ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF ELECTRICAL ENGINERING AND COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Bs.c project

AI BASED CROP DETECTION AND HERBISIDE SPRAYING SYSTEM

Name	ID	Section
1. Anatoli Bezu	A/UR4506/09	4
2. Dawit Tegegnework	A/UR4491/09	4
3.Yadeta Abera	A/UR4387/09	1
4.Olijira Tesfaye	A/UR4371/09	3
5.Gemechis Waktole	A/UR4338/09	1

Approval

Name	*ID*	*Signature*
1. Anatoli Bezu	A/UR4506/09	_____
2. Dawit Tegegnework	A/UR4491/09	_____
3. Gemechis Waktole	A/UR4338/09	_____
4. Olijira Tesfaye	A/UR4371/09	_____
5. Yadeta Abera	A/UR4387/09	_____

Approved by:

Project Advisor: **Mr. Dawit Kefyalew**

Signature:_____

ACKNOWLEDGMENT

We would like to express our sincere gratitude to **Mr.Dawit** for inspiring us and helps us in doing and completing project work successfully. We would like to express deep regards to our beloved, dynamic and devoted **Dr.Biruk,** Head of Department, for his continual support in providing all facilities, motivation and encouragement in all aspects of completion of project work. We highly would like to express our heart full thanks to **Mr. Endale A.** and **Mr. Ayenew,** project supervisor for their valuable and constructive suggestions during planning and development of this project. They were willing to give their time so generously has been very much appreciated and as well as their dedication towards work and hardworking nature is the motivation for us. We would like to thank all our friends and families for their direct and indirect support for successful completion of this project.

ABSTRACT

Crop care is mainly concerned on sowing fertilizer and removing herbs. Most conventional techniques of removing herbs are either manually through hands or using sprayers by applying Agrochemical's uniformly to the whole field. This leads to wastage of valuable herbicide, increased costs of production, retarded crop production, environmental pollution. Sowing fertilizer is done manually, it results the wastage of fertilizer, increase cost of production, allow herbs to share fertilizer equally. Despite the fact that distribution of herbs is typically random, the proposed solution has mechanisms to spray herbicide only to the herbs and sowing fertilizer only to the crops. An Artificial Intelligence (AI) based automatic crop care vehicle system is based on the principles of removing weeds and sowing fertilizers without affecting the natural process of crops. Arduino interface inside the vehicle system allow the camera to capture the image of 40x40 inch cross sectional area to send that data to the server. Object detection process is held inside the computer using deep learning algorithms of "You Only Look at Once" (YOLO) and AlexNet, which can have a perception of identifying crops out of the whole image cross-section captured based on the training data. The computer sends information about coordinates of the field where crops and non-crops back to Arduino interface via Local Area Network (LAN). Based on the arrival data the Arduino interface decided to sow fertilizer through fertilizer nozzle or spraying herbicide based on the customers need. The valving system and Arduino controller are supposed to get power they needed from solar panel interface of the vehicle system. Vehicles reliability of doing tasks within a proper time no matter the size of the weeds helps agriculture sector to improve productivity and efficiency. Many vehicle systems at the remote area can share the computer as per the number of ports, reduces cost of controller such as raspberry pi, minimize the power consumption, reduce the load onto the vehicle. It can also help to reduce cost of chemicals and fertilizers saving from spraying over unintended areas of the crop.

Table of Contents

List of figures

List of tables

List of abbreviations

AI……. Artificial Intelligence

YOLO ……. You Only Look at Once

YOLOv3 ……. You Only Look at Once version 3

GSM ……Global System for Mobile Communications

VM……. Vector Machine

ANN……... Artificial Neural Network

RNN ……...Recurrent neural network

GAN ……...Generative Adversarial Networks

CNN ……...Convolutional Neural Network

RAM ……. Random-Access Memory

SSD……... Single Shot Multi-Box Detector

RCNN……. . Regional Convolutional Neural Network

R-FCN ……...Region-based Fully Convolutional Network

GPU ……. Graphics Processing Unit

CPU……. Central Processor Unit

VGG-Net ……Visual Geometry Group Network

RoI……... Region of Interest

RF-RCNN……. Refining faster Regional Convolutional Neural Network

ResNet ……. Residential Energy Services Network

RPN ……. Risk Priority Number

WT……. Wavelet Transform

SNR ……Signal-to-Noise Ratio

IOU ……. Intersection of Union

FPS ……Frame per Second

YOLOv2 ……You Only Look at Once version 2

GPS ……Global Positioning System RBG Red Green Blue

ReLU ……. Rectified Linear Unit

FC……. Fully Connected

Open-cv ……Open-source computer vision

MAP……. Mean Average Precision

CHAPTER ONE

1.1 INTRODUCTION

Usually in this diverse nature crops are not found lonely; there are always herbs act as a pest to crops and so that they compete for space, nutrients, water, light and hinders the metabolic process of growth of crops in the field. There are existing traditional techniques of eliminating weed from the field such as manual plucking or to spray herbicides. Desired crops should get enough fertilizers without being another resource competitor herb [1].

Our project is about caring the plant in a way of sowing fertilizers only to the crops and getting rid of herbs by spraying herbicide chemicals through the nozzle interface. By applying YOLOv3 with tiny YOLO frame work and AlexNet existing object detection mechanism. Object detection mechanism and vehicles interface are independent devices that can able to communicate via GSM module.

1.2. Background

The manual plucking weed removal method is a tedious task as it needs huge labor work. They remove it by their own hands due to this their life becomes full of discomfort; they are suffering heavy rain, children labor is abused by work in their early age, workers health condition become critical especially in desert like Humera become infected by malaria frequently, they might be sciatic, herniated disk, muscle strain, spinal stenosis, posture, degrative disk disease [2]. There is also inefficiency in productivity. It has to be waited for the weed until weeds are catchable to workers hands; the process needs much time to finish getting rid of weeds so the crop will be damaged, humans are also prone to error.

Figure 1. 1: Manual ways of removing of herbs

Conventional method removal of herbs by spraying herbicide manually affect the metabolic growth of the plant as it sprayed uniformly. Spraying chemicals also have double damage by wasting the amount of chemical that supposed to be only on the herbs. To destroy herbs selectively from the entire field and to minimize herbicide loss, herbicide spray must be done using modern technology autonomously. The first step in this regard is to classify weed, crop, and soil separately and efficiently.

(a) (b)

Figure 1. 2: (a) Traditional way of spraying herbicide, (b) Traditional way of sowing fertilizer

Weed detection based on digital imaging processing and computer vision is the most investigated and widely used techniques. Spectral features, biological morphology, visual textures, spatial contexts, and patterns present in digital images can be used to discriminate weeds and crops. These features are typically extracted by observations and experience, and then programmed/designed and translated to conventional image processing techniques. Another typical approach to handle the weed detection problem is machine learning. This approach utilizes machine learning models to extract features automatically from example images, and some complicated models not only conduct weed recognition but also can localize multiple weeds in one image.

In recent years there are a lot of object detection and classification platforms such as Support Vector Machine (VM), Artificial Neural Network (ANN), Recurrent neural network (RNN), Generative Adversarial Networks (GAN) and Convolutional Neural Network (CNN). All are deep learning approach: ANN is used for data related work and prediction. For instance, for credit card fraud detection, student identification system.

CNN is used when we have to process the image or text. Example face recognition, object detection like our project. RNN is used when we have to process text. Example: Chat bot, Sentiment Analysis. GAN is used for image process, like image augmentation. The performance existing classifiers of weed detection algorithms is executed on the Open computer vision (CV) and Keras platform using python programming language [2].

These platforms are detecting objects are Color based classification, threshold-based classification, and learning-based classification. Color and threshold-based approaches suffers loss in accuracy when the light is too high or too low, learning-based approaches promise more precision on the other hand. From learning-based approaches, two best approaches that stand out due to high accuracy are support vector machines and deep learning. Deep learning has placed itself in the first position by delivering maximum accuracy compared to all other techniques [3].

CNN: It is the major concern for object detection so that, it is commonly used for Vision-related tasks using its special kind of multi-layer neural networks, designed to recognize visual patterns directly from pixel images with minimal preprocessing. They work by passing a “filter” which is a small array over all the pixels of the image and are responsible for detecting various shapes and objects in the image. The deeper the convolutional layers are, the more complex objects the network

can detect. Combined with Pooling layers, CNNs are “shift-invariant” that means if a certain object is in a certain part of an image in the training data, the CNN is accurate enough to detect the same object in other parts of the image too [4]

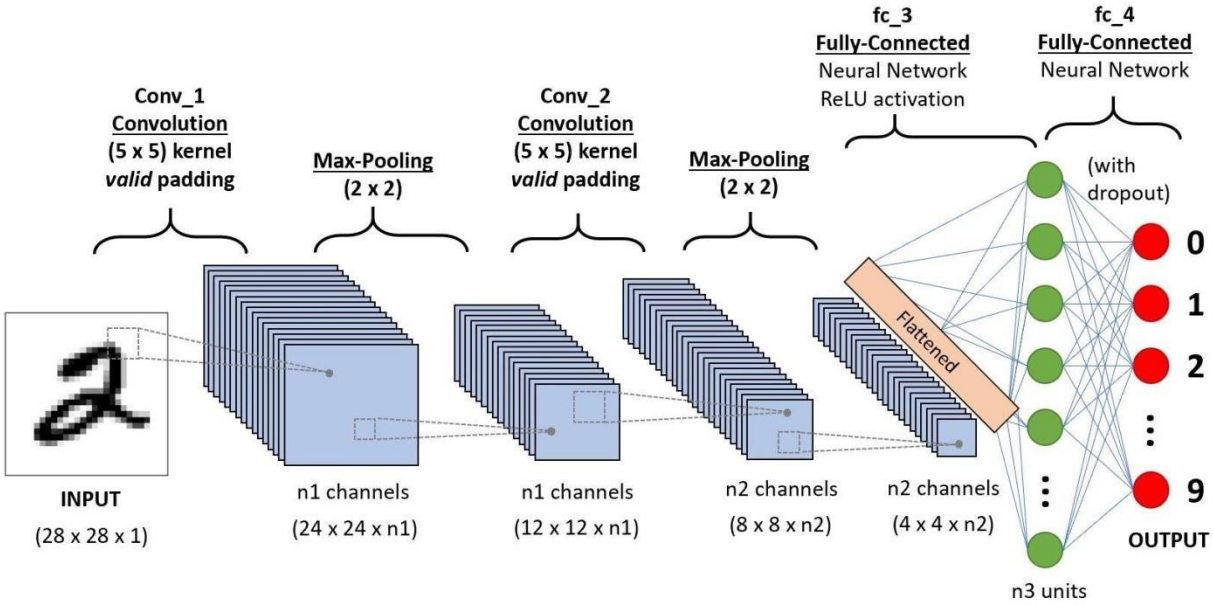


Figure 1. 3: Typical CNN architecture [4]

Activation functions the activation function is a node that is put at the end or in between Neural Networks. They help to decide if the neuron would fire or not. We have different types of activation functions just as in the figure above, but for this post, my focus will be on Rectified Linear Unit (ReLU). This function simply returns 0 if your value is negative else it returns the same value you gave, nothing but eliminates negative outputs and maintains values between 0 to +infinity. CNN using ReLU was able to reach a 25% error are six times faster than a CNN using tanh [5].

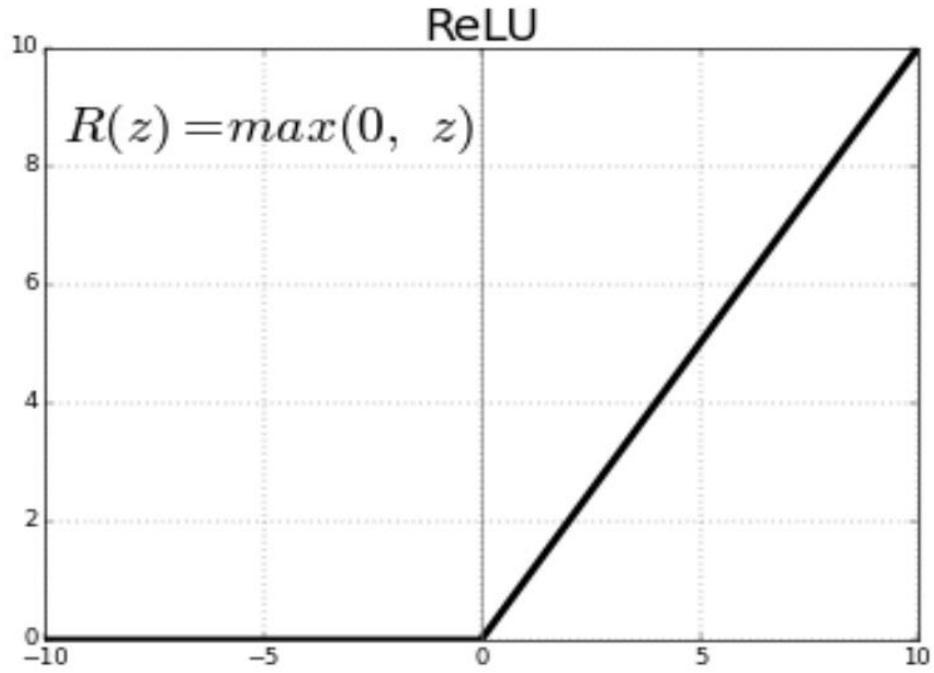


Figure 1. 4: ReLU function graph [5]

AlexNet: It consists of 5 convolution layers and 3 full connection layers, including 60 million connections, 60 million parameters, and 650000 neurons and it takes the picture of RGB three channels and the resolution of 224×224 as the input. The architecture consists of eight layers: five convolutional layers and three fully-connected layers. We had chosen AlexNet over ImageNet because for small number of images it is nearly accurate [5].

