposes. The output file from this export contains the coordination of each shoe object with its coordinates, as shown in Figure 3.11 Roboflow is encouraged to label the coordination of the shoe object in each photo. Import data and class with the below properties:

- Image size 416x416
- No rotation slips
- None auto-orientation

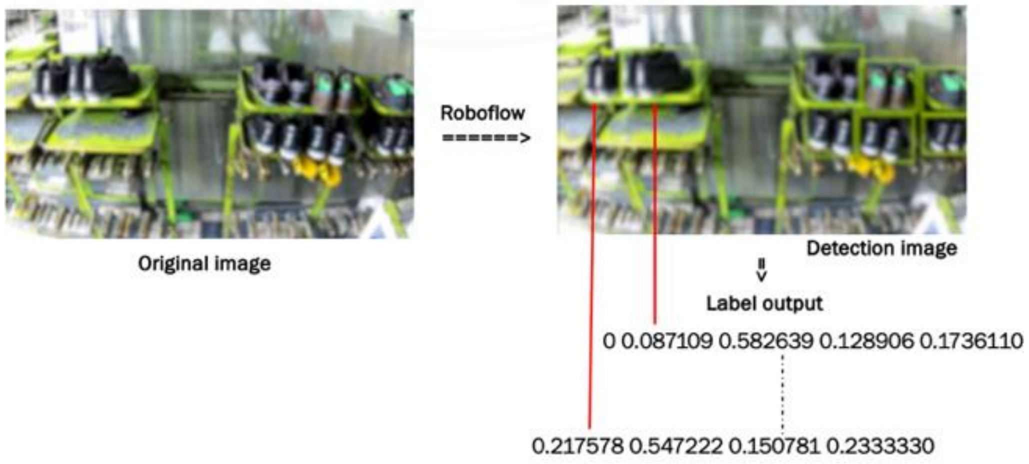


Figure 3.11 Shoes Object Label Technique

3.5.2 Object Training algorithms

There are eighty labels and images, sixteen validations, and ten testing images, which need to be imported to the Jupiter Notebook program to practice ten folds algorithms by shuffling all input data into 10 sets. We train our system with the YOLOv4-tiny darknet framework using a python programming language with the OpenCV2 library. Then, the system used a detector train from the YOLOv4-darknet-23 framework to prepare each 10 sets of training data with YOLOv4-tiny convolution network model 29, and we need to set up some values in the configuration file as shown in 3.12 Then we need to compare the precision and recall of each output and choose only the best weight with the best result.

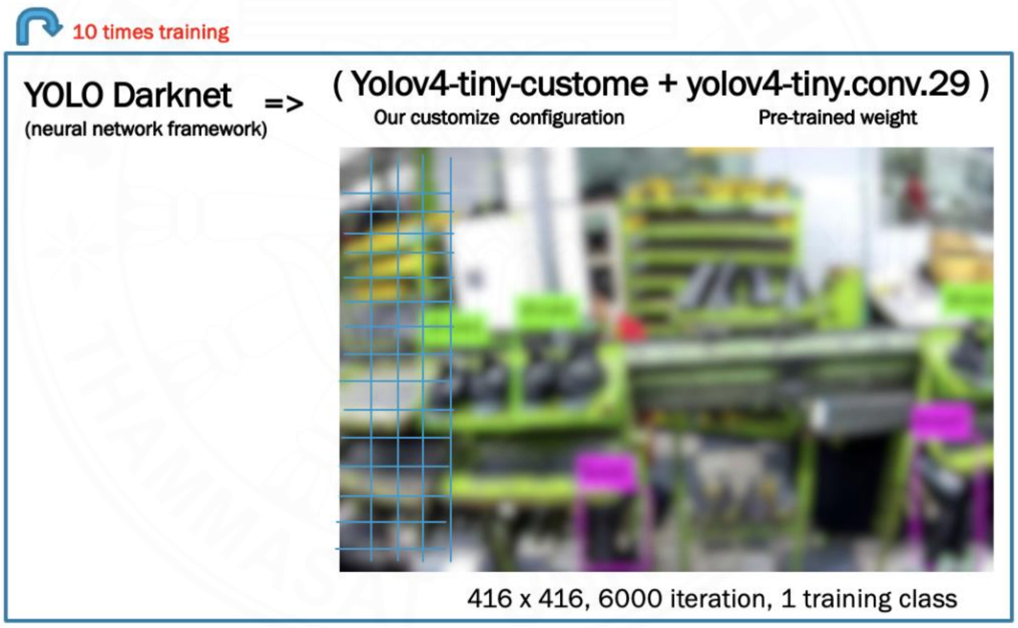

10 times training
YOLO Darknet
(neural network framework)
=>
(Yolov4-tiny-custome + yolov4-tiny.conv.29)
Our customize configuration
Pre-trained weight
416 x 416, 6000 iteration, 1 training class

Figure 3.12 The System Training Model for the Shoe Object

3.5.2.1 Model object Training iteration

Bochkovskiy, Wang, and Liao (2020) proposed two-stage detector was used to find the optimal balance of network resolution, convolution layer, and other parameters. Then proposed system will select an additional block to improve the receptive field by using a different backbone for different detector levels. The authors used YOLOv4 with SPP, PAN, and SAM to enhance performance with a high frame detection rate. Figure 3.4 shows how the object generates each bounding box.

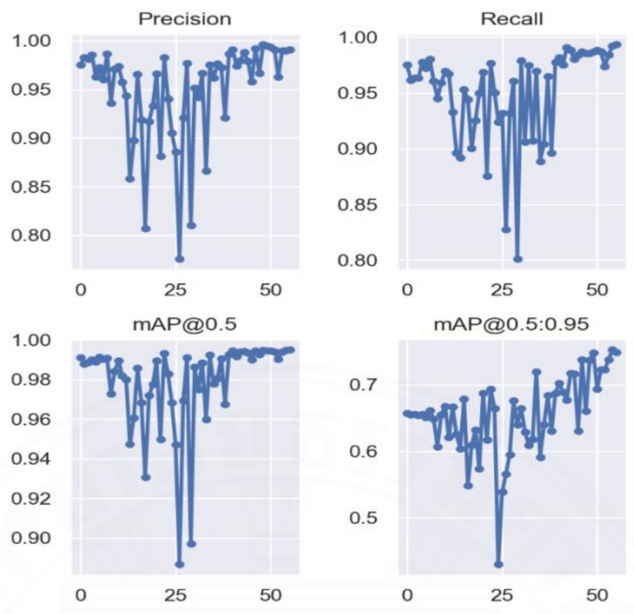


Figure 3.13 Recall and Precision of Object Training Graph

A high Recall and precision can verify that the proposed model can be realistic and best practice in real-life applications because the system will rarely produce a high detection bias. As the Figure above, the probability of our training result almost reaches the total score. The recall helps us identify the system's correctness based on True-Positive.

3.6 Object Detection and Counting Performance measurement

3.6.1 Counting performance with timestamp extraction

From the previous tasks of Ren, Wang, Fang, Song, and Djahel (2020), we tested 8 video clips of 5 min. We drew the counting line at the exact position where workers usually start to count the shoe manually. We compared the actual system counting amount with the manual count from each recorded Video with a timestamp on the video frame. We separate the quality of the evaluation into three types:

- 1st P status means the counting amount of manual count and system count is the same, and the manual timestamp matches the extraction timestamp.
- 2nd T status means the counting amount of both systems is matched, but the timestamp can be slightly different

- 3rd F status means 100% of counting match, but the timestamp is greater than +-3 sec

3.6.2 Real-time Counting Performance

The system daily connects to the camera based on the time interval from 8:00 AM to 12:00 PM. Our proposed system will count the shoes with a timestamp and then save it to a .csv file. At the same time, we also use that streaming video to do manual count with a camera time frame and note it on paper. We compare the total pair of shoes from the system and report file. So, we will know how many shoes our proposed system cannot detect.

