Final Report:

Developing a Deep Learning Model to Automatically Detect Microscale Objects in Images and Videos

Senior Design Group: sddec23-10

Ethan Baranowski, Chris Cannon, Matthew Kim, Katherine Moretina

Client/Faculty Advisor: Dr. Santosh Pandey

1 Introduction **3**
1.1 Acknowledgement 3
1.2 Problem and Project Statement 3
1.3 Operational Environment 4
1.4 Requirements 4
1.5 Intended Users and Uses 5
WHO has the problem? 5
WHAT is the problem? 5
WHERE is the problem occurring? 5
WHEN is the problem occurring? 5
WHY is it important? 6
HOW will it be solved? 6
Use Cases: 6
1.6 Assumptions and Limitations 6
1.7 Relevant Standards 7
1.8 Expected End Product and Deliverables 9
2 Previous Work **10**
2.1 Previous Work And Literature 10
2.1.1 Existing Processes 10
2.1.2 Existing Technology 11
2.1.3 Existing Algorithms 11
3 CprE 491 Design **13**
3.1 Original Hardware Conceptual Sketch 13
3.2 Detectron2 Implementation of Faster R-CNN 16
4 Evolution Since Cpr E 491 ..... **16**
4.1 Hardware Evolution 16
4.2 Software Tools Developed 18
5 Testing and Optimization **19**
5.1 Notable Hyperparameters 19
6 Results **21**
6.1 Hardware Results 21
6.2 Graphical User Interface Results 22
6.3 Integrating Detectron 2 to Raspberry Pi 4 Results 23
6.4 Detectron2 Optimization Results 23
7 Closing Statements **23**
7.1 Conclusion 23
7.2 References 25
6.3 Appendix 26

1 Introduction

1.1 Acknowledgement

The team would like to acknowledge the following people and groups for their contribution to our project:

- Dr. Santosh Pandey
  - For his guidance, leadership, and efforts facilitating our understanding of the problem and possible solutions.
- Yunsoo Park (Ph.D. Student)
  - For his help in understanding existing object detection algorithms, identifying cysts in images, and sharing his general knowledge of the problem with us.
- Dr. Greg Tylka (Morrill Professor, Nematologist, Director of the Iowa Soybean Research Center)
  - For his detailed explanation of existing processes and methods and his tour of the lab where that happens on Iowa State's campus.
- Facebook's Research team
  - For their implementation of Mask R-CNN in their open source Detectron2 project, which will likely serve as a strong foundation for our project.

1.2 Problem and Project Statement

Every year 15-30% of soybeans are infected with parasitic cyst nematodes *[1]*, limiting their yield and forcing farmers to use pesticides. Farmers are currently faced with the issue of expensive and slow lab analysis that provide insight into how many cysts are on soybean plants. These lab techniques cost the farmers valuable time as they must wait for the results to apply the fertilizer necessary for the continued health of their soybean crops.

Our goal is to make a product that can detect and count microscale objects, which is cysts in our case. In computer vision, there are different algorithms that can detect objects. In similar fields, for example, there are some algorithms that can detect cars or people in images *[2]*. Therefore we want to implement an object detection algorithm designed for small objects to determine how many cysts are on the roots of soybean plants. We will also create a device to integrate image capturing with the machine learning algorithm, therefore, increasing productivity on the farm. Detecting cysts is the first step in eliminating them. Geneticists will be able to run tests to find if certain species of soybeans are more resistant to cysts, and farmers will know the amount of pesticides to use. This process will increase soybean productivity and decrease the overuse of pesticides.

1.3 Operational Environment

In our case, to maximize the advantages of not damaging the plant's root, we are considering a portable hardware solution intended for outdoor use, specifically on farm fields by the researcher, geneticists and farmers. Since our products are allowed to be used outside of the labs, the product must be designed to function without access to Wi-Fi or electronic plugs for charging. It should be able to handle various weather conditions, such as sunny, cloudy, or windy days. Additionally, temperature fluctuations should be taken into account, though they will not be extreme. The target environment involves soybean cultivation, which typically occurs between 68°F and 86°F.

1.4 Requirements

Requirement Type	Details
Functional Requirements	● Software(Algorithm) ○ Functional without internet access ■ Independent of software upgrades (no server) ○ Better than 50% accuracy for detecting cysts in images **(Constraint)** ○ Less than 5 second of processing time per image **(Constraint)** ○ Less that 1GB space taken by the application **(Constraint)** ○ Image processing independent of MATLAB ● Hardware (Image Capture) ○ High-enough image resolution to use for object detection ○ Portable (Wireless, can be used on a field) ○ Does not damage plants
Resource Requirements	● Processors/Processing time to train the algorithm ● Materials