YOLOv3 is vital for our project it works real time crop detection using darknet framework at the time of training using computer which have Graphics Process Unit (GPU) and only open-CV to detect images from previously trained data. YOLO is a system for detecting objects on the Pascal VOC 2012 dataset. By the time it can detect the 20 Pascal object classes. YOLOv3 uses a variant of Darknet, which originally has 53-layer network trained on ImageNet.

For the task of detection, 53 more layers are stacked onto it, giving us a 106 layer fully convolutional underlying architecture for YOLOv3. YOLOv3 makes prediction at three scales, which are precisely given by down sampling the dimensions of the input image by 32, 16 and 8 respectively [6]**.**

We are more concerned about the herbicide must only onto herbs rather than the speed. The growth of crops may significant growth rate difference in one agricultural field. So, we prefer YOLOv3 above all other YOLO versions.

1.3. Statement of the Problem

Our project is going to solve the following problem from traditional method of caring crops. These problems make the agriculture sector lacks productivity because of four major reasons. The first one is removing herbs manually using hands must wait for the herbs until it becomes easily catchable by hand. It makes the herbs competent for resources till and it also needs very much money for labor for plucking. The second one is Spraying herbicide uniformly throughout the field manually will damage metabolic process of crops. The third one is sowing fertilizer through the whole filed will also help herbs to grow and share resources intended to be for crops. The fourth one is wastage of 35% of herbicide chemicals to unintended that crops are found and around 65% of fertilizer sowed to unintended area that herbs are likely found. There are also problems from the conventional methods of caring plants using existing object detection algorithms. Detecting herbs to care the plant will be so difficult to be accurate do the variety and different size of herbs in the field so that it may spray herbicide to the herbs that coexist nearby with the crops, it may sow fertilizer or spray fertilizer for the place where non-crops and non-herbs are found. In fact, there is a trade-off between spraying herbicides to non-crops and non-herbs wastes chemicals while using crop identification. It is less risky to identify crops instead of identifying herbs. There are also problems related to working environment such as the geographical environment (number of obstacles), mechanical design where camera and nozzle are locating, the place where Altitude the camera locate on the device, the time whether night time or day time, the center of mass of the vehicle are taken account for our project or at least mentioned in the future work.

Existing herbicide machines are based on taking raspberry pi as the brain of herb detection this makes the circuit inefficient because raspberry pi can’t be shared to another vehicle, increase the power consumption of the vehicle. So, for this typical problem we made the server computer which can able to support multiple vehicles at once as per its Random-Access Memory (RAM) capacity

1.4. OBJECTIVES

1.4.1. General objective:

The main objective of this project is to develop AI based crop detection and automated crop care controlling vehicles system in order to improve overall productivity and efficiency in the agricultural sector.

1.4.2. Specific objectives:

- To achieve real time identification of crops with high degree of accuracy.
- To design two types vehicle system structure with different proficiency algorithm.
- To optimize the amount of herbicide and fertilizer utilization by spraying herbicides only to the herbs and sowing fertilizer only to the crops.
- To minimize the cost of removing herbs and sowing fertilizers.
- To increase the rate of production process.

1.5. Significance of the project

Our project plays significant role in the academic area since the design and implement procedures are able to compromise the tradeoff about performance based on speed of CNN response and actual machine speed, the altitude where the camera found and the version of CNN uses, cost of the server computer and amount vehicles it can control. It also helps us to apply our theoretical knowledge about image processing as we made AI based crop detection. The first technique Uses Alexnet model of CNN trained by small amount of dataset can identifying weather there is plant or not within a very small cross-sectional area. The vehicle implements have camera interface in a way of covering only one crop. This is helpful for the crops which have large size of leaf such as cabbages. The second technique uses YOLOv3 with the framework of tiny YOLO makes the project trained with a small amount of data gives real time localization of the intended crops with fast frame per second. Besides increasing our academic competence this plant care vehicle is very cost effective, reliable and multi-tasking
(fertilizer sower or herbicide sprayer) for every customer who works on the

agricultural sector especially for those working as medium and higher enterprise which can able to buy one server computer which contains the trained data and open-cv platform and they can control every vehicle that are working on their field within that shared server. Some other significances of our project are mentioned below:

For User

• User can get immediate response for their problem

• It reduces time needed to identify and spraying herbicides

• It reduces human labor and cost needed during spraying herbicides

• It increases productivity by spraying herbicides only on the targets

For the developers or team members

• Obtain knowledge and experience from it.

• Use it for partial fulfillment of degree program. Gain income if it is complete.

1.6. Scopes and limitations

Our project needs have some prerequisite about how to plough and sow the crop at the beginning. It doesn't applicable for the crops which the sowing process is done randomly. It should be in proper order at least it should have the place where the roller of the vehicles can safely roll. Our project is limited data set (around thousand) despite the models we are using cans support millions of data set to be trained. It takes a lot of time to train such amount dataset and we can't easily find this amount of dataset as per the schedule. The datasets we used are only cabbages and es that are relatively easily found in this season.

This project doesn't support diversified harvesting of two or more crops in one agricultural field because of the training is only to identify one particular crop and consider the rest of the places as it herb.

Our project final achievement is designing spraying system and identification of crops due the shortage of time and the cost it needs to implement is very high. here are also miss some interface such as the vehicles self-deriving interface is not included for the time being, the valve system

doesn’t support multiple nozzle system one image, Despite the plant care is multitasking, it doesn’t work them simultaneously for both fertilizer and herbicide. But here we would like to mention that it is optional to implement or apply the processes of spraying herbicide due to the shortage of time and resources meaning we are going to develop the program that detect the crop and apply.

1.7 Feasibility study

The main concept here is that determining whether a proposed system can be made or achieved i.e. checking the acceptance of the system, formally it is deciding the phase of acceptance. We analyzed three different types of feasibility test.

1.7.1 Technical Feasibility

It states to determine whether or not the technical resources like Hardwires, Software and other assets the team have meet the requirement of the project. Implementation of the proposed system will use some materials listed below on the budget description section; programming language. By Assuming required hardware and software resources are available for the development and implementation of proposed system. Therefore, it is technically feasible.

1.7.2 Operational Feasibility

It answers the questions like will the proposed project solve the problems the team hope it will solve? And whether the solution is maintainable, reliable and affordable. This project is surely operationally feasible because the proposed AI based system is a good solution maker of the existing problem or specific solution will work in the existing system and create a good environment towards the user so when the system is reached the end users every personal or group that have a contact to the system will gain some training form the developer how they use it and accessing every segments of the system part.

1.7.3 Economical feasibility

The proposed system is economically feasible because the proposed system uses software and hardware tools that are available by low cost or even for free (open source software) except for few major hardware components.

1.8 Methodology

1.8.1. Data collection methodologies

1.8.1.1 Observation

AI based weed detection uses an image as an input so we need an image recognition tool to use and train the device. And this image has to be captured with sample input for a command and possible answers from our AI in the training part.

1.8.1.2. Data analysis

Since most of the project is done within the develop artificial intelligence (for instance solid work, Pycharm notebook, anaconda) that will recognize crop from the weeds and from this project yolo object detection is chosen. The command which is an input for our AI based weed detection we need yolo object detection to process an input and infer the result form the trained data set.

1.8.1.3. System Development Methodologies

Since most of the project is done within the computer, it uses various software's to develop artificial intelligence (for instance solid work, Pycharm notebook, anaconda) that will recognize crop from the weeds and from this project **yolo object detection** is chosen. The method of developing this project into real is that partitioning task's and test them individual if it starts to response as expected it will integrate with the other part of the project.

1.9. Beneficiaries of the project

Benefit analysis such as Intangible and Tangible and in determining cost analysis Onetime (Initial) cost and recurring costs. The following are the tangible and intangible benefits

Tangible benefit: - are those our project benefit that can convert into monetary values. For this project, we have identified the following tangible benefits.

Reduction of error and time.

Improved productivity.

Reduce the cost and labor wasted.

Intangible benefits: are those our project benefit that cannot convert into monetary values.

- Knowledge gain by project developer.
- Increasing the competitiveness of the individual.

No	Software tools	Description	Used for
1	Microsoft office	We used this to write documentation of our project	Write the program
2	Python	Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation.	
3	Libraries	These are a set of useful functions that eliminate the need for writing codes from scratch. It plays a vital role in developing machine learning, data science, data visualization, image and data manipulation applications and more.	Facilitate program development

1.10 Development Tools

1.10.1. Hardware

No.	Hardware tools	Description
1	Personal computer	We used personal computer to develop a system itself and to write and edit the documentation part of our project.
2	External Hardware Disk	We used this for keeping backup and recovery of all data that we have been using during developing time
3	USB Flash(16GB)	We use this to send and receive documents from one computer to another computer
4	Paper	We used this to take notes during data collection and group discussion

4	Integrated Development Environment (IDE)	IDE is a program dedicated to software development. As the name implies, IDEs integrate several tools specifically designed for software development. These tools usually include: an editor to handle codes, build, execution, and debugging tools. E.g. IDLE, Visual Studio Code, Atom, OpenCV	-Write codes - Debug codes - Test code output
5	Image capturing and processing software	Image editing and processing software used to manipulate the image input. We use software like Adobe photoshop.	- Process image -Edit image input

1.10.2 Software tools

1.11Resources with Cost

No.	Item	Quantity	Model/Part No.	Estimated Cost in Birr Per each Item
1	**Microcontroller (Arduino Uno)**	**1**	**Core:Atmega 2560**	1050
2	Camera module	1	OV7670	500
3	Stepper motor	4	NEMA 17	600
4	Wi-Fi module	1	ESP8266	150
5	Motor driver	4	L298n	1400
6	Cristal Oscillator	2	XTAL 12MH XTAL 24MH	80
7	Solar panel or power supply	1	18V, 50Watt	3800

9	Transistors	8	4, each IGBT part no RGPR30BM40HRTL and 4, each CMOS	120
10	LCD display	1	16x4	200
11	Resistors	6	4, 1K resistors 2, 10K resistors	60

Table: - materials and their costs for the project

N.B some of the materials mentioned in the chart are used in the prototype but we will use other alternatives for example instead of camera module we will use webcam as an alternative.

1.12 Task and schedule

NO	Tasks	Time required
1.	Building artificial intelligence	
2.	Developing simulation And material preparation selection	
3.	Developing mechanical design and material preparation	

Although we plan each partition of the work timed this way the first three is done parallel way. The followings are the plans we are going to deal with: -

- ➢ Develop object detection AI within the computer and train with several images till it can capable of identifying crops, herbs
- ➢ Doing some simulation to make sure electrical system of our project is functional as expected.
- ➢ Writing some controlling codes for the simulation, which can work in response to the decision of the AI.

1.13 Team Composition

No	Name	ID	Responsibility
1	Anatoli Bezu	A/UR4506/09	Coordinating, Designing ,analysis, requirement and resource gathering, implementation and Programming
2	Oljira Tesfaye	A/UR4371/09	Designing, Programming, analysis, implementation and Testing
3	Dawit Tegegnwork	A/UR4491/09	Requirement Gathering, analysis, design, implementation and Testing
4	Yadeta Abera	A/UR4387/09	Programming, analysis, design, implementation and Testing
5	Gemechis Waktole	A/UR4338/09	Programming, analysis, design, implementation and Testing

CHAPTER TWO

2. LITERATURE REVIEW

These days, there are some state of art works about detecting objects and doing some automation based on the intention of the customer. Related Object detection models are group these algorithms into two major categories based on how they perform their tasks. Fundamentally they all are going to apply similar steps: make a prediction, receive a correction, and adjust the prediction mechanism based on the correction, at a high level, they all trying to make their object detection quite similar to how a human learns.

2.1. CNN

The first group of algorithms is based on classification of objects. This kind of CNN is composed of algorithms based on classification and working into two stages. For CNN first select the exciting parts of image in the preprocessing stage and then they classify objects within those Regional Convolutional Neural Networks (R-CNN), Faster R-CNN, Fast R-CNN, Mask R-CNN, R-FCN, Single Shot Multi-Box Detector (SSD) and RetinaNet, which is usually too slow to be applied in real-time situations. The developments of advanced convolutional neural networks, deep learning techniques, and the increase of the parallelism processing power offered by the GPU such as the more powerful and accurate NVIDIA GTX 1070 Ti, and less powerful with relatively low precise and accurate NVIDIA Jetson TX2.

R-CNN To bypass the problem of selecting a huge number of regions. Proposed a method where we use selective search to extract just 2000 regions from the image and he called them region proposals. Therefore, now, instead of trying to classify a huge number of regions, you can just work with 2000 regions and warping it. It then computes the CNN feature and classify images [9].

$$x = (1 + \Delta x) \cdot x$$

$$\hat{y} = (1 + \Delta y) \cdot y$$

$$\hat{w} = exp(\Delta w) \cdot h$$

$$\{ \hat{h} = exp(\Delta h) \cdot h$$

From the formula *x, y, w, h* corresponds to the horizontal and vertical coordinates of the Anchor center point and the width and height values respectively; Δx, Δy, Δw, Δh are the corresponding correction values obtained after RPN regression; $x^$, $y^$, $w^$, $h^$ are the corrected values. After the

Anchor is revised, non-maximum suppression is used according to the confidence of the foreground and background, and the more accurate Anchor is selected as the candidate area. But it takes a lot of time to finish for instance 13s/image on a GPU, 53s/image on a CPU, VGG-Net and it makes 7x slower than the regular one. Warping will also change the appearance of the input images.