3.7 Real-time Shoes Counting with Timestamp

PYTHON PSUESDO CODE: *Object detection algorithm3*

```
-   Import numpy, datetime, cv2, pytesseract, imutils
-   Using cv2 to load Video
-   net = load configuration, weight file to dnn-network
-   create a dnn-detection model with a net parameter
-   While True:
      o Extract classid, score, and box from model detection
      o Then draw a rectangular bounding box with a circle center containing blank
        frame line detection.
      o If Video = END
      o Then END the program
      o Else: counting each object with
      o Then: If the Object touches the line by algorithms1
      o Then: count 1, extract the timestamp from that frame
      o Return: 1 object count with specific detected time.
```

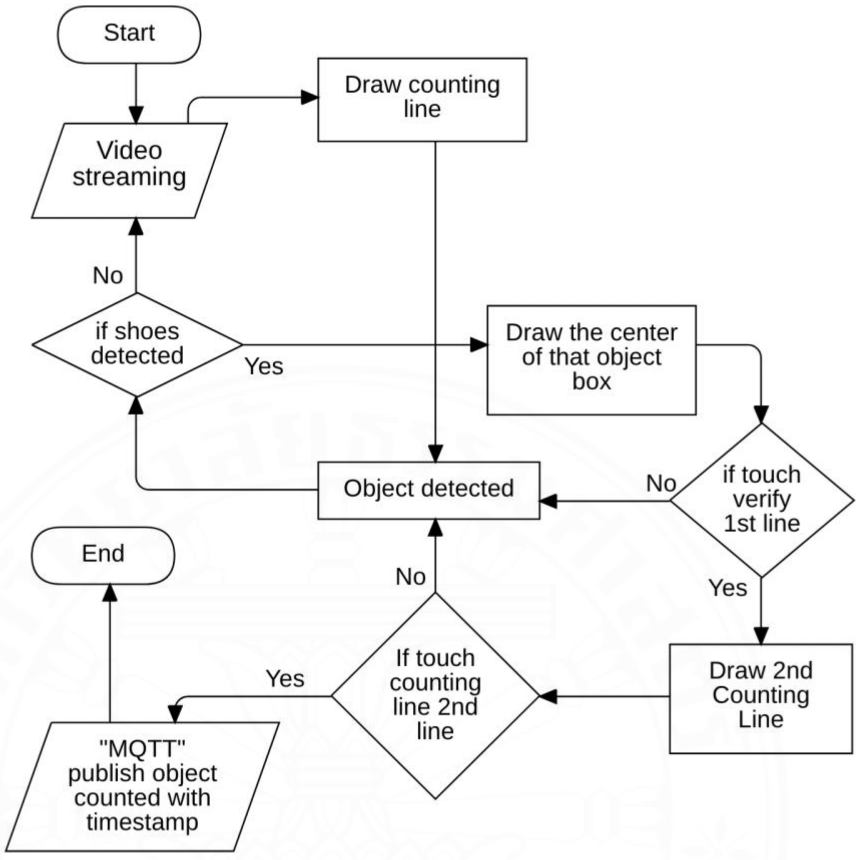


Figure 3.14 Real-time Object Counting Process with Server Time Return

The system connected with real-time streaming Video, and the counting line was drawn in the proper position in the frame, shown in Figure 3.14. We proposed that position because the machine moved forward and temporarily stopped before the counting line. Next, the system accepts only the object “shoes,” which has a more significant threshold than 0.7. When the object is detected, our system will draw the center point of that object. We aimed to find the union point of the detected object and the counting line using the Figure 3.16 algorithm. If the detected object moves to touch the 1st line, the 2nd line of the counting line will appear, and then the system will count that object whenever it continues to pass 2nd with a timestamp. The counting data will suddenly be published to the Dx-server, the MQTT broker.

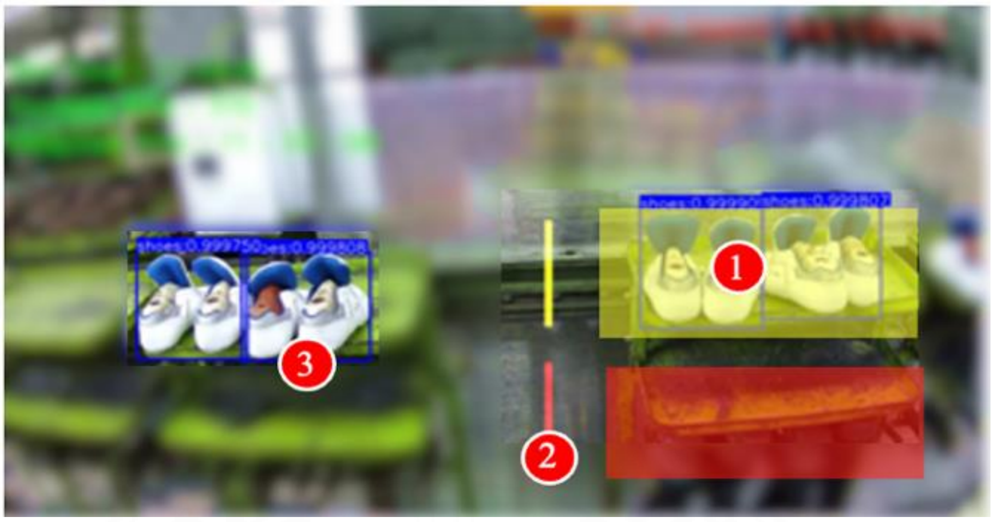


Figure 3.15 Actual Case Applied for Object Counting with Proposed Algorithms

We got a 420x420 image frame of HD resolution from camera streaming. In the actual case study in the manufacturing floor plan, the counting lines are separated into two parts, UP (Yellow) and DOWN(Red), as shown in Figure 3.15. The objects in the upper area will count when it moves to touch the yellow line, the same process with the red line. Whenever the shoe is counted, the system publishes data with a timestamp to the broker, and then it will repeat this action until losing the camera's connection. Finally, the planner can collect these data with a CSV file or use our proposed system design, as shown in appendix A, to see the real-time update. 2-line for object counting is proposed to solve some problems caused by workers placing the shoe incorrectly or unstandardized on the conveyor.

The concept method above has been changed for real-time application due to resource consumption and the constraint broken. So, we need to identify and optimize the root cause of the resource overflow or any interrupt errors during production time. All these errors can lead to the counting problem. The root cause of over-resource consumption caused by the input frame and unwanted object detection process and shoe object detection accuracy rate drop-down was troubled due to the working behavior of workers. The Figure below will explain how our challenge can be our opportunity to solve the problem simply.

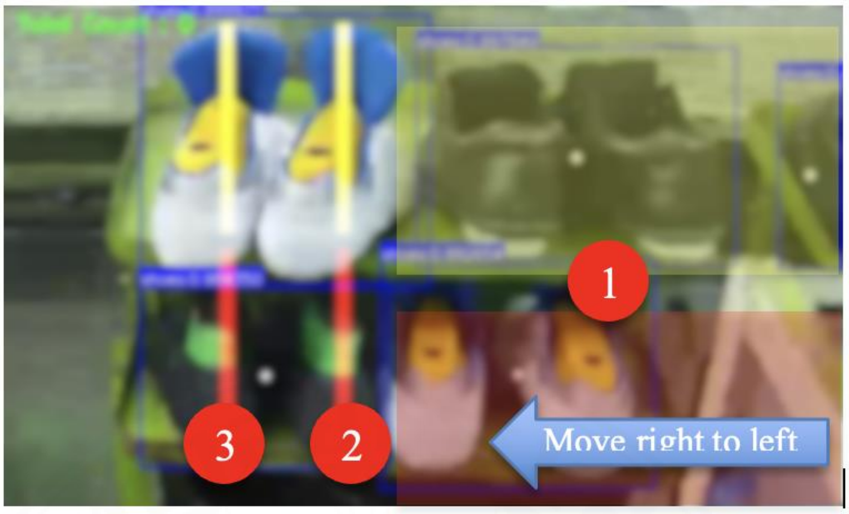


Figure 3.16 Optimization of Resource Consumption and Broken Constraint

- (1) shoe counting await area, which divides into the yellow and red area
- (2) verify counting line, which passed the Boolean to (3) line to appear
- (3) confirm the counting line, which the shoe is finally count

For real-time objects, the counting technique is quite a challenging task to achieve. The system also makes the time latency due to network connection problems, computation resources, and other issues generally caused by workers' working behavior. We try to minimize the dimension from HD input to the new frame, in which the counting line position is set to the middle of the image frame, and its width is from the length of 3 detection objects. Then, the system will need to restart every 4 hours to load a new resource after the counting data are saved to a .csv file and sent back to clients as an MQTT service. The system will publish a message from the message queue protocol with time server time whenever the shoe passes the second counting line (3). The distance between both lines (2) & (3) was designed to prevent some detection errors during counting. It can play a role in regenerating the object center point verification. Sometimes, one object can generate two center points due to the constraint being broken, as shown in 3.8 below.