Fast R-CNN follow existing R-CNN, starting from a Convolutional neural Network initialized with discriminative pre-training for ImageNet classification. They consider architectures that have several convolutional (conv) and max pooling layers, followed by a region of interest (RoI) pooling layer, and then several fully-connected (fc) layers. The new thing from fast R-CNN is Pre-trained networks so all of the region of interest end with two sibling layers, one that outputs SoftMax probability estimates and another layer that outputs four real-valued numbers for each of the *K* object classes perform regression. These 4*K* encode values refined for regression of the bounding boxes for each class. This basic issue remains limitation of this, is due to slow detector training and testing. It can be improving detection quality by reducing spurious false positives meaning improve proposals per image.

The Faster R-CNN is another state-of-the-art CNN-based deep learning object detection approach. In this architecture, the network takes the provided input image into a convolutional network which provides a convolutional feature map. The common selective search algorithm of identifying the region proposals is made by previous iterations [10, 12]. The authors proposed a refining block for Fast R-CNN which merge the block and Faster R-CNN into a single network of RF-RCNN to detect different kinds of road view that consists of high-resolution street images. The proposed system is based on a novel two-stage cascade multi-scale proposal generation networks to address the problem of Faster R-CNN. In the proposed system; a separate network is used to learn and predict these regions. The predicted region proposals are then reshaped using a region of interest (ROI) pooling layer, which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. So that adopting Faster R-CNN's network architecture for object detection was present, and it increase anchors' density and feature maps' resolution.

Faster R-CNN has the best comprehensive performance in series of target detection algorithms based on region convolutional neural network. However, it presents a low detection precision in case of multiple and small targets it can't accommodate detection under different light environments for cracks in varying lengths**.** To solve the problem of insufficient samples of dam cracks, transfer learning methods are utilized to assist network training and data enhancement. ME-Faster R-CNN, ResNet-50 network is proposed adopted to extract features of original images and obtain the feature map. Then, the features map is input into multi-task enhanced RPN module to generate the candidate regions through adopting the appropriate size and dimension of anchor box to be processed and detected the dam cracks. ME-Faster R-CNN is the best, to have better in the precision of crack location detection it has limited on having better recalling different target detection algorithm.

The paper made by two Korean guys introduce shot category detection based on object detection using convolutional neural networks (CNN). They made a data set collection using a Shot which is a portion of movie scene captured by the camera, which composes different aspects of image set. The basics of Shot category is the movie angle is decided by camera's distance so that they divided the whole picture into nine class of shots long shot, medium shot, full shot, close up, intro and credit, big close up, waist shot, up shot, Bust shot. This kind of Object detection finds consistent sizes of the image rather than basic image segment annotating. It is different from of segmentation, featured by several elements, images are captured by color pixels because itaffects image resolution regardless of specific objects. In case of multiple objects inside one picture, the effect of varying the shot style becomes good when shot categories are considered that.
Constant size of object detection is efficient for shot category to fill in specific parts of the image.

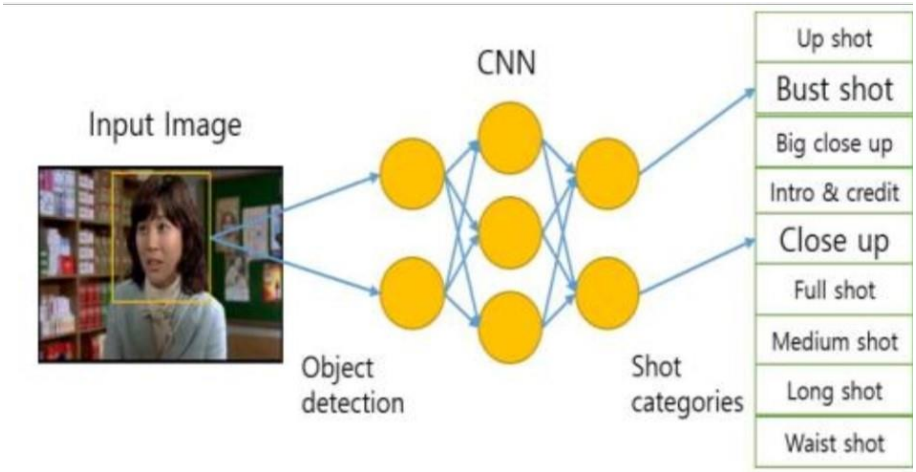


Figure *2. 1*: Procedure of shot category detection on CNN.

In addition to slower real time processing using existing CNN model mentioned in the above, it can also have difficulty If the images from the dataset are contain some degree of tilt or rotation then CNNs usually have difficulty in classifying the image. This can be solved by adding different variations to the image during the training process otherwise known as Data Augmentation. Another weakness is because of If the CNN takes an image along with some noise it recognizes the image as a completely different image whereas the human visual system will identify it as the same image with the noise. This also proves that CNNs are using very different information from a regular visual system in order to recognize images. So that it needs a diverse amount of dataset to train fully.

2.2. AlexNet

AlexNet is well known for the very rapid down sampling of the intermediate representations through stridden convolutions and max-pooling layers. The last convolutional map is reshaped into a vector and treated as an input to a sequence of two fully-connected layers of 4096 units in size. The paper called “Modulation Recognition using Wavelet Transform based on AlexNet” is an AlexNet based deep learning technique applying for modulation recognition method is able to implement the complex modulation recognition in non-cooperation communication systems In the image preprocessing step, the first thing they do is convert the time-domain diagrams of different complex modulated signals to the spectrogram images by using wavelet transform (WT), which is applied to each sub image, a 2-dimensional image is transformed to a weighted shape matrix IO secure invariance in translation. Scaling, rotation and split into four dyadic sub images in order to further explore its details into different directions and to achieve image sub bond decomposition.

This technique helps them to make an efficient and effective 2-D image fractal algorithm is used to extract each sub band coefficient as a feature for classification. In the training step, with the feature extraction of AlexNet model to the input spectrogram images, we can attain the training model, which can be used to classify the different modulations of signals using eight digital modulated signals for recognition. The simulation result of this research shows that the classification accuracy of eight signals is almost 100% at higher signal-to-noise ratio (SNR). Compared with using initial time-domain diagrams without wavelet transform as input this method is of classification of objects accuracy is significantly improved.

The main drawback of AlexNet applied by many literatures has limitation on removing any of the convolutional layers will drastically degrade AlexNet's performance. AlexNet is prone to Over fitting Problem, it has 60 million neurons will cause over fitting. Object detection using Shot category detection has limitation about wasting too much time without having a different result.

2.3. YOLO

The algorithms in the in the second group is based on regression - they scan the whole image and make predictions to localize, identify and classify objects within the image. Algorithms in this group, such as YOLO, are faster and can be used for real-time object detection.

The state of art real life application of YOLO is differing because of the same time bounding box predictions and class predictions can be done by YOLO. The input image is sliced up into $S \times S$ grids and the bounding boxes are included in each grid cell, each with a score of confidence. The probability of an object exists in each bounding box is known as confidence and is defined as:

$$C = P_r(object) * \text{IOU}_{pred}^{truth}$$

IOU is the intersection union which describe the measure of two bounding boxes for accurate detection.

The IOU should be near to 1, so that the predicted bounding box lies closer to the ground truth.

IOU can be expressed as:

$$IOU = \frac{\text{Area of overlap}}{\text{Area of Union}}$$

CPU based object detection (YOLO) doesn't need any GPU it is enough having CPU and good RAM to train objects within a faster time. Therefore, in this study we use the YOLO object detection model to detect objects in a real-time manner because the processing speed and accuracy in detecting objects are very high. This model is able to execute videos from external source or from webcam with 10.12 frame per second (FPS), 80-99% confidence and with 31.05% mean average precision (mAP) that is suitable for real time application within low cost and less effort and as compared to other CNN architectures it has small frame per second is typed as fast. However, its precision becomes very small.

The paper called" YOLO based Detection and Classification of Objects in video records" works on the task usually encounters slow processing of classification and detection or the occurrence of erroneous detection due to the incorporation of small and lightweight datasets using YOLO detection and classification approach YOLOv2 for improving the computation and processing speed and at the same time efficiently identify the objects in the video records. This YOLO based detection and classification YOLOv2 version use of GPU to increase the computation speed and processes at 40 frames per second which is better than the previous version explained above.

Bringing the above state of art concepts, the following reviews of literatures are mainly concentrate toward weed detection algorithms used, new weed detection technologies have been developed to improve the speed and accuracy of weed detection, mitigating the conflict between the goals of improving soil health, and to achieve sufficient weed control for profitable farming. A literature review is conducted in this section, and all the emerging weed detection techniques and methods in the recent 5 years are organized.

Image processing is one of the common tools used in weed detection; its typical procedures include pre-processing, segmentation, feature extraction, and classification examined wavelet texture features to verify their potential in weed detection in a sugar beet crop. A discrimination algorithm was designed to determine the wavelet texture features for each image sub-division to be fed to an artificial neural network. Co-occurrence texture features were determined for each multi-resolution image created by a single-level wavelet transform. A neural network was finally used to label each sub-division as weeds or crops. Results showed that the wavelet texture features could discriminate weeds from crops effectively. The paper written by Indian guys which had a title named

“Implementation of artificial intelligence in agriculture for optimization of irrigation and application of pesticides and herbicides” describe application of artificial intelligence in agriculture in general implementation of automation in agriculture, the weeding systems through the robots and drones.

Technically they advised to make a robot for irrigation purpose. Artificial intelligence helps to analysis the moisture of the soil content through its dielectric constant in other word dielectric constant of the soil with different moisture is different and analysis the rain drops and the robot act based on the given data to give information to irrigation canal. The second section of their work is madding a robot or drone which can actually identify the plant weather it is weed or crop by analyzing their moisture content (moisture content of each plant is different). But the paper is just limited on showing that how much efficient system they develop. For example, the second section of their work is very impractically since the robot must analysis the moisture content of each plant.

And they are not actually made one, instead they direct possibilities about how to identify weather the farming land needs extra water or not and identify weather the plant is weed or crop.

YOLO versions of CNN algorithms are generally used as object detection and localization algorithms. It is an object detection technique in which the detection process is considered as single backsliding problem which takes an input image and generate the confidence level of each object in the image it is the descendent of primitive YOLO algorithm. Though it is not the most accurate algorithm it is a very good choice for real-time object detection without losing to much accuracy.

This algorithm uses a single convolutional neural network (CNN) to detect multiple objects with in the image. It means it divides the image into multiple grid box or region and predict the bounding box and the associated probability or confidence level for each region. The confidence level of each region infers the precision of the localization of the objects. Most of the grid boxes and the bounding boxes are removed accounting to fewer threshold value leaving behind the particular class of object which it is trained to detect.

The state of art weed detection uses color image is converted to binary by extracting the green parts of the image. The number of white pixels present in the region of interest is determined and regions

with higher white pixel count than the predefined threshold are considered as weeds which contains the lesser amount of ROI (region of interest) because of assumption of weeds have narrower leaf than that of normal plants (crops).

The hard ware implementation of the herbicide is stored in a container fitted with water pump motors attached to spray nozzles by the order of Raspberry-Pi. Once the weeds are identified, a signal is sent from Raspberry-Pi to the motor driver IC controlling the water pump motors to spray. All the training process is go through preprocessing (First, little bit of variance is need to be added to the data since the images from the dataset are very organized and contain little to no noise or resizing the image), splitting the dataset to save time by training shorter dataset, building convolutional network.

It has problematic on assumption of weeds will have narrower leaf than crops even if it is true there is also problematic on which weeds may found densely and cover large place so it becomes hard to train the machine independently so there might happen a missing of weeds. The cost of raspberry-pi is so high and there is also a waste of materials which might be useful for other parallel work since raspberry-pi robust. It is also easier to work using YOLO instead of other CNN for those which needs real time process.

The paper called" Development and evaluation of a low-cost and smart technology for precision weed management utilizing artificial intelligence" written by American guys in 2019 provide a designed a prototype of spray machine which have nozzle with valve having relay controlled by Arduino controller based the herb detection by the high graphics card computer (NVIDIA GTX 1070 Ti GPU. The GTX 1070 Ti GPU) using yolo object detection which implement convolutional neural network.

The vision-based detection system and the overall precision sprayer system are calculated using the collected evaluation metrics upon detection, the detected box center where weed relative coordinates are used to calculate the nozzle position to be sprayed.

The last and best qualities of this work is its valve control system An Arduino script was developed to read the serial data coming from the computational unit containing the "target to be sprayed" (weed) position and the vehicle speed calculated from the GPS dat

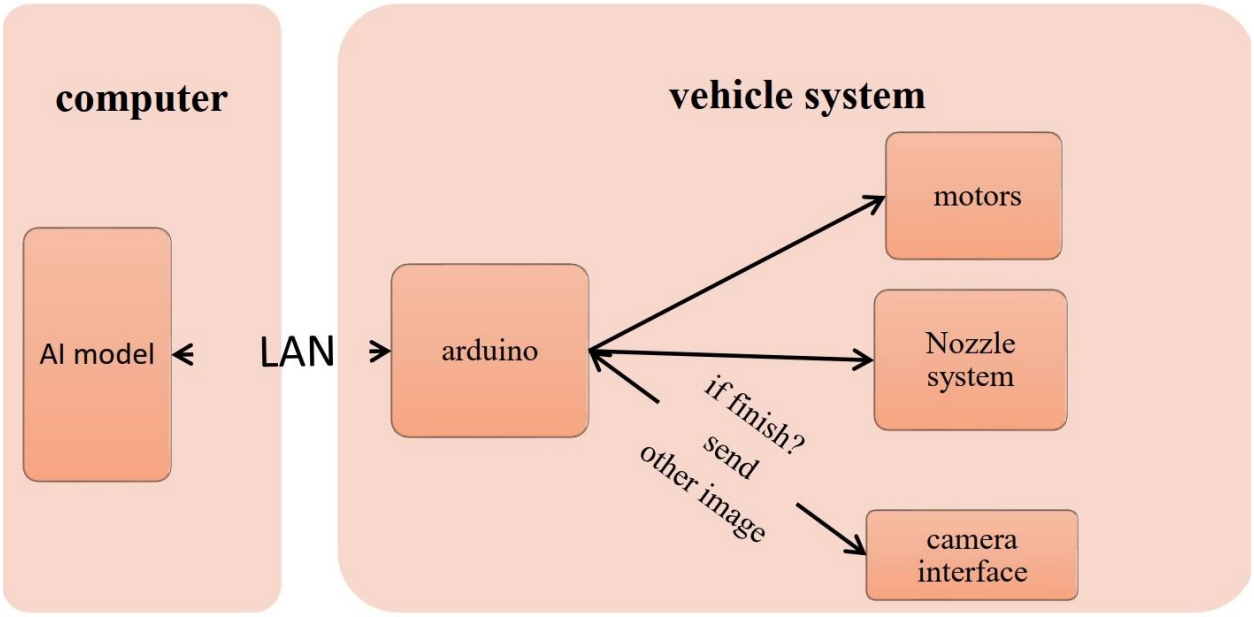
CHAPTER-3

METHODOLOGY AND SYSTEM DESIGN

3.1. Methods

Our crop care vehicle system has two major designs. The first design of the vehicle can cover is 40x40 inches of cross-sectional area and the second one covers the area equal to or slightly greater than one image cross-sectional area depending on the image type in our case cabbage.