3.8 Constraints May Affect the Counting System

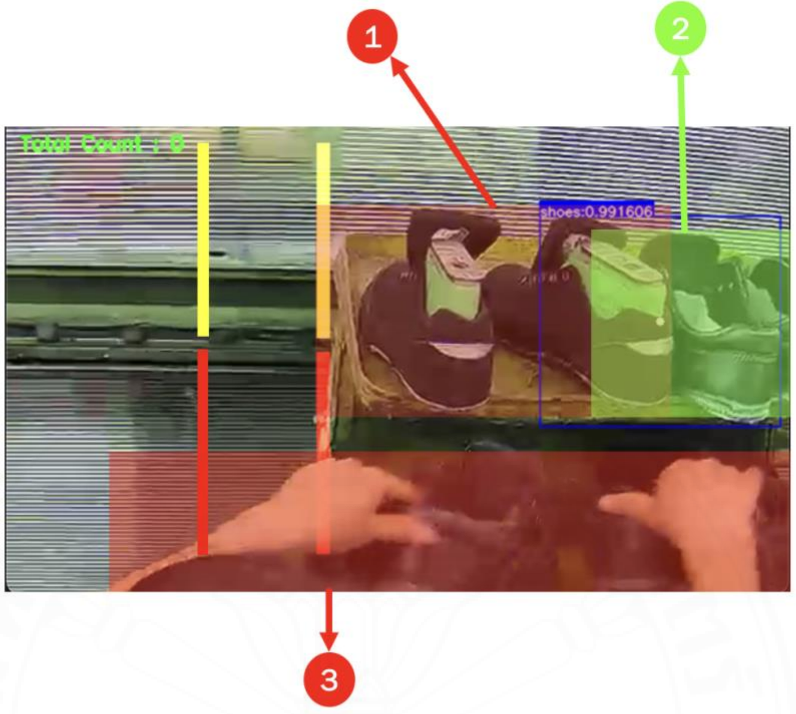


Figure 3.17 Worker's Behavior Errors that can Affect the Counting System

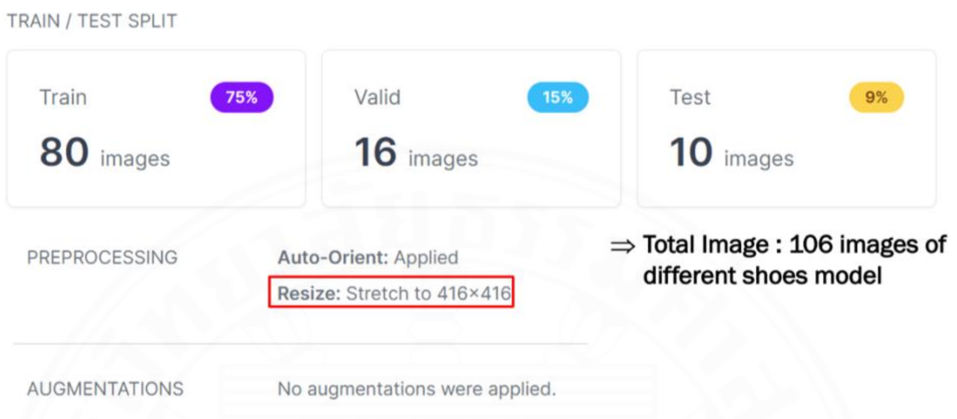
The constraint is a rule that can make the system work with less error. Working in an actual factory, different workers have different work habits, so to make our counting system can count shoes smoothly, we need to study and define some constraints for the worker to obey, as shown in Figure 3.8.

Constraints explanation:

- (1) object shoe placed not as a pair
- (2) each pair of shoes is placed too near to the other
- (3) Workers took shoes before they passed the counting line, and in some cases, our system detects the worker's head as shoes

CHAPTER 4
SYSTEM EVALUATION & PLANNING

4.1 Dataset


TRAIN / TEST SPLIT

Train 75% 80 images

Valid 15% 16 images

Test 9% 10 images

PREPROCESSING

Auto-Orient: Applied

Resize: Stretch to 416×416

⇒ Total Image : 106 images of different shoes model

AUGMENTATIONS

No augmentations were applied.

Figure 4.1 Training Dataset & its Preprocessing Properties

We recorded excellent videos through the online Rapid Spanning Tree Protocol for our system evaluation. We trained our system with 106 images of 40 types of shoes which were selected from many recorded videos. We separated data into 80 training, 16 validations, and 10 image tests. We also recorded 2 minutes of video clips to test untrained shoe objects.

Table 4.1 Untrain Tested Data for 2 Minutes Video Clip

N	Video	Duration	Frame Rate	Shoe Amount Count	Unseen Shoe Count
1	L_V 1	2 min	13.75	10	3
2	L_V 2	2 min	13.75	12	2
3	L_V 3	2 min	13.73	11	2
4	L_V 4	2 min	13.70	13	0
5	L_V 5	2 min	13.70	15	3
6	L_V 6	2 min	13.73	11	5
7	L_V 7	2 min	13.73	15	5

After testing the counting system with the input of 15 video clips from the previous proposal tasks, we tried to apply the Real-time video image streaming from station *lasting_in* and *lasting_out*, which was installed at the object's visible position and can be accessible by rapid spanning tree protocol as an IP camera through the network. Our streaming video input dimension is 320 pixels, and a .csv file with the exact shoes detection amount and timestamp will provide every 4h, twice a day, as shown in the graph below, except OT.

- 1h: is usually the afternoon resting time
- 3h: is normally working overtime

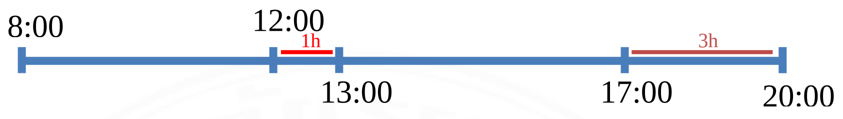


Figure 4.2 Working Time Interval

4.2 Experimental Result

This part illustrates the performance test of our application's implementation for object counting with the timestamp and the comparison between manual and camera counts. We used 15 video clips to evaluate our proposed approach's performance, as shown in Table 4.4. At the end of this part, our system will define the critical attribute for report comparison of Manual and System counting for real-time application.

4.2.1 Shoe object training

Our approach got 90% true-positive and 10% false-positive, so the system rarely confused to detect another object as a shoe. We also got a confidentiality rate of 0.995 for each shoe that appears in the detection frame. The percentage of precision and recall is almost perfect. Figure 4.3 will display a curve of loss function for each iteration training. When the training process reached the 2400th iteration, the YOLOv4-tiny convolution network 29 model's convolution network performed well, making the line go to zero point. The critical point of the customize training attribute of the YOLOv4 configuration file with one-shoe class detection must be to set the *batch size* to 64 to increase the computation power with GPU and the *subdivisions* to 24 for memory capacity running. The system will use more resources if subdivisions become smaller. Then, we load the input image with 416 x 416 dimensions. We chose 6000 iterations to train our proposed object detection with Darknet-23. We set the running steps to 80% maximum batch size to 4800 and 5400. At the end of the configuration file editing, we added the filter to 32 for several layer outputs. The outcome of our proposed configuration value will make detection accuracies up to 99% for trained objects and 85% for untrained shoe models. We have also considered choosing the number of thresholds > 0.7. So, the system rarely detects objects such as the worker's head and conveyor's hanger as shoes.