The core difference is that the fist method uses YOLOv3 algorithm to detect the entire crop inside that cross-section and can identify or the location of each crops which is fallen inside the cross-section area. The second one is based on AlexNet it can identify whether there is crop or not within the given cross-sectional area but, it does not localize the exact location of the crop inside the image.



General block diagram for the two methods

The vehicle system capture image inside a defined cross-sectional area and send the captured image to the computer or any server/ system which is capable of performing image processing via configured LAN system or any wireless communication system. Then, the computer/server perform image processing algorithm on the image that have been sent from the vehicle system. The server performs two kinds of image processing algorithm depending on different situations.

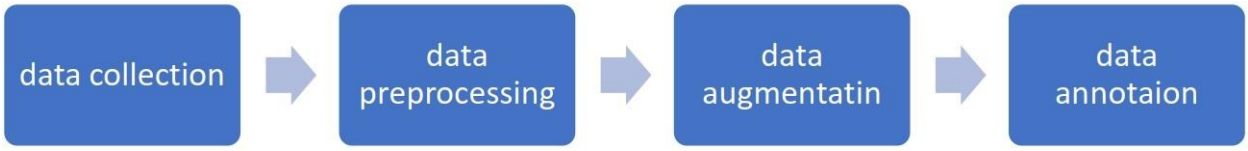
In case of method, I the AI system performs two tasks which are image detection and localization. On the other hand, method II determines whether crop is found or not in a given cross sectional area but it doesn't know exact location of the crops. The main criteria that differentiate the two kinds of image processing is based the cross-sectional area covered by the vehicle system.

3.2. Method 1: crop care vehicle controlling system based on crop detection and localization by YOLOv3

For the image processing is done by YOLOv3 Architecture which uses CNN algorithm for object detection and localization. It is an object detection technique in which the detection process is considered as single backsliding problem which takes an input image and generate the confidence level of each object in the image it is the descendent of primitive YOLO algorithm. Though it is not the most accurate algorithm it is a very good choice for real time object detection without losing to much accuracy.

3.2.1.1. Dataset preparation

Images should be prepared (preprocessed) in a certain way before staring training process. Figure below shows all the steps of data preparation.



General preparation procedure for YOLO object detection

Data collection: Our dataset is made up of 2000 images, 1000 for Cabbages. The RBG camera takes images from different perspective, and saves these images in a resolution of 1444X2192 in .jpg format.

Data preprocessing: Our dataset contains images with different resolution and quality so we need to resize the images in with the same width and height before the training.

Data augmentations: In order to combat over fitting problem, which it is imposed if the dataset is too few to learn from, rendering you unable to train a model that can generalize to new data. Data augmentation generates more training data from the existing training samples by augmenting the samples via a number of random geometric transformations like rotation, translation, shearing

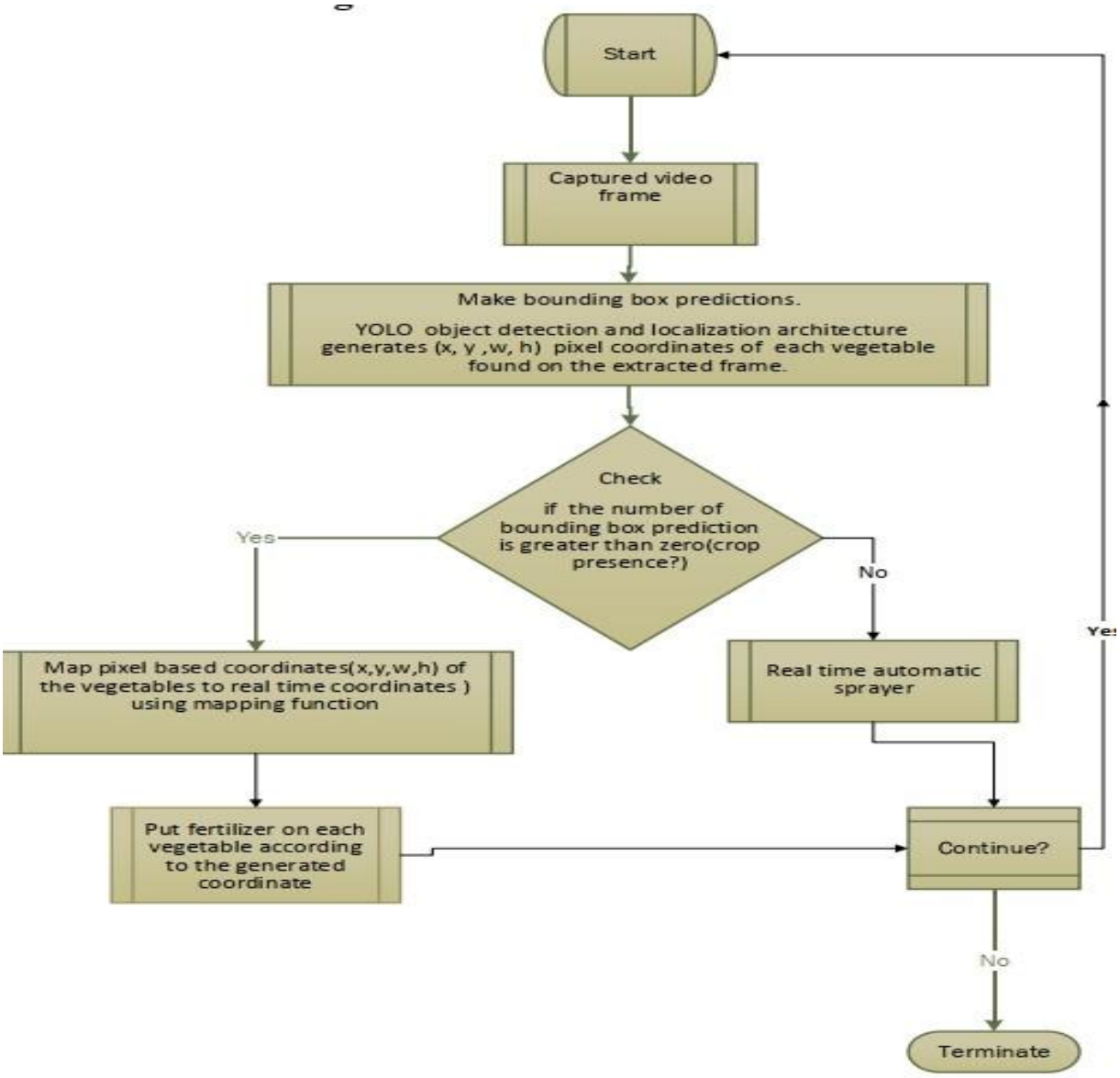
zooming, and horizontal flipping. The main aim is to exposed our model to more aspects of the data and generalize better.

Image annotations: Image annotation increases precision and accuracy by effectively training these systems. The crops annotated by rectangular box by software called dataset_ tool master labeled by yolo and pascalVOC to generate txt or xml files respectively necessary for the training.

3.2.1.2. Step two: Training

We have trained YOLOv3 Full network which uses the Darknet-53 Architecture for feature extraction with 100 images dataset of one class with the parameter of learning rate (lr) 10-5, batch size of 16, and image size of 608X608 for 2000 epoch for 6 hours in the Google colab. The computational development here presented was performed one cloud provided by Google colab with CPU at 2.8 GHz, 16 GB of RAM, NVIDIA-SMI 460.27.04 Driver Version: 418.67 CUDA Version: 10.1.

3.3. Flow chart for image detection and localization



Flowchart of operation using method one

The above flowchart is explained procedurally as follow:

➢ First step is image acquisition which is done by webcam or any camera which is mounted facing downward on the actuator or herb-sprayer vehicle at a certain height, h.

➢ Then, the AI-algorithm extract each frame from the input video which is acquired by the camera.

➢ Then, the extracted frame is preprocessed like resizing, normalizing of each pixel by 255. ➢ Then the preprocessed image or frame is fed into image localization algorithm for bounding box prediction.

➢ The bounding box prediction generates the x-axis, y-axis, width, heights (x, y, w, h) of each vegetable on each frames of the input video.

➢ The generated coordinates from bounding box prediction are the pixel location of each vegetable on the extracted frame so, we need a mapping function. $(x , y, w, h) \rightarrow (x', y' , w', h')$ which maps the pixel coordinate (x, y, w, h) to the real time coordinates (x', y' , w', h').

mapping function: The generated coordinates from bounding box prediction are the pixel location of each vegetable on the extracted frame so, we need a mapping function $f(x', y' , w', h') \rightarrow (x, y, w, h)$which maps the pixel coordinate (x', y' , w', h'). to the r The actuator put fertilizers to each vegetable according to the real time coordinate information which is generated by the image detection and localization algorithm. time coordinates (x, y, w, h) mapping function can be calculated as follow:

Let the camera mounted at certain height H and the capturing Area coverage of the camera be WXL.

$$\boldsymbol{x = \frac{x'}{W} \cdot w'} \qquad \boldsymbol{y = \frac{y'}{H} \cdot h'}$$

$$\boldsymbol{w = \frac{y'}{W} \cdot w'} \qquad \boldsymbol{h = \frac{y'}{H} \cdot h'}$$

Once we calculate the real time coordinates of each vegetable on each frame, we fed it to the actuator which is x-y axis mover

3.3. Method 2: crop care vehicle controlling system based on image classifier using AlexNet

CNN based crop classification is done based on convolution of input images, and detect features based on filters that are learned by the CNN through training. In this case, AlexNet version of CNN architecture for detection of the presence of crop or not. The training is based AlexNet which has using CPU version and there is batch normalization after each convolution layer. Since, our dataset is very small we don't need GPU.

The modified CPU version of AlexNet has input layer and 5 convolutional layers followed by two fully connected layers at the end we have output layers with 2 SoftMax channels. The feature extractor of ConvNet is a sequence filtering and pooling layer parts of convolutional layer. The activation function in this case is ReLU=max (0, X). After several Convolutions and pooling operation, the input image size is reduced and more complex features are extracted. Finally, a small feature map that is extracted by the convolutional layer is merged or squashed into one dimensional vector and fed into fully connected (FC) layers for classification.

3.4.1. Step one: Data preprocessing

✓ Resizing: the image resized into the size of 277x277

✓ Augmentation: perform different geometrical transformation like rotation, translation, horizontal and vertical flipping, zooming of random object in a particular image, and some other transformation to combat over fitting problem.

We have split our image datasets into 3 parts which are training, validation, and test.

• Training: there is a total of 920 training dataset. 460 for each class

• Validation: there is a total of 100 validation datasets. 50 for each class

• Test: there are a total of 100 test datasets. 50 for each class

Training parameters

✓ learning rate: 1e-4

✓ Number of epochs: 10

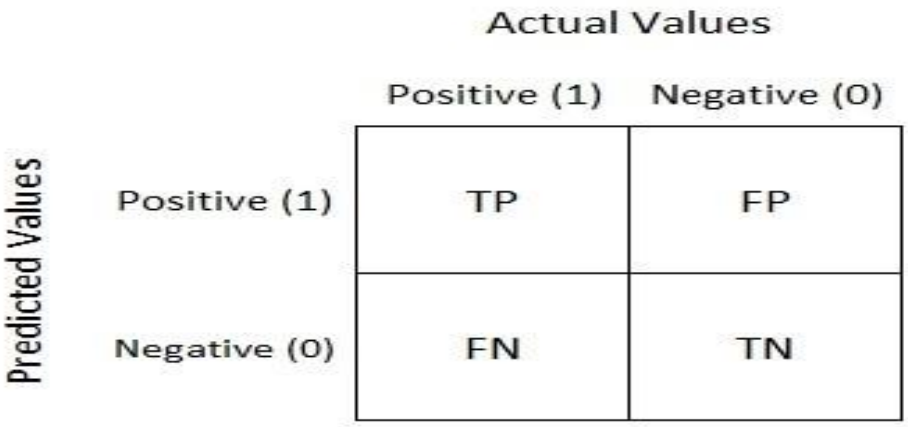
✓ Loss function: binary cross entropy ✓ Optimizers: RMSprop optimize

✓ Not trainable parameters: 0

✓ Batch size: I used batch size 64 which means 64 images will be passed to the network in single pass.