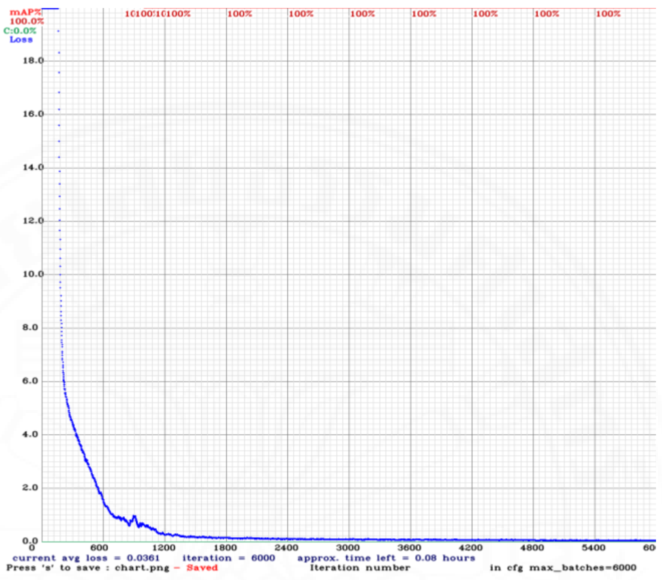
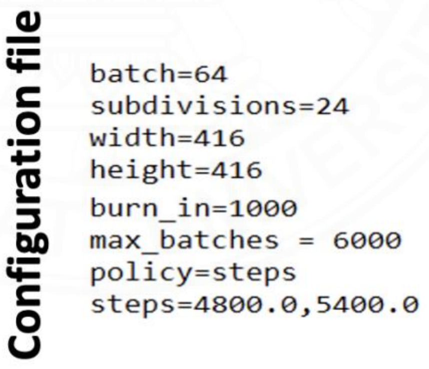


Figure 4.3 Loss Graph During Training



Configuration file

```
batch=64
subdivisions=24
width=416
height=416
burn_in=1000
max_batches = 6000
policy=steps
steps=4800.0,5400.0
```

Figure 4.4 Input Configuration File Detail

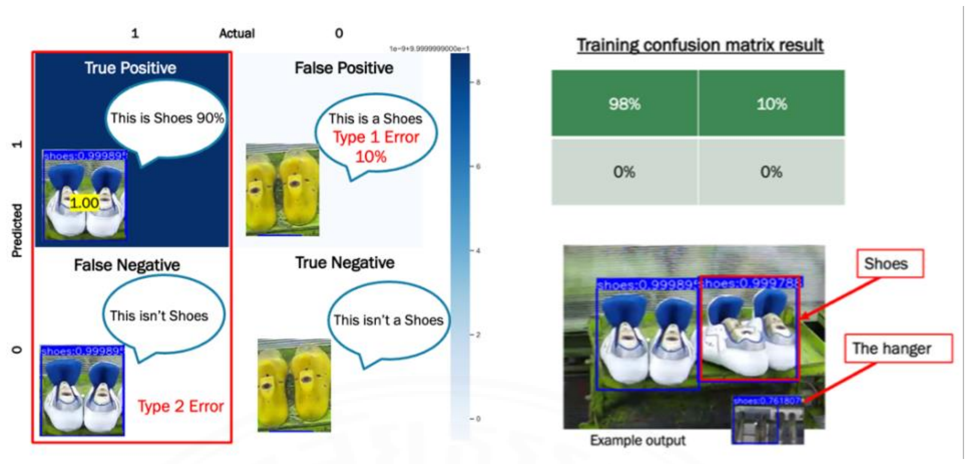


Figure 4.5 Confusion Matrix Result True/False Negative/Positive Graph

The confusion matrix helps us verify our prediction's correctness and compare the result with the actual value. In the case of the Figure above, we saw that our proposed training model could achieve 90% *True Positive*, which means that the system rarely detects another object as a shoe. Other 10% of the *False Positive* verification is usually caused by the distance and the color of some image area, which made the system false to separate which object is a shoe. For example, the hanger is sometimes detected as a shoe. Anyway, even if the False Positive occurred, some of the confidential rates of that object are still lower than 0.7

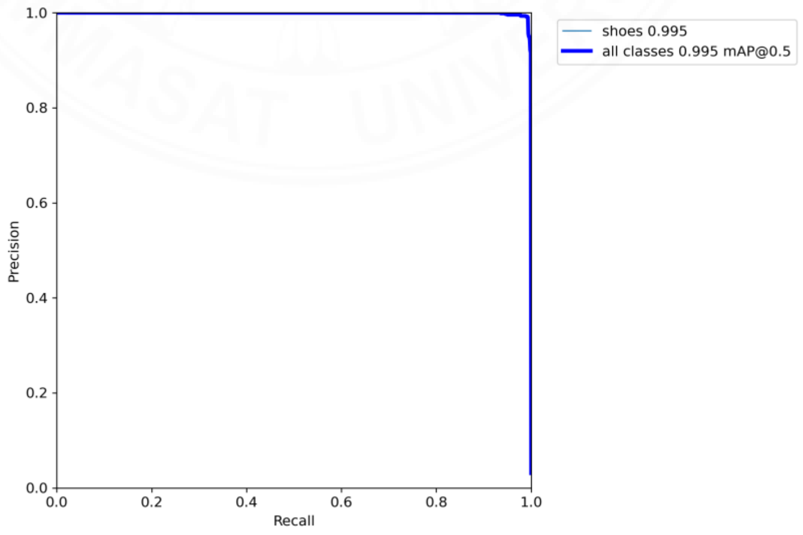


Figure 4.6 Shoe's Recall and Precision Result Graph

4.2.2 Unseen shoes in object detection

An untrained object means that the object shoe model is not in the group of 40 models of 106 training images. Our system is also supposed to train that model to reduce shoe detection issues when the shoe company starts to produce a new model.

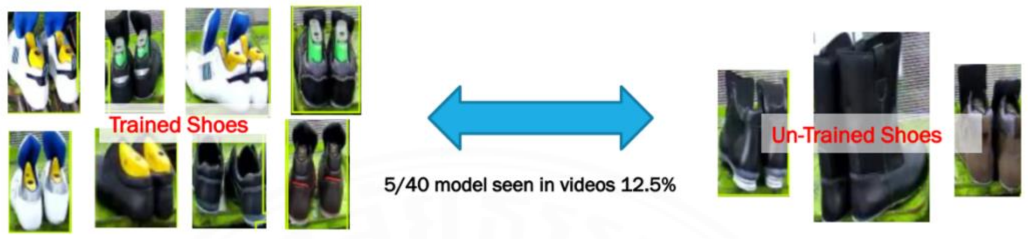


Figure 4.7 Train & Untrain Shoe Identify

Table 4.2 Object Detection Model Untrain Evaluation

Testing VDO	Duration (min)	Trained Shoes		Average Threshold of train shoes	Untrained Shoes		Average Threshold of Untrain shoes	Total count Manual/ System
		*MC	*SC		*MC	*SC		
L_v1	2	7	7	0.99	3	3	0.70	10/10
L_v2	2	10	10	0.99	2	2	0.75	12/12
L_v3	2	9	9	0.99	2	2	0.74	11/11
L_v4	2	13	13	0.99	0	0	0.73	13/13
L_v5	2	12	12	0.99	3	3	0.70	15/15
L_v6	**2**	**6**	**6**	**0.99**	**5**	**4**	**0.65**	**11/10**
L_v7	2	10	10	0.99	5	5	0.75	15/15

***MC:** Manual Count, * **SC:** System Count

There are about 5 shoe models detected with a low confidential threshold. These 5 models were verified as the untrained shoe in our video recording clip 6 (L_V6). Sometimes, untrained object shoes can also affect our proposed counting system, but most likely, as shown in Table 4.2.2 above, our system still achieves almost 85% for untrained object detection. There were also cases where miscounting was caused by broking constraints, as shown in Figure 3.8.

4.2.3 Object Counting with Timestamp Result

In this part, we will see how our proposed system evaluates the time that a shoe object is counted by comparing Manual Count with System Count, as below table.