Evaluation metrics for this project: It is used for evaluation of the performance of our CNN model is Confusion metrics: is one of widely used for accurately evaluation the classification models. For a classification problem with class C, the confusion matrix M is C x C metrics with Mij. And its interpretation is as follow:

• If $\neq$, it means the actual class labels are i but they are classified as j • If $=$, it means the actual class are I and they are classified as i



The model used as a confusion matrix

The interpretation of confusion matrix is explained as follow:

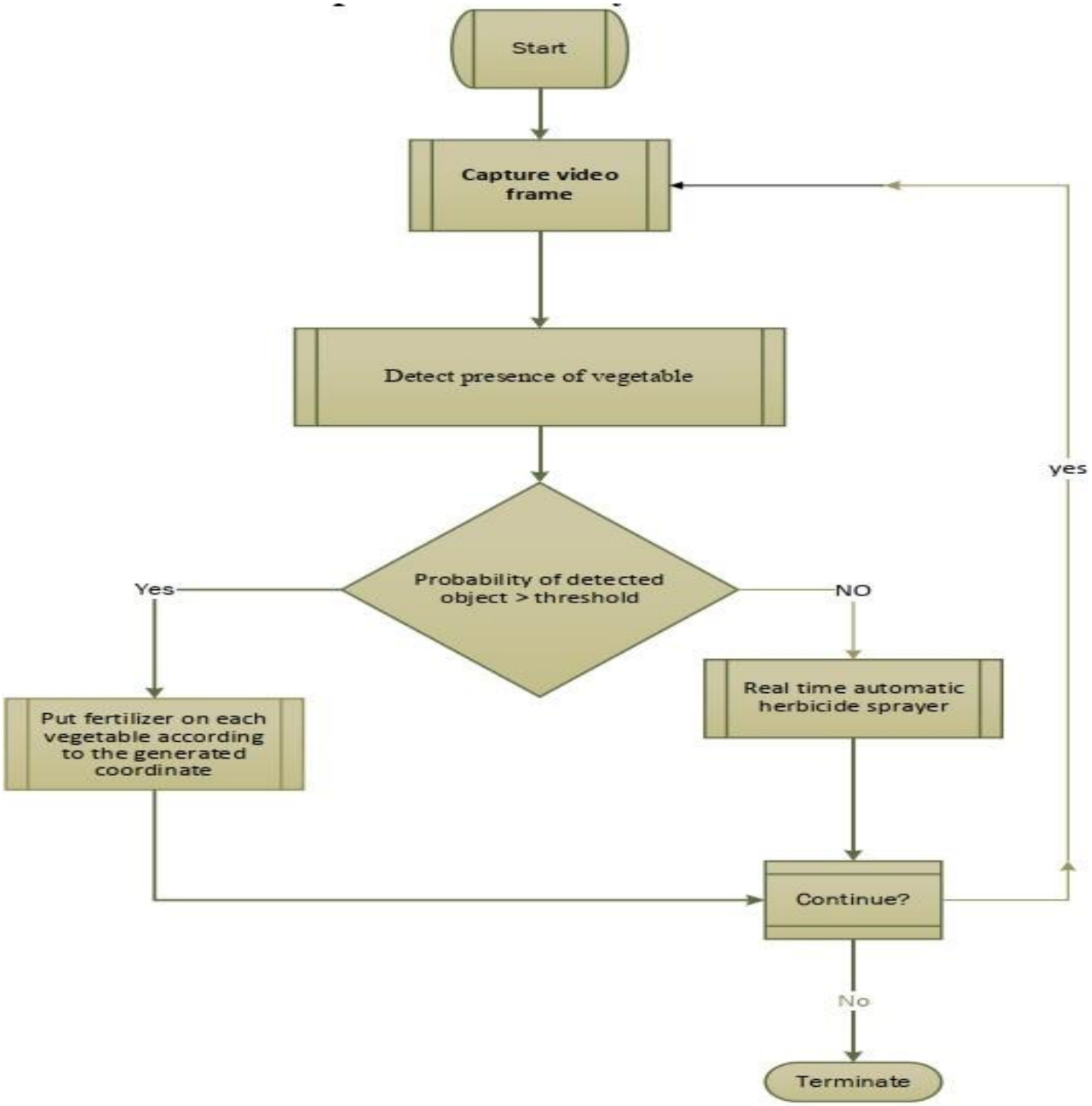
• True positive: shows the number of samples whose actual class is one and classified as one.

• True negative: shows the number of samples whose actual class 0 and classified as 0.

• False negative: shows the number of samples whose actual class is 0 and classified as 1.

• False positive: shows the number of samples whose actual class is 1 and classified as 0.

3.5. Flow chart for crop classification



- It captures the input image from the vehicle system. First step is image acquisition or extraction of frames from the video stream which is done by webcam or any camera which is mounted facing downward on the actuator or herb-sprayer vehicle at a certain height, h.

- Preprocess the extracted frame o images which includes resizing the image, normalizing the images pixel between 0 and 1 dividing each pixel with 255.
- Fed the preprocessed images to the proposed CNN model for prediction of its class label as 0 for crop or 1 for herb. If the predicted class label of the images is 0 and its confidence probability is greater than the threshold value. The threshold for this particular method is 0.9 it means crop is detected. If the predicted class label is 1, it means herb is detected.
- The actuator acts according to the predicted value driven from the proposed CNN. If 0 is sent it put or inject fertilizer to the crop. If 1 is sent it spray herbicides to the herb.

CNN architecture

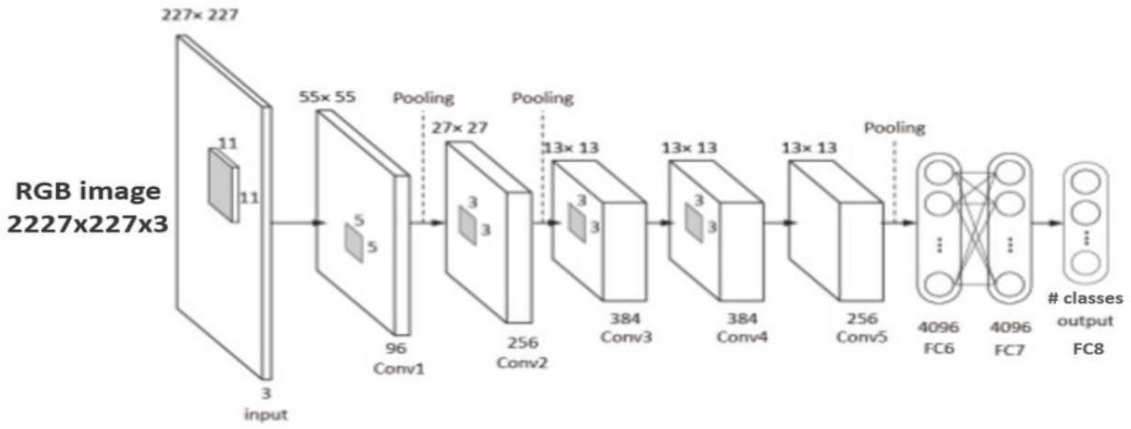


Figure: CNN architecture

We have used one of the most famous CNN architecture for detection of the presence of crop or not which is AlexNet. We have modified the AlexNet from GPU version to CPU one and we also add batch normalization after each convolution layer. Since, our dataset is very small we don't need GPU. It is the modified CPU version of AlexNet.

The AlexNet has input layer and 5 convolutional layers followed by two fully connected layers at the end we have output layers with 1 sigmoid channel and the function inside each layer is explained as follow.

The input layer: The input image is 277x277 RBG image.

First Layer: The input for AlexNet is a 227x227x3 RGB image which passes through the first convolutional layer with 96 feature maps or filters having size 11×11 and a stride of 4. The image dimensions changes to 55x55x96. Then the AlexNet applies maximum pooling layer or subsampling layer with a filter size 3×3 and a stride of two.

The resulting image dimensions will be reduced to 27x27x96.

Second Layer: Next, there is a second convolutional layer with 256 feature maps having size 5×5 and a stride of 1. Then there is again a maximum pooling layer with filter size 3×3 and a stride of 2. This layer is same as the second layer except it has 256 feature maps so the output will be reduced to 13x13x256.

Third, Fourth and Fifth Layers: The third, fourth and fifth layers are convolutional layers with filter size 3×3 and a stride of one. The first two used 384 feature maps where the third used 256 filters. The three convolutional layers are followed by a maximum pooling layer with filter size 3×3, a stride of 2 and have 256 feature maps.

Sixth Layer: The convolutional layer output is flattened through a fully connected layer with 9216 feature maps each of size 1×1.

Seventh and Eighth Layers: Next is again two fully connected layers with 4096 units. **Output Layer:** Finally, there is a sigmoid output layer ŷ with 2 possible values.

Table 1: Summery of each layer

Layer		Feature map	Size	Kernel size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55x55x96	13x13x384	4	Relu
2	Max pooling	96	27x27x96	3x3	2	Relu
3	Convolution	256	27x27x256	5x5	1	Relu
4	Max pooling	256	13x13x256	3x3	2	Relu

5	Convolution	384	13x13x384	3x3	1	Relu
6	Convolution	384	13x13x384	3x3	1	
7	Convolution	256	13x13x256	3x3	1	Relu
8	Max pooling	256	6x6x256	3x3	2	Relu
9	FC	-	9216	-	-	Relu
10	FC	-	4096	-	-	Relu
11	FC	-	4096	-	-	Relu
12	FC	-	1	-	-	Sigmoid

Components and their operations we used for the convolutional layers

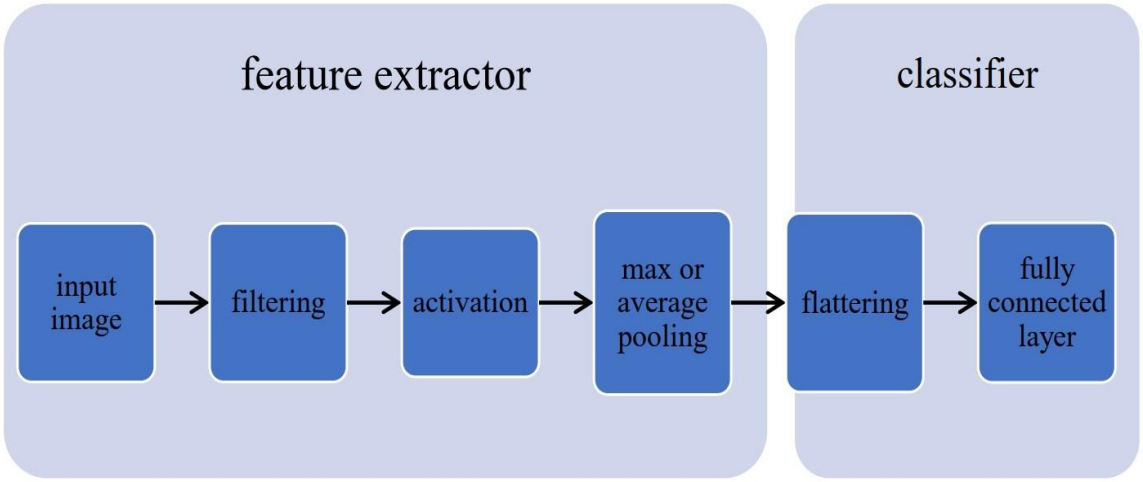


Figure: Operation of our model

Input layer: the input to this layer is images. The images are fed in a form of batches as four dimensional vectors or tensors which is (, *h*, ,) where the specifies the image index, **h** and **w** represents height and width of the image and the fourth dimension represents the channel of images, which is 3 in case of RBG and 1 in gray scale images.

Convolution Layer: This layer is the most vital or the heart of any CNN. it generates new images called feature maps by performing a series of 3D convolution, activation and pooling operation. the input to the convolution layer is multidimensional image I E or another feature map form the previous layer.

Filter banks: If we have multidimensional or multichannel image I E and a banks of D filters K, 12 and biases , one for each filter. Then, using this filter banks we perform convolution operation on the input image or feature maps.

The output of feature maps after filtering operation with filter banks will have the following forms: $\square \in {}^{-}\mathbf{k}^{+}\,{}^{-}\mathbf{k}^{+}$, from this we infer that the third dimension of the filter bank must be equal to the input feature map channel and the height and width of the feature map is depend on the height and width of the filter K. but, we can choose any positive integer value for the fourth dimension of the filter, K and the number of output feature maps is the same as the number of filter banks, which is the fourth dimension.

Activations: It simply consists of point wise Relu or tanh(.) function applied to each pixels of the feature maps which is generated from filtering operation. The ReLUs layer adds non-linearity in the network and provides non-saturating gradients for positive net inputs. In this case the dimensions of the feature maps do not change.

Pooling: For this model it is selected one pixel from the neighboring pixels as a single representative value so that we can reduce the trainable parameters since the neighboring pixels in each feature maps are considered to be highly correlated; generally, it is non-linear down sampling of feature map.

There are two types of pooling those are:

- **Max pooling:** In max pooling, the maximum pixel intensity of a locality is taken as the representative of that locality.
- **Average pooling**: the average of the pixel intensities around a locality is taken as the representative of that locality

3.6. Advantages and disadvantages of each methods

The first method is advantageous for its high level of prediction and localization with the bounding box around the target crop. It makes efficient when it finds shadowed, partial viewed, different

stage of growth, damage level of the crops will be bounded by rectangular box. It is robust as works within both GPU and CPU based processing. It is good for those who have large amount of agricultural investment because of its size and many vehicles can operate within one server which have multiple port to support them. Its disadvantage is training procedure is very complex and takes longer time to finish. It needs a greater synchronization between number of dataset and the number of epochs needed for the training.

Method two is advantageous for those who can't cost much many for vehicle system. As it needs simple vehicle system it is more advantageous to sowing fertilizer. It takes short time to train within a CPU. Method two designed to find whether crops are there or not in a certain cross-sectional area. This makes unnecessary fertilizer if there is not crop as well as no herbs in that area. Method two is also disadvantageous, if it loses the pattern of ploughing with straight line, it will find half of the crops it become as wasteful as it determines there is crop it needs fertilizer or skipping from spraying herbicide.

3.7. Software used

The few frame and software used for our works for running algorithms and constructing circuits are listed below:

- TensorFlow: is one of a deep learning framework developed by Google which is used for designing, building, and training the model. But it needs huge amount of GPU power and is more flexible on Linux operation system.
- Darknet: It is YOLO based on Linux. And it runs faster on GPU base computer.
- Dark flow: It was made by adapting darknet to Tensor flow and works very fast in GPY based computers.

It also runs in CPU based computer but it is very complex task to install it
in windows and is very slow.

- OpenCV: it was built by Intel and also it has a deep learning framework. It works only in CPU and installation in windows is easy.
- Proteus: it is a simulation software in this case it can construct electrical circuit for tasting the developed image processing with Arduino ide and PyCharm notebook.
- Virtual communication port: to send serial communication to Arduino found in proteus.

CHAPTER FOUR 4

RESULT AND SIMULATION

4.1. Circuit diagram

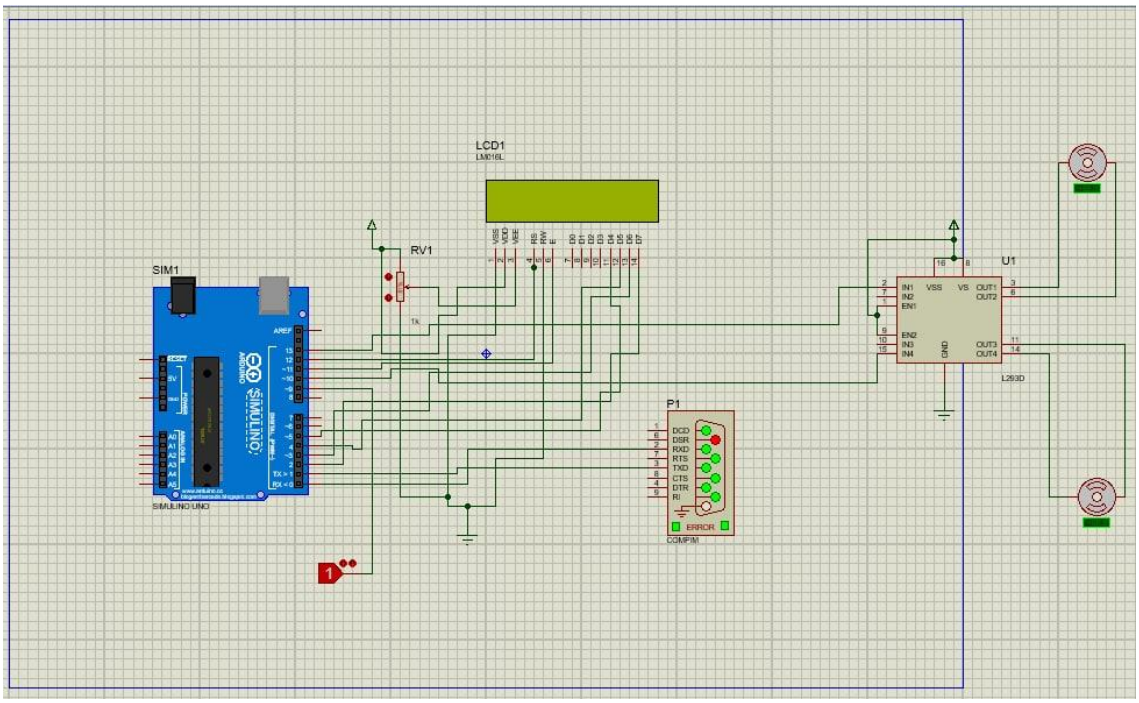


Figure 4. 1: Circuit diagram using Proteus.

Arduino Uno: the control structure that plays a role of accepting the coming data from Pycharm note book to LCD display.

Comport: Allow serial communication between Arduino inside the proteus and the Pycharm note book of the computer after whole image classification is completed,

LCD: helps to display the decision of computer by the instruction of Arduino uno via comport.

Dc Motors: helps to spray the herbicides.

4.2. Evaluation techniques

Recall: out of all the positive classes, how much we predict correctly. It is calculated as shown in equation

$$\text{Recall}=\frac{TP}{TP+FN} \quad \ldots (7)$$

Precision: Out of all the positive classes we have predicted correctly, how many are actually positive. It is calculated as shown in equation

$$\text{Precision}=\frac{TP}{TP+FP} \quad \ldots (8)$$

Accuracy: Out of all the classes, how much we predicted correctly

$$\text{Accuracy}=\frac{TP}{TN+TP+FP+FN} \quad \ldots (9)$$

Computation time: the time taken to finish image processing.

4.3. Image classification and localization using YOLO

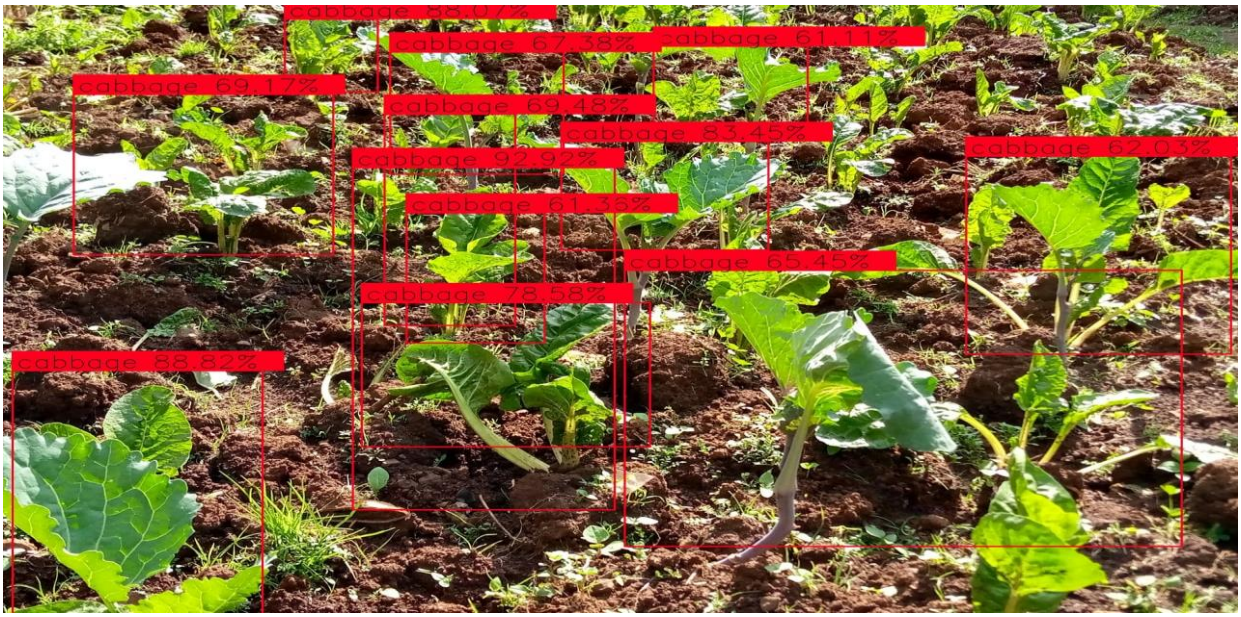
In this section depicts the performance of the YOLOv3 model on both still images and on video records. portrays the detection and labeling of the objects in a single image. Figure 9 shows the average lose during training.

First the image is fed to the yolov3 Architecture for feature extraction, classification and localization. After the image processing the model returns the class label, in addition to the class label it returns the bounding box prediction which includes x-coordinate, y-coordinate, width and height of each crops found in the image, but the bounding box prediction is not the actual position of the crop rather it is the pixel position in the image. Then, the pixel coordinate and the class label value of each crops in the images is sent to the Arduino via virtual communication port to be displayed on LCD.

(a)

(b)

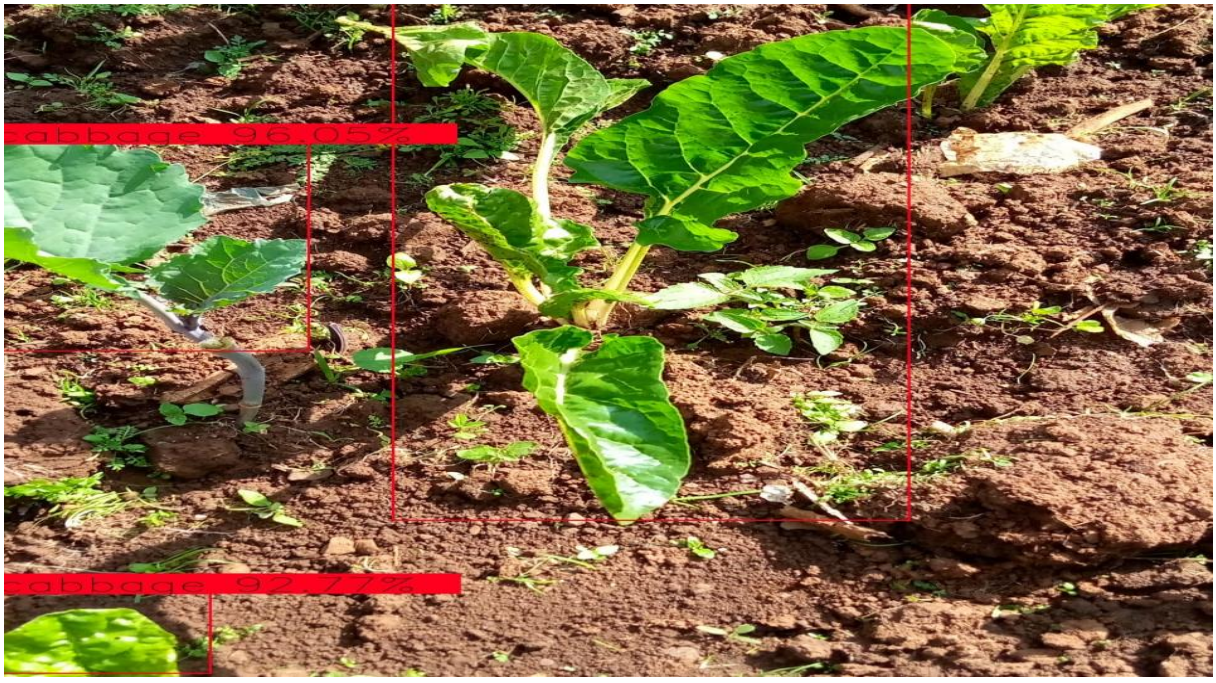
(c)



(d)

Figure 4. 2: Cabbage (vegetable) detection and localization algorithm using CPU Images in at (a) and (c) are input images and (b) and (d) the right one shows processed image bounding box prediction of the input image. The red line from the right image shows the plant identified is localized at that particular position.

(a)



(b)

(c)

(d)

Figure 4.3: GPU based YOLOv3 object detection the image (a)and(c) in the above figure are input images that prepared to be processed in the trained data whereas the (b) and (d) in the above figure are the processed images that are localized in a certain position with its corresponding confidence level within red color.

GPU based YOLOv3 image processing algorithm training yields highly accurate detection of crop positions from the entire image cross-section. It finishes its prediction within a few microseconds. Its drawback is it needs higher number of datasets and epochs for the training. While CPU based training is very easy for training with image processing speed of 1 minute which it is very slow and generate one images at a time and it finishes after one minute.

During training, YOLO uses differential weight for confidence predictions from boxes that contain object and boxes that do not contain objects. It penalizes errors in small and large objects differently by predicting the square root of the bounding box width and height.

4.4. Result and discussion for method one

.

4.4.1. Evaluation metrics

The YOLO-based threat object detector was evaluated using the PASCAL VOC detection metric at an Intersection Over Union IoU threshold of 0.5. After training it for 2000 epochs on Google colab. The IoU and class loss is depicted in the table for 1,100,500,1000,1500.

Epochs	IOU	Class loss
1	180.8411	230.7014
100	34.60583	57.41690
500	3.743498	8.631455
1000	0.702182	0.73793
1500	0.136682	0.373523
2000	0.0032143	0.012890

Table4.1 : GPU based YOLOv3 training performance

The table dictates that as the number of epochs for the training decrease IoU and Class loss.

4.5. Object classification using AlexNet

The simulation is done on Proteus simulation software and its implementation is explained as follow. First the image is fed to the AlexNet for feature extraction and classification. After the image processing the model returns the class label of the input image as 0 or 1. 0 represent if it is

crop in our case Cabbage and 1 represent if it is any kinds of herb. Then, the predicted class label is sent to the Arduino Serially with the help of Virtual communication port. Finally, the Arduino display the class label on the 16x2 LCD.