Table 4.3 Manuel and System Object Counting with Timestamp Evaluation

		Manuel System		System Count		Count (c)	Timestamp (t)
N	Type	Count	Timestamp	Count	Timestamp	P/T/F	P/T/F
1	Shoe	1	10:01:10	1	10:01:10	Pc	Pt
2	Shoe	1	10:01:57	1	10:01:57	Pc	Pt
3	Shoe	1	10:02:00	1	10:02:05	Pc	Tt
4	Shoe	1	10:02:50	1	10:02:50	Pc	Pt
Total		4		4		Pc	Pt

We divide our system evaluation status into 3 fundamental values, such as P: Perfect match, which means that the manual count matches the System count. T status represents matching counting, but the result has 1-minute different conditions. (F status) refers to the mismatch condition. As a result, our proposed system performs well with a timestamp. This technique was also applied in section 4.2.5.

4.2.4 System Evaluation Test

The manual and camera result comparison illustrates that our proposed system performed very well with shoe counting with constraint time ±3 seconds. However, sometimes our application also met F status for timestamp evaluation. The detail of the 15 videos evaluation test is shown in the below table.

Table 4.4 Result of 15 Videos Evaluation, Du: Duration, Co: Count, EP: Evaluation

N	Video Name	System Performance Parameter				EP	
		P (%)	T (%)	F% > ±3	Co	% Ram	% Cpu
1	L_v1	56.2	43.7	0	28/28	2	65
2	L_v2	70	30	0	14/14	2	61
3	L_v3	13.3	86.6	0	22/22	3	64
4	L_v4	13.3	80	6.6	19/19	1	65
5	L_v5	12.5	68.7	18.5	23/23	2	66
6	L_v6	50	50	0	19/19	2	65
7	L_v7	100	0	0	7/7	2	62
8	L_v8	29	71	0	26/26	2	60
9	L_v9	80	20	0	22/22	2	61
10	L_v10	50	50	0	10/10	3	60
11	L_v11	25.7	64.3	10	16/16	2	61
12	L_v12	31.25	62.5	6.25	22/22	2	60
13	L_v13	60	40	0	25/25	1	63
14	L_v14	100	0	0	22/22	2	60
15	L_v15	40	60	0	27/27	1	61

The system achieved 99% object counting accuracy, then 70.0% for perfect match (P status) timestamp and 20.0 % for (T status), and 10% for (F status). Our proposed approach still has some limitations. It produced an F status because the recorded video clips have low resolution, usually messed up because of an internet connection problem during video streaming. Our device's computation uses GTX960M VGA, 4GB of Ram, direct X = 12 with CPU Intel Core i.7

4.2.5 Real-time Evaluation Test

We applied our proposed approach with jetson-nano, installed onside with a Camera station. We tested our real-time performance by counting shoes every 4h from 8 AM until noon with Jetson-nano computation with 4GB of Ram, Maxwell GUP: 921MHz + ARM Cortex-A57: 1.43GHz, 16GB storage

Table 4.5 The 4 hours of Real-time Shoe Object Counting Evaluation Report

Date	Station	Manual Count (pair)	System Count (pair)	Missing
04 July 2022	Lasting_Out	540	540	0
05 July 2022		500	500	0
06 July 2022		**490**	**485**	**-5**
07 July 2022		480	480	0

=> **The Missing shoes**: The identification of miss counting reason is divided into 2 cases (Human and System).

- Human case: worker broke the constraint as shown in section 3.4, and the Camera position has been moved, or the worker put things to block camera vision, etc.

- System case: untrained shoe model with less than 0.7 thresholds, detect another object as shoe, and in the rare case, resource overheats caused by long-term processing.

4.3 Cycle Time and Flow Time Calculation

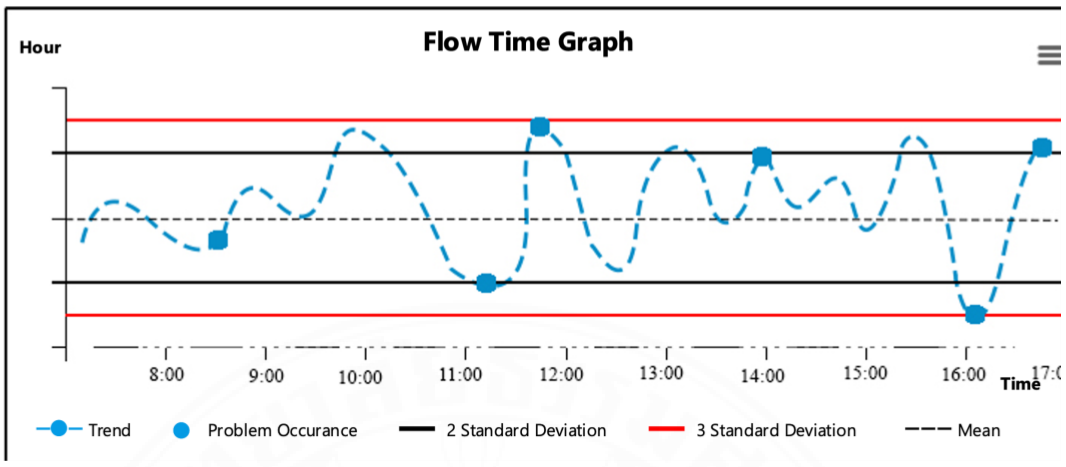
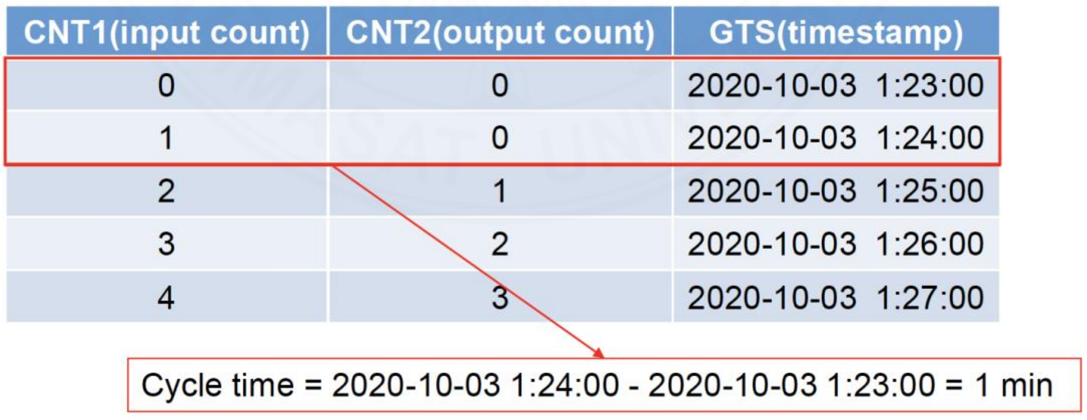


Figure 4.8 Quality Control Graph

Flow Time and Cycle time graphs are essential to our design system outcome. This graph can let the planner track the production flow and the problem during production. There is a sample formula for flow time and cycle time, as shown below:

(1) Cycle time input = input time of current unit – input time of the previous unit

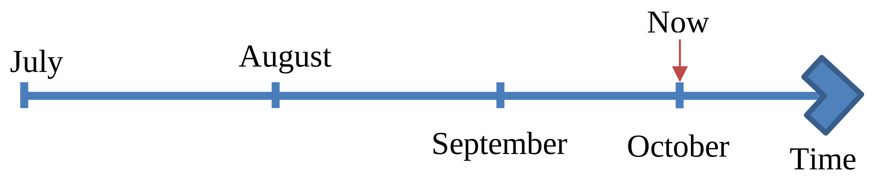
(2) Cycle time output = output time of current unit – output time of the previous unit



CNT1(input count)	CNT2(output count)	GTS(timestamp)
0	0	2020-10-03 1:23:00
1	0	2020-10-03 1:24:00
2	1	2020-10-03 1:25:00
3	2	2020-10-03 1:26:00
4	3	2020-10-03 1:27:00

Cycle time = 2020-10-03 1:24:00 - 2020-10-03 1:23:00 = 1 min

We can also calculate the mean and standard deviation (SD) of each data receiving graph above:



⇨ Mean: $= \frac{\sum_{i=1}^{N} x_i}{\text{n}}$

⇨ Standard deviation (SD): $sqrt\left(\frac{sum((-Mean)^2)}{n-1}\right)$

Where:

$x_i : value\ of\ dat\ at\ \ i^{th}\ item$

$n : number\ of\ items\ in\ the\ sample$

(1) 3 SD line: upper line = avg + 3*sd, lower line = avg - 3*sd

(2) 2 SD line: upper line = avg + 2 *sd, lower line = avg – 2*sd

(3) 1 SD line: upper line = avg + 1*sd, lower line = avg-1*sd

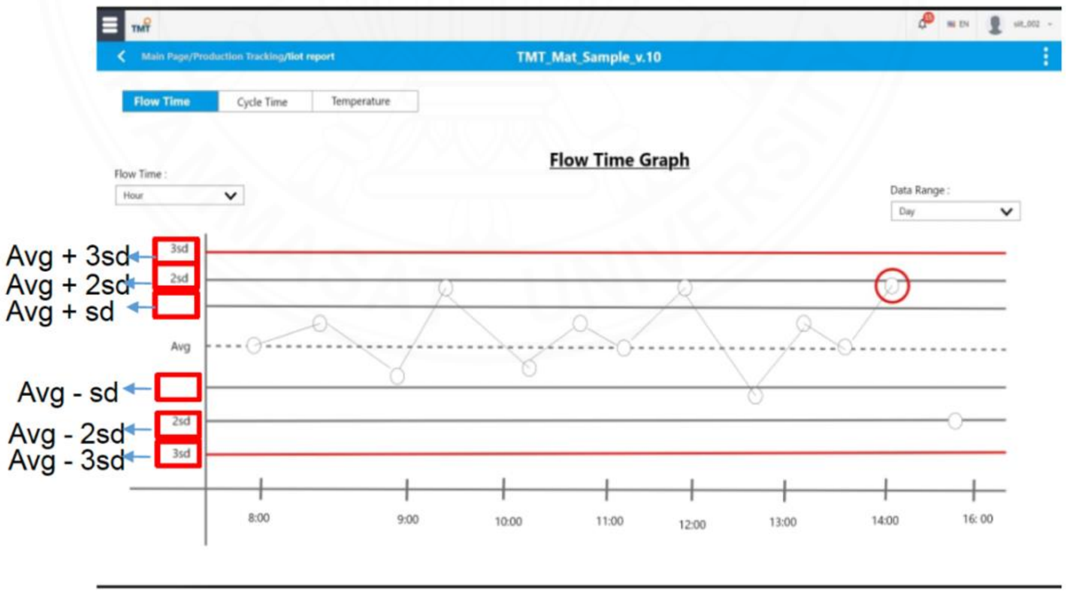
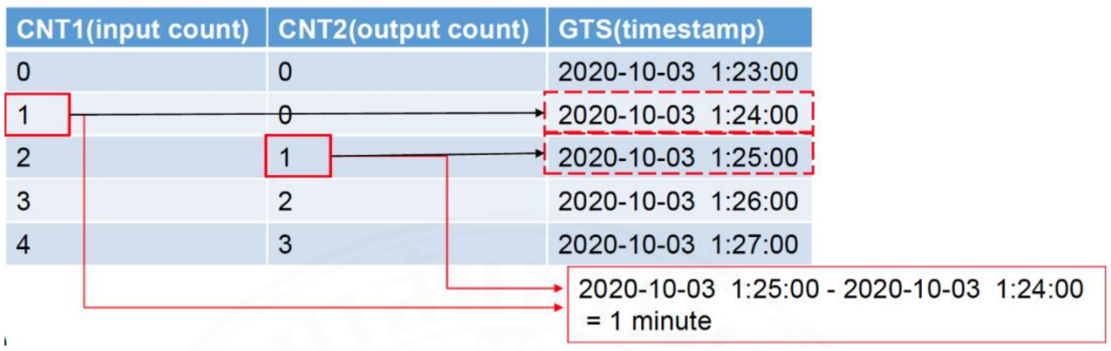


Figure 4.9 Standard Deviation Graph

FLOW CHART OF ONE MACHINE FORMULA:

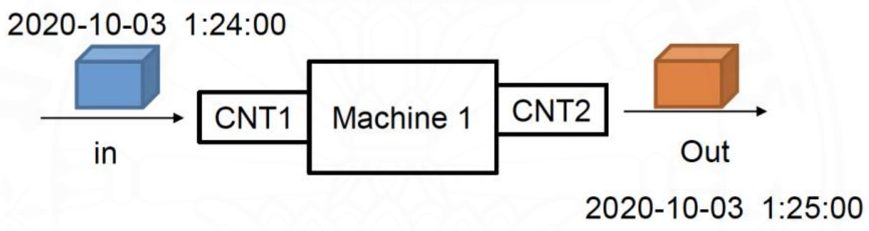
(1) Flow time = output time – input time



CNT1(input count)	CNT2(output count)	GTS(timestamp)
0	0	2020-10-03 1:23:00
1	0	2020-10-03 1:24:00
2	1	2020-10-03 1:25:00
3	2	2020-10-03 1:26:00
4	3	2020-10-03 1:27:00

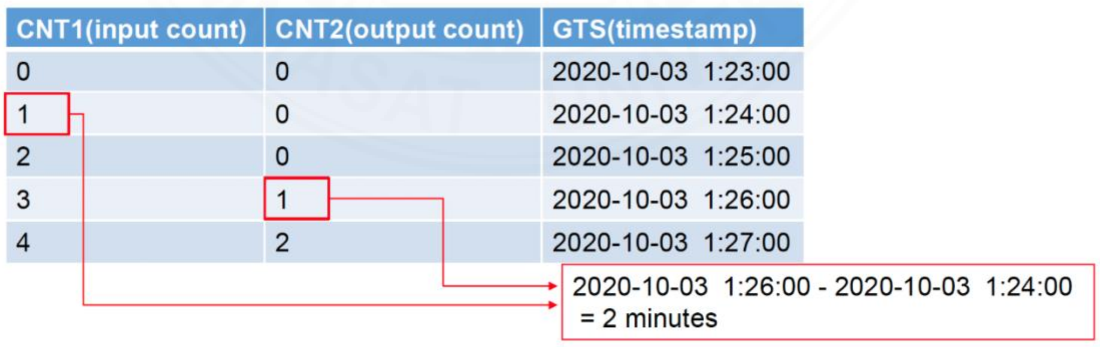
2020-10-03 1:25:00 - 2020-10-03 1:24:00
= 1 minute

Sample Explanation:



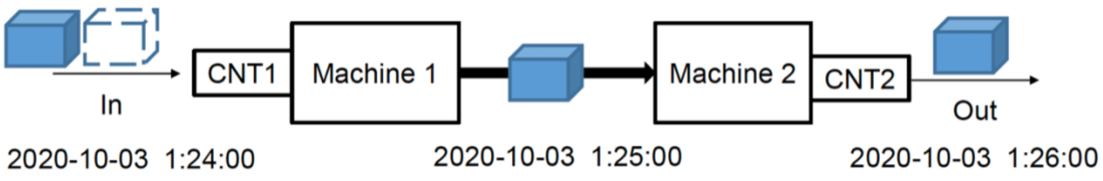
⇨ **FLOW CHART OF TWO MACHINE FORMULA**

(2) Flowtime = output time – input time



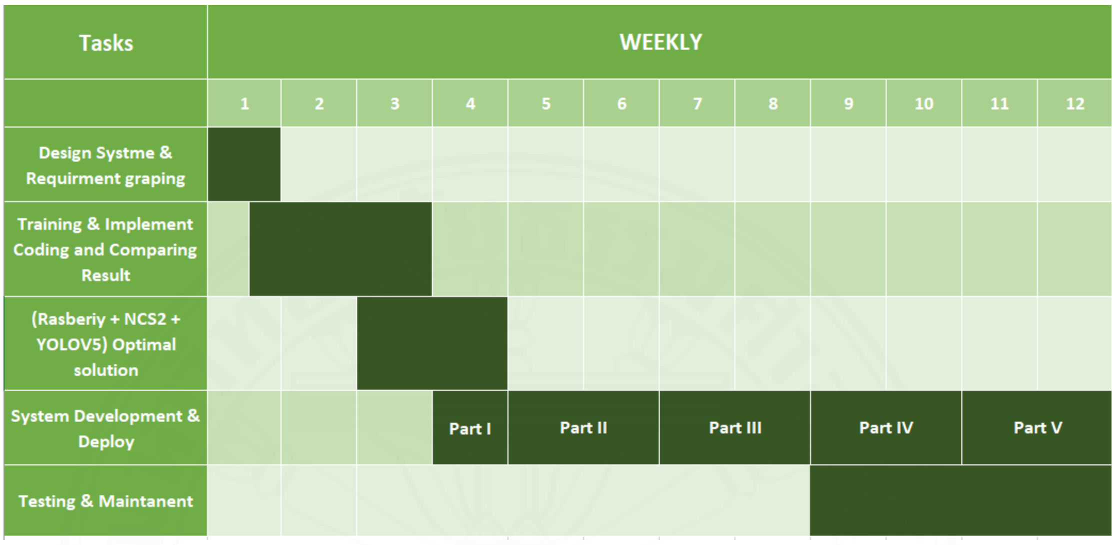
CNT1(input count)	CNT2(output count)	GTS(timestamp)
0	0	2020-10-03 1:23:00
1	0	2020-10-03 1:24:00
2	0	2020-10-03 1:25:00
3	1	2020-10-03 1:26:00
4	2	2020-10-03 1:27:00

2020-10-03 1:26:00 - 2020-10-03 1:24:00
= 2 minutes



CHAPTER 5
PLANIFICATION

5.1 Project Development Planning



Tasks	WEEKLY											
	1	2	3	4	5	6	7	8	9	10	11	12
Design Systme & Requirment graping	■											
Training & Implement Coding and Comparing Result		■	■									
(Rasberiy + NCS2 + YOLOV5) Optimal solution			■	■								
System Development & Deploy				Part I	Part II		Part III		Part IV		Part V	
Testing & Maintanent									■	■	■	■

Figure 5.1 System Planification

This planification helps us achieve the application development separate a straightforward task and deadline. It was divided into five different phases. We start with the design system and the system requirement and then choose the best technology suitable for our development. Next, we need to design our machine-learning model with applied counting algorithms. Ultimately, we will break down all tasks and develop an application with specific time feature release expectations to let the client test. If any changes or errors happen, we will have more time to fix them.

CHAPTER 6
CONCLUSION

6.1 Application Summarize

This research applied YOLOv4-tiny with DNN in the OpenCV framework to make shoe detection more accurate while using a 2-point intersection algorithm to count the object and the Mqtt message protocol and time server for a real-time counting system. The object detection threshold is a crucial point of object counting accuracy, so we applied our selected 106 images with 10-folds algorithms, which shuffle data into 10 packages. Yolo-Darknet-23 with Convolution neural model 29th to train 10 times with different training data of that 10 packages output. The system illustrated that our proposed training model could detect an untrained object, as shown in section 4.2.2.

We evaluated our system with 4h real-time evaluation. Our system got 99% accuracy of object counting, while 90% of timestamps and the design system able to detect 90% of unseen training shoes during the production process. Anyways, the system needs more resources for computation for the whole day. So, our proposed method of frame detection area reduction and streaming dimension input should be deducted to make our suggested solution use resources more efficiently. Table 4.2.4 illustrates that this proposed technique is suitable for a low-computation device such as Raspberry Pi4 or Jetson Nano for a real-time counting system.

However, counting with timestamps sometimes does not match the actual case due to the network latency for long-hour detection and counting. In the following tasks, we will try to optimize the computation resource, which makes our system can work more robustness. We will also train the worker to interact with our proposed system.

REFERENCES

Alsanabani, A., Saeed, S.A., Al-Mkhlafi, M., & Albishari, M. (2021). A Low Cost and Real Time Vehicle Detection Using Enhanced YOLOv4-Tiny. *2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA),* 372-377.

Bochkovskiy, A., Wang, C., & Liao, H. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934.* https://doi.org/10.48550/arXiv.2004. 10934

Biradar, K. M., Gupta, A., Mandal, M., & Vipparthi, S. K. (2019). Challenges in Time-Stamp Aware Anomaly Detection in Traffic Videos. *IEEE Computer Vision and Pattern Recognition Workshops (CVPRW-2019).* https://doi.org/10.48550/ arXiv.1906.04574

Covavisaruch, N., & Saengpanit, C. (2004). Time Stamp Detection and Recognition in Video Frames. *The 2004 Internation conference on Image Science, System and Technology (CISST 2004)*, Las Vegas, Nevada, USA

Doan, T., & Truong, M. (2020). Real-time vehicle detection and counting based on YOLO and DeepSORT. *2020 12th International Conference on Knowledge and Systems Engineering (KSE)*, 67-72.

Intalar,N.,Chumnumporn,K.,Jeenanunta,C. & Tunpan,A.(2021).Towards Industry 4.0: digital transformation of traditional safety shoes manufacturer in Thailand with a development of production tracking system. *Engineering Management in Production and Services*,13(4) 79-94. https://doi.org/10.2478/emj-2021-0033

Jocher, G., Stoken, A., Borove, J., et al. (2020). Utralytics/YOLOv5: Bug Fixed and Performance improvement. *Zennodo*. https://zenodo.org/record/ 4154370#.Y5PSKy8RppQ

Jiang, Z., Zhao, L., Li, S., & Jia, Y. (2020). Real-time object detection method based on improved YOLOv4-tiny. *Journal of Network Intelligence, Volume 7, Number 1, February 2022*. https://doi.org/10.48550/arXiv.2011.04244

Li, Z., Abu Farha, Y., & Gall, J. (2021). Temporal Action Segmentation from Timestamp Supervision. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 8361-8370.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. *arXiv:1506.02640*. https://doi.org /10.48550/arXiv.1506.02640

Ren, P., Wang, L., Fang, W., Song, S., & Djahel, S. (2020). A novel squeeze YOLO-based real-time people counting approach. *International Journal of Bio-Inspired Comput*ation, 16, 94-101.

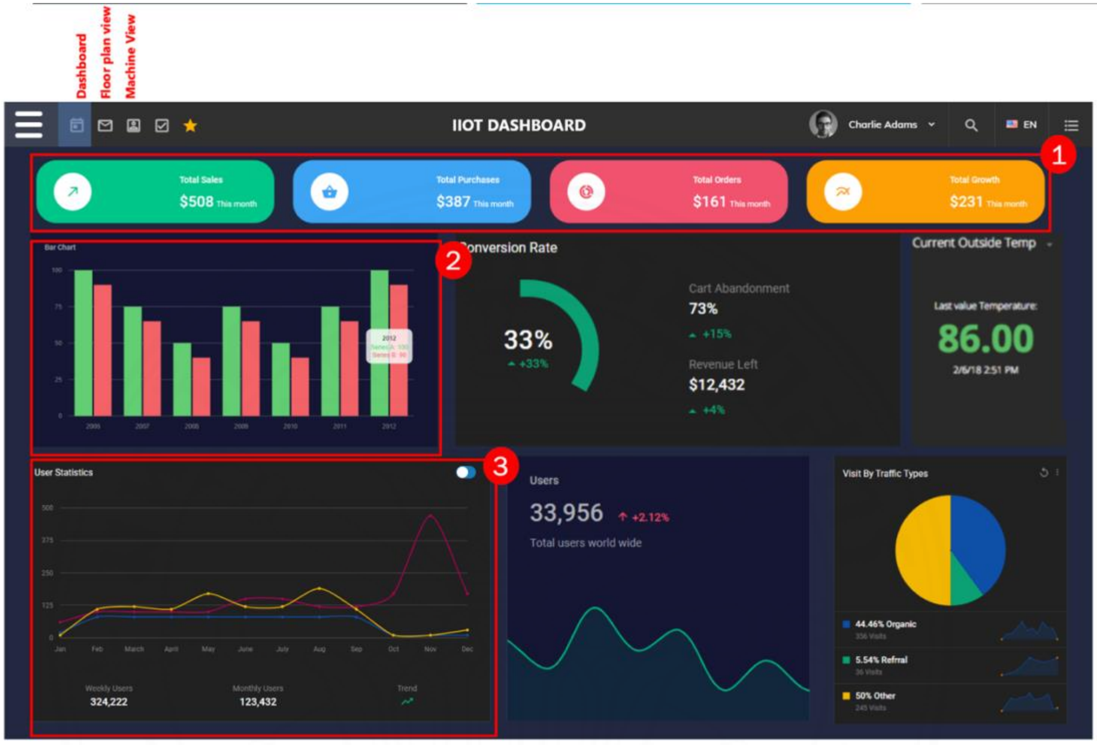
Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6517-6525.