4.6 Result and discussions

The computation result of image classification is about 1 minute for CPU and microseconds using GPU. Using GPU increase the production cost of the vehicle system

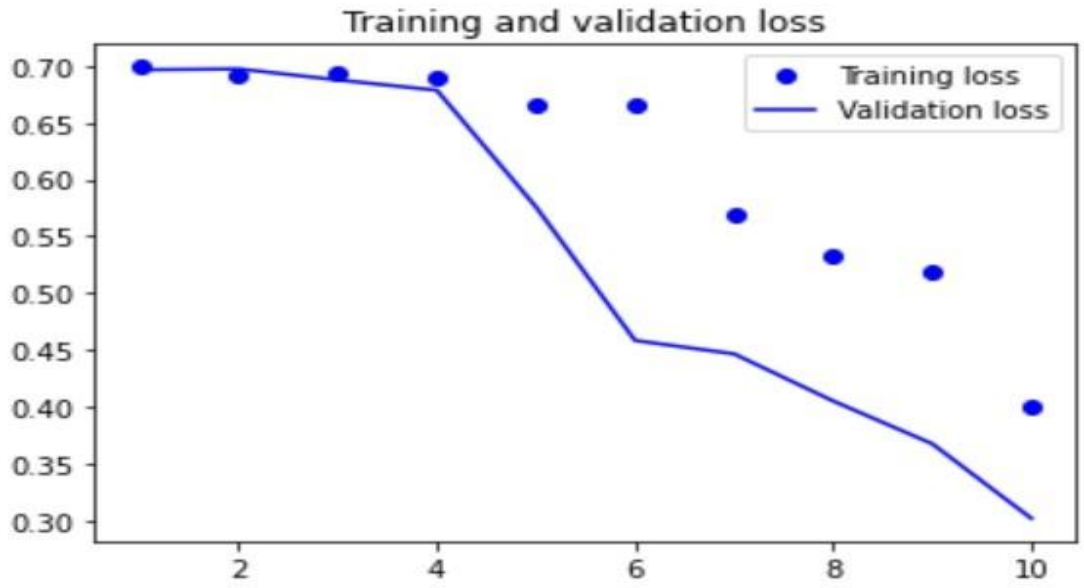


Figure 4. 5: Training and validation loss of the proposed CNN for 10 epochs and 20 steps per epochs

From the graph in the above figure 4.5, we can infer that the validation loss and training loss at the beginning of the training was 0.7 up to epoch 4 and after 4 epoch of the training it starts to decrease, after 10 epoch the validation loss settles to the value between 0.3 and 0.35 and the training loss settles to the value between 0.35 and 0 .4.

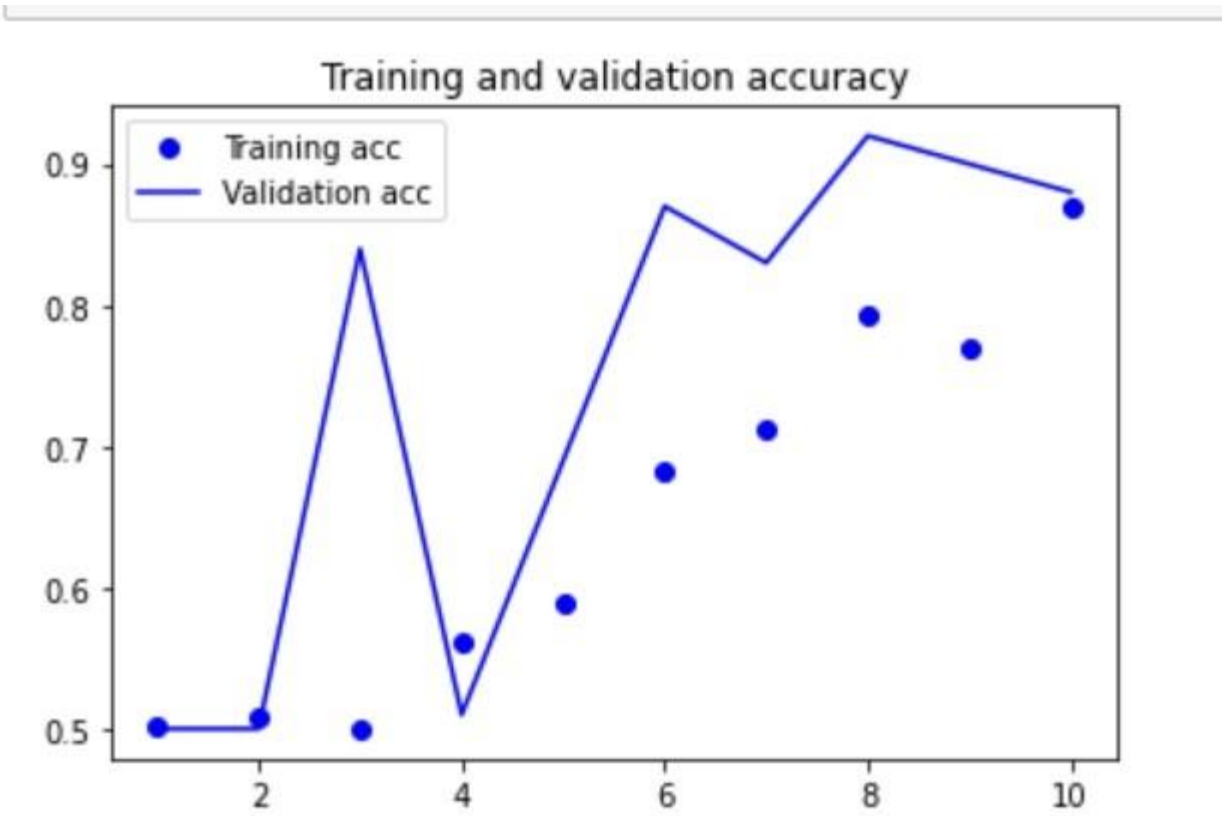


Figure 4. 6: Training and validation accuracy of the proposed CNN for 10 epochs and 20 steps per epochs

From the above graph, we can infer that the validation and training accuracy at the beginning of the training was the model has trained for 10 epochs and 20 steps per epoch using 720 training images, 100 validation images and 100 test images. The training accuracy is around 90% as show the graph of validation and training accuracy during the training process. we can infer that the validation and training accuracy at the beginning of the training was 0.5 up to epoch 2 and after 10 epochs both the training and the validation accuracy settle between the value of .8 and .9 which is desirable value. The confusion matrix result can be interpreted as follow:

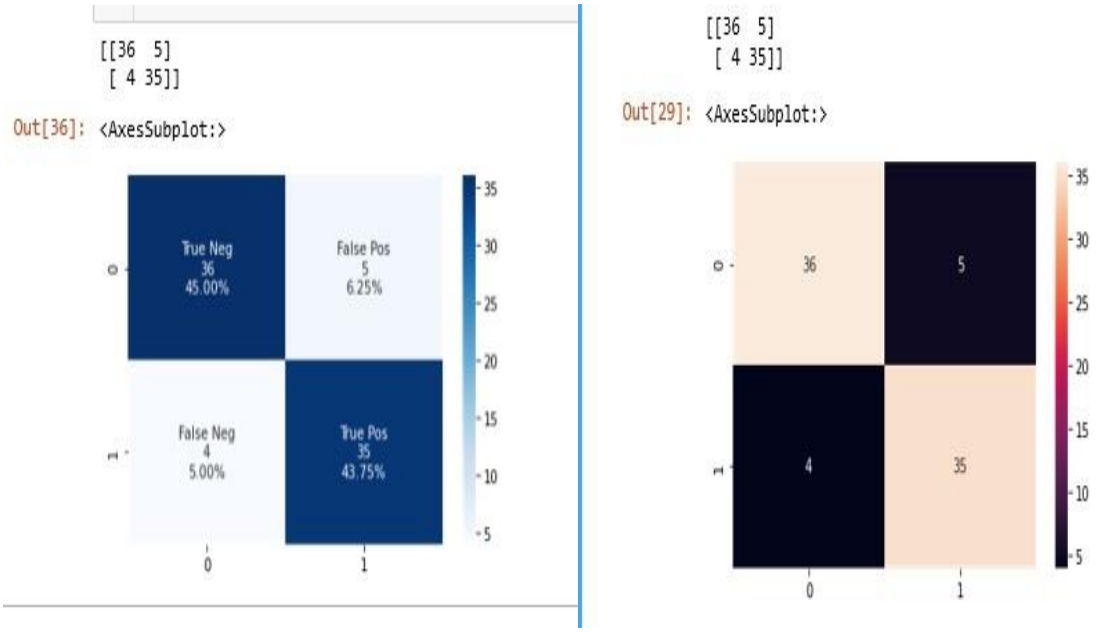


Figure 4. 7: a) confusion matrix for AlexNet in percentage b) confusion matrix of AlexNet in actual value

A total of 80 images 40 for crops and 40 for herbs is fed into the AlexNet for evaluation of the model and from the confusion matrix the following it can be interpreted as follow:

1.There is 36 TN which means out of 40 classes 0 the classifier predicts 36 of them correctly as class 0

2. The false negative result of the prediction is 5, which means out of 40 class 0 samples, it predicts 5 of them as class 1

3.False positive result of the prediction is 4, which means out of 40 class 1 samples it predicts 4 of them as 0

4. True positive results of the prediction is 36, which means out of 40 class 1 samples it predicts 36 of them as 1

From the confusion matrix explained above result we can calculate the precision, the callbacks, and accuracy of our model.

Class	Precision	Recall		Accuracy	Average Computation time per image
0	0.875	0.897		0.875	1 second
1	0.9	0.878			1 second

Table4.2: Performance of image classification by method I

Evaluation of the test classification efficiency of the proposed model using the confusion matrix is summarized in table shows the confusion matrix for Alex Net. The table shows the recall, precision, and accuracy derived from the confusion matrix obtained by the AlexNet Architectures.

4.6. Components of mechanical design of vehicle system

Camera: The vehicle has a camera system attached in the front side of the vehicle system. It has connected to the Arduino system placed inside the box below the solar panel.

Main frame: It holds the whole components and subcomponent of the vehicle system. **Ball screw**: It is float structure on the thread system. The main function of the ball screw is to support the pipe comes from the cylinder with fertilizer or herbicide. **Cylinder tank**: It is a container structure. It carries fertilizer of herbicide prepared for sowing or spraying.

Thread: It has rotationally locked with motor structure that allows the ball screw movement to intended position.

Pipe: It plays a role of transporting the herbicide or fertilizer from the cylinder to the intended cross-sectional area. It holds the nozzle that will have a valve system opened and closed based on the instruction of Arduino system.

Solar panel: It is the power house of the vehicle system**.**

CHAPTER FIVE
CONCLUSION AND FUTURE WORK

5.1. Conclusion

The designed Crop care vehicle system used in sowing and spraying fertilizer comprised a computer vision algorithm that uses deep learning to detect specific target crops, and a hardware with fast response to nozzles of herbicide sprayer and fertilizer sower. It is found crop detection and localization using YOlOVv3 can process one image per minute by CPU. GPU based YOLOv3 is takes very little computation time around hundreds of microseconds with accuracy of 97% after 2000 epochs of 75 image datasets. However, it has relatively costly and takes longer days for training. Crop classification using AlexNet without any GPU yields 90% accurate crop classification and it is very efficient for the vehicle system which cover small cross-sectional area equivalent to the size of the crop or slightly bigger than it. Crop classification takes around 100 milliseconds, if it finds efficient hardware it can finish 300x cross-sectional area of that particular crop. On this project the training datasets are cabbage and it saves 35% of herbicide chemicals to unintended that crops are found and around 65% of fertilizer sowed to unintended area that herbs are likely found.

There are lands that will not have both herbs and fertilizers but the machine detects it as noncrop so that it will execute spraying herbicide at that particular location makes the machine's resource utilization less optimal. Generally, it can be concluded the designed Crop care vehicle system used in sowing and spraying fertilizer can able to reduce the time for the production of crops, increase efficiency of resource utilization and highly accurate crop detection

5.2. Future works

I. The vehicle system should work with the maximized accuracy than now by training a Training CNN using tens of thousands of datasets which consists of different conditions of appearances such as shadow, night-vision, overlapping crops, young crops, grown up crops, damaged crops, covered crops (it might be covered with soil), partial images of crops.

II. The image analysis for weed detection can be further improved by dividing the image into a greater number of regions and has as many nozzles to spray the chemicals.

III The proposed system is limited to make the crop detection only for cabbage detection in different farming field. So, for the future it is expected to trained and detect a greater number of crops, as well as more than one crops at the same farming field.

IV The vehicle's mechanical system should be done by experts and the cost analysis will be done.

V The use compatible cost-effective microchips such as AVR and PIC for taking cameras and processing the python algorithms.

VI It can be turned into a very robust closed loop system by incorporating a memory module.

The image processing algorithm can be developed further so that the detection becomes more generic.

REFERENCES

[1] Seneshaw Tamru, Bart Minten, FAntu Bachewe, Dawit Alemu, "The rapid expansion of herbicide use in smallholder agriculture in Ethiopia: Patterns, drivers and implications", December 16th, 2016.

[2] January 22, 2018 . [Online]. Available: https://www.healthline.com.

[3] A. Stroud, "Weed managment in Ethiopia", food and agriculture organization of the United Nations., Rome, 1989.

[4] Sa I, Chen Z, Popvic M, Khanna R, Liebisch F, Nieto J, "Dense semantic weed classification using multispectral images for smart farming", *In IEEE Robot Autom Lett,* 2017.

[5] 28 october 2020. [Online]. Available:

https://www.analyticsvidhya.com/blog/2020/10/what-is-the-convolutional-neural-network architecture/ .

[6] 3, july 2019. [Online]. Available: https://towardsdatascience.com/alexnet-the-architecture that-challenged-cnns-e406d5297951 .

[7] 23 Apr 2018. [Online]. Available: https://towardsdatascience.com/yolo-v3-object detection-53fb7d3bfe6b .

[8] B. Hardjono, A. E. Widjaja, R. Kondorura, and A. M. Halim, “Deep Learning Methods", *IEEE 9th Annu. Inf. Technol. Electron.,* 2018 .

[9] “Vehicles detection of traffic flow video using deep learning”, DDCLS, Proc. 2019 IEEE 8th Data Driven Control Learn. Syst. Conf.

[10] HongJun Wang, Qisong Mou, Youjun Yue, Hui Zhao, "Research on Detection Technology of Various Fruit Diesease Spots Based on Mask R-CNN," in *proceedings of 2020 IEEE International Conference on Mechatronics and Automation* , Beijing, China, 2020.

[11] R. Girshick, "Fast R-CNN," *arXiv:1504.08083,* vol. v1, 30 Apr 2015.

[12] Ren S, He K, Girshick R, Sun J, " Faster r-cnn: Towards real-time object with region proposal networks," *In: Advances in Neural Information Processing Systems,* pp. 91-99, 2015.