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv:1804.02767*, 1. https://doi.org/10.48550/arXiv.1804.02767

Yang, Y., Xie, G., & Qu, Y. (2021). Real-time Detection of Aircraft Objects in Remote Sensing Images Based on Improved YOLOv4. *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, 5, 1156-1164.

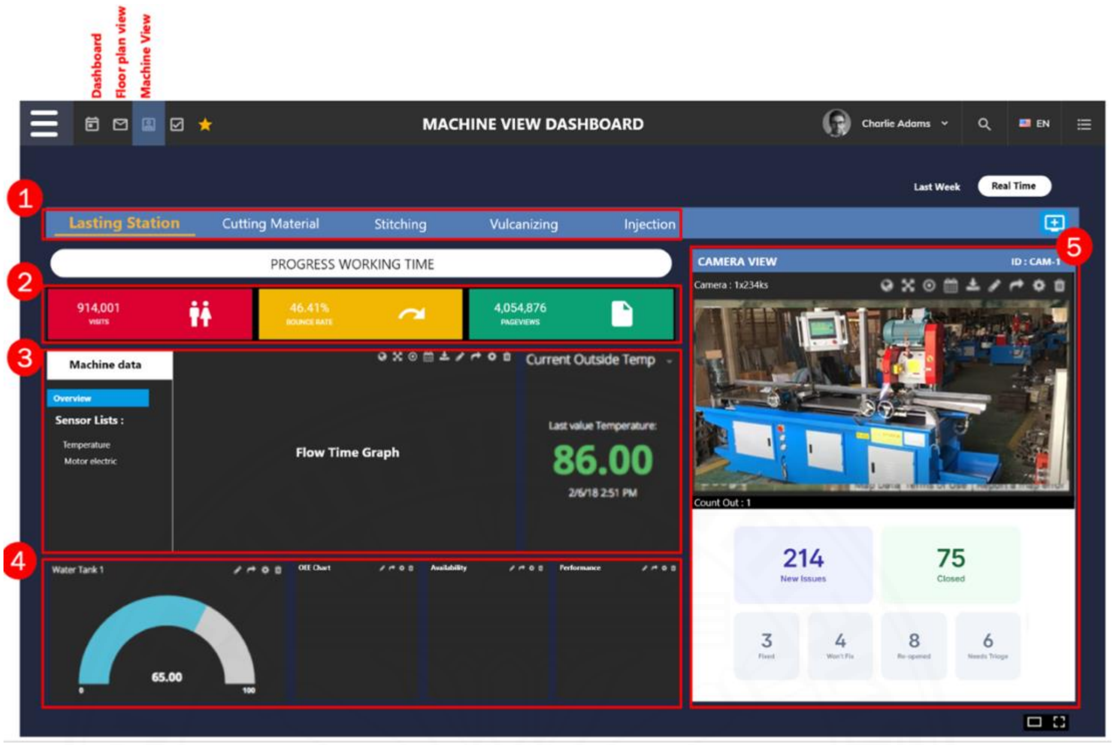
APPENDICES

APPENDIX A
REAL-TIME OBJECT TRACKING ON LASTING STATION (IN-OUT)



Dashboard 1: IoT Dashboard

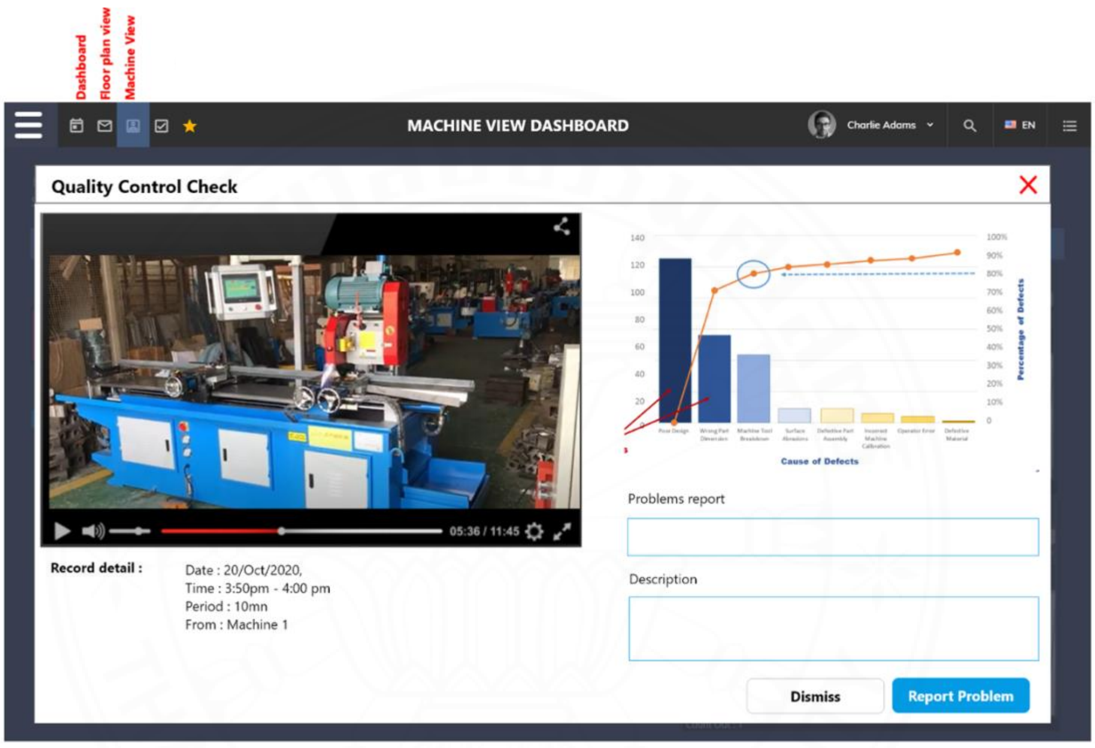
1. Users can compare planning with actual production graphs for each product SKU.
2. Users can see overall production planning compared with actual
3. Users can see the production planning graph compared with real projects monthly
4. And users can also see the machine's temperature, IoT data, their customer's overall production order, etc.



Dashboard 2: Machine view dashboard

1. Users can select each station they want to visual
2. User can see the number of production planning + Actual amount + defective product
3. User can see the machine report with cycle time + flow time graph
4. User can see the machine's OEE
5. User can use a visual camera on that station the counting product IN and OUT along with the defective product in real-time.

APPENDIX B
CYCLE TIME AND FLOW TIME WITH PARETO CHART ON QUALITY CONTROL SYSTEM



Record detail :

Date : 20/Oct/2020,
Time : 3:50pm - 4:00 pm
Period : 10mn
From : Machine 1

Dashboard 3: Machine view (Quality Control Check)

1. Users can check the problem with the Pareto chart for each station with a video clip recorded during the trial, and then they can report that problem to the plan manager.

BIOGRAPHY

Name	Tith Vong
Education	2014: Bachelor of Engineering (Computer Engineering)
	Institute of Technology of Cambodia

Publication

T. Vong, C. Jeenanunta, A. Tunpan, and N. Sirimarnkit, “The Low Computation and Real-Time Shoe Detection with Timestamp for Production Tracking in Shoe Manufacturing,” 2021 16th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), 2021, pp. 1-5, doi: 10.1109/iSAI-NLP54397.2021.9678163.