[13] Richardson Santiago Teles de Menezes, Rafael Marrocos Magalhaes and Helton Maia,"Object Recognition Using Convolutional Neural Networks Object Recognition Using Convolutional Neural Networks," 2019. 4647

[14] Roh, Myung-Cheol, and Ju-young Lee, "Refining faster-RCNN for accurate object detection", *in Proc. of Fifteenth IAPR International Conference on Machine Vision Applications (MVA), pp. ,* pp. 514-517, 2017.

[15] Han, Guangxing, Xuan Zhang, and Chongrong Li, "Revisiting Faster R-CNN: A Deeper Look at Region Proposal Network", in *in Proc. of Conference on Neural Information Processing*, 2017.

[16] Yu Liu Antai, "An Improved Faster R-CNN for Object Detection," *11th International Symposium on Computational Intelligence and Design*, 2018.

[17] Jianghong Tang, Yingchi Mao, Jing Wang, Longbao Wang, "Multi-task Enhanced Dam Crack Image Detection Based on Faster R-CNN," in *IEEE 4th International Conference on Image, Vision and Computing*, 2019.

[18] Deokkyu Jung, Jeong-Woo Son, Sun-Joong Kim, "Shot Category Detection based on Object Detection Using Convolutional Neural Networks," in *International Conference on Advanced Communications Technology (ICACT)*, 2018 February 11 ~ 14.

[19] S. Das, "medium data analysts," 16 Nov 2017, Available: https://medium.com/analytics vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5 .

[20] Klemen Grm, Vitomir Struc , Anais Artiges, Matthieu Caron, Hazim Kemal Ekenel, "IET BIOMETRICS Strengths and Weaknesses of Deep Learning Models for Face Recognition Against Image Degradations," 4 Oct 2017.

[21] P. Zhang,T.D.Bui, C.Y.Suen, "Recognition of Similar Objects Using 2-D Wavelet-Fractal Feature Extraction," Montreal, canada, 2002.

[22] Jie Yang, Fan Liu, "Modulation Recognition using Wavelet Transform based on AlexNet," in *IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, china, 2019.

[23] Rosa Andrie Asmara, Dodit Supriyanto, Bimo Syahputro, Anik Nur Handayani, "Prediction of Traffic Density Using YOLO Object Detection and Implemented in Raspberry Pi 3b + and Intel NCS 2," pp. pp 391-396.

[24] Md. Bahar Ullah, "CPU Based YOLO: A Real Time Object Detection Algorithm," *2020 IEEE Region 10 Symposium (TENSYMP),* 5-7 June 2020.

[25] Arka Prava Jana, Abhiraj Biswas, Mohana Bengaluru , "YOLO based Detection and Classification of Objects in video records," MAY 18th & 19th 2018 .

[26] Bakhshipour A, Jafari A, Nassiri SM, Zare D. , "Weed segmentation using texture features extracted from wavelet sub-images.," in *Biosyst Eng.* , 2017.

Appendix

Appendix A

YOLO Image Detection and Localization Code

Predict.py

```
#! /usr/bin/env python

import os

import argparse

import json

import cv2

from utils.utils import get_yolo_boxes, makedirs

from utils.bbox import draw_boxes

from tensorflow.keras.models import load_model

from tqdm import tqdm

import numpy as np

import time


def _main_(args):

    config_path  = args.conf

    input_path   = args.input

    output_path  = args.output

    with open(config_path) as config_buffer:

        config = json.load(config_buffer)

    makedirs(output_path)

    #   Set some parameter
```

```
    net_h, net_w = 320, 320 # a multiple of 32, the smaller the faster
    obj_thresh, nms_thresh = 0.5, 0.45
    #   Load the model
    os.environ['CUDA_VISIBLE_DEVICES'] = config['train']['gpus']
    infer_model = load_model(config['train']['saved_weights_name'])
    #   Predict bounding boxes
    if 'webcam' in input_path: # do detection on the first webcam
        video_reader = cv2.VideoCapture(0)
        # the main loop
        batch_size  = 1
        images      = []
        detected = []
        while True:
            ret_val, image = video_reader.read()
            if ret_val == True: images += [image]
            if (len(images)==batch_size) or (ret_val==False and len(images)>0):
                batch_boxes = get_yolo_boxes(infer_model, images, net_h, net_w,
config['model']['anchors'], obj_thresh, nms_thresh)
                for i in range(len(images)):
                    draw_boxes(images[i], batch_boxes[i], config['model']['labels'], obj_thresh)
                    cv2.imshow('video with bboxes', images[i])
                # # The Code below is to get detected class.
                boxes = batch_boxes[0]
```

```
            detected = []
            for i in range(len(config['model']['labels'])):
                for box in boxes:
                    if box.classes[i] > obj_thresh:
                        detected.append(config['model']['labels'][i])
            images = []
        if cv2.waitKey(1) == 27:
            break # esc to quit
        elif len(detected) > 1: detected = []     # In case multiple detection, ignore.

    cv2.destroyAllWindows()
    print('{} is detected object! '.format(detected[0]))
    return detected[0]  # If cabbage is detected and break the while loop by some method/may be after detection,
                        # the returned value here is 'cabbage'
  elif input_path[-4:] == '.mp4': # do detection on a video
    video_out = output_path + input_path.split('/')[-1]
    video_reader = cv2.VideoCapture(input_path)

    nb_frames = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
    frame_h = int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))
    frame_w = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))

    video_writer = cv2.VideoWriter(video_out,
```

```
                    cv2.VideoWriter_fourcc(*'MPEG'),
                    50.0,
                    (frame_w, frame_h))
    # the main loop
    batch_size  = 1
    images      = []
    start_point = 0
    show_window = False
    for i in tqdm(range(nb_frames)):
        _, image = video_reader.read()
        if (float(i+1)/nb_frames) > start_point/100.:
            images += [image]
            if (i%batch_size == 0) or (i == (nb_frames-1) and len(images) > 0):
                # predict the bounding boxes
                batch_boxes = get_yolo_boxes(infer_model, images, net_h, net_w,
config['model']['anchors'], obj_thresh, nms_thresh)
                for i in range(len(images)):
                    # draw bounding boxes on the image using labels
                    draw_boxes(images[i], batch_boxes[i], config['model']['labels'], obj_thresh)

                    # show the video with detection bounding boxes
                    if show_window: cv2.imshow('video with bboxes', images[i])
                    # write result to the output video
```

```
                video_writer.write(images[i])
            images = []
        if show_window and cv2.waitKey(1) == 27: break  # esc to quit
    if show_window: cv2.destroyAllWindows()
    video_reader.release()
    video_writer.release()
  else: # do detection on an image or a set of images
    image_paths = []
    if os.path.isdir(input_path):
        for inp_file in os.listdir(input_path):
            image_paths += [input_path + inp_file]
    else:
        image_paths += [input_path]
    image_paths = [inp_file for inp_file in image_paths if (inp_file[-4:] in ['.jpg', '.png', 'JPEG'])]
    # the main loop
    for image_path in image_paths:
        image = cv2.imread(image_path)
        print(image_path)
        # predict the bounding boxes
        boxes = get_yolo_boxes(infer_model, [image], net_h, net_w, config['model']['anchors'], obj_thresh, nms_thresh)[0]
        # draw bounding boxes on the image using labels
        draw_boxes(image, boxes, config['model']['labels'], obj_thresh)
```

```
            # write the image with bounding boxes to file
            cv2.imwrite(output_path + image_path.split('/')[-1], np.uint8(image))
if __name__ == '__main__':
    argparser = argparse.ArgumentParser(description='Predict with a trained yolo model')
    argparser.add_argument('-c', '--conf', default= 'config.json', help='path to configuration file')
    argparser.add_argument('-i', '--input', default='webcam', help='path to an image, a directory of images, a video, or webcam')
    argparser.add_argument('-o', '--output', default='output/', help='path to output directory')

    args = argparser.parse_args()
    _main_(args)
```

Appendix B: Training Code

Train.py

```
#! /usr/bin/env python

import os
import argparse
import json
import cv2
from utils.utils import get_yolo_boxes, makedirs
from utils.bbox import draw_boxes
```

```
from tensorflow.keras.models import load_model
from tqdm import tqdm
import numpy as np
import time
def _main_(args):
    config_path  = args.conf
    input_path   = args.input
    output_path  = args.output

    with open(config_path) as config_buffer:
        config = json.load(config_buffer)

    makedirs(output_path)
    #   Set some parameter
    net_h, net_w = 320, 320 # a multiple of 32, the smaller the faster
    obj_thresh, nms_thresh = 0.5, 0.45
    #   Load the model

    os.environ['CUDA_VISIBLE_DEVICES'] = config['train']['gpus']
    infer_model = load_model(config['train']['saved_weights_name'])
    #   Predict bounding boxes
    if 'webcam' in input_path: # do detection on the first webcam
        video_reader = cv2.VideoCapture(0)
```

```
    # the main loop
    batch_size  = 1
    images      = []
    detected = []
    while True:
        ret_val, image = video_reader.read()
        if ret_val == True: images += [image]
        if (len(images)==batch_size) or (ret_val==False and len(images)>0):
            batch_boxes = get_yolo_boxes(infer_model, images, net_h, net_w,
config['model']['anchors'], obj_thresh, nms_thresh)


            for i in range(len(images)):
                draw_boxes(images[i], batch_boxes[i], config['model']['labels'], obj_thresh)
                cv2.imshow('video with bboxes', images[i])
            # # The Code below is to get detected class.
            boxes = batch_boxes[0]
            detected = []
            for i in range(len(config['model']['labels'])):
                for box in boxes:
                    if box.classes[i] > obj_thresh:
                        detected.append(config['model']['labels'][i])
            images = []
        if cv2.waitKey(1) == 27:
```

```
            break  # esc to quit
        elif len(detected) > 1: detected = []      # In case multiple detection, ignore.

    cv2.destroyAllWindows()
    print('{} is detected object! '.format(detected[0]))

    return detected[0]  # If cabbage is detected and break the while loop by some method/may be after detection,
                        # the returned value here is 'cabbage'

elif input_path[-4:] == '.mp4': # do detection on a video
    video_out = output_path + input_path.split('/')[-1]
    video_reader = cv2.VideoCapture(input_path)

    nb_frames = int(video_reader.get(cv2.CAP_PROP_FRAME_COUNT))
    frame_h = int(video_reader.get(cv2.CAP_PROP_FRAME_HEIGHT))
    frame_w = int(video_reader.get(cv2.CAP_PROP_FRAME_WIDTH))

    video_writer = cv2.VideoWriter(video_out,
                        cv2.VideoWriter_fourcc(*'MPEG'),
                        50.0,
                        (frame_w, frame_h))
    # the main loop
    batch_size  = 1
```

```
    images      = []
    start_point = 0
    show_window = False
    for i in tqdm(range(nb_frames)):
        _, image = video_reader.read()

        if (float(i+1)/nb_frames) > start_point/100.:
            images += [image]

            if (i%batch_size == 0) or (i == (nb_frames-1) and len(images) > 0):
                # predict the bounding boxes
                batch_boxes = get_yolo_boxes(infer_model, images, net_h, net_w,
config['model']['anchors'], obj_thresh, nms_thresh)

                for i in range(len(images)):
                    # draw bounding boxes on the image using labels
                    draw_boxes(images[i], batch_boxes[i], config['model']['labels'], obj_thresh)

                    # show the video with detection bounding boxes
                    if show_window: cv2.imshow('video with bboxes', images[i])

                    # write result to the output video
                    video_writer.write(images[i])
```

```
                images = []54

            if show_window and cv2.waitKey(1) == 27: break  # esc to quit

    if show_window: cv2.destroyAllWindows()
    video_reader.release()
    video_writer.release()
  else: # do detection on an image or a set of images
    image_paths = []

    if os.path.isdir(input_path):
      for inp_file in os.listdir(input_path):
        image_paths += [input_path + inp_file]
    else:
      image_paths += [input_path]

    image_paths = [inp_file for inp_file in image_paths if (inp_file[-4:] in ['.jpg', '.png',
'JPEG'])]

    # the main loop
    for image_path in image_paths:
      image = cv2.imread(image_path)
      print(image_path)
```

```
        # predict the bounding boxes
        boxes = get_yolo_boxes(infer_model, [image], net_h, net_w, config['model']['anchors'], obj_thresh, nms_thresh)[0]

        # draw bounding boxes on the image using labels
        draw_boxes(image, boxes, config['model']['labels'], obj_thresh)

        # write the image with bounding boxes to file
        cv2.imwrite(output_path + image_path.split('/')[-1], np.uint8(image))
if __name__ == '__main__':
    argparser = argparse.ArgumentParser(description='Predict with a trained yolo model')
    argparser.add_argument('-c', '--conf', default= 'config.json', help='path to configuration file')
    argparser.add_argument('-i', '--input', default='webcam', help='path to an image, a directory of images, a video, or webcam')
    argparser.add_argument('-o', '--output', default='output/', help='path to output directory')

    args = argparser.parse_args()
    _main_(args)
```

Appendix C

Arduino Pumper Code

```
int data;
int pump=13;
void setup() {
  Serial.begin(9600);                    //initialize serial COM at 9600 baudrate
  pinMode(pump, OUTPUT);                 //declare the pump pin (13) as output
  digitalWrite (pump, LOW);              //Turn OFF the Pump in the beginning
  Serial.println("Hello!,How are you Python ?");
}
void loop() {
while (Serial.available())    //whatever the data that is coming in serially and assigning the v
alue to the variable “data”
{
data = Serial.read();
}
if (data == '1'){
digitalWrite (pump, HIGH);
}              //Turn On the pump
else if (data == '0'){
digitalWrite (pump, LOW);
}              //Turn OFF the pump
}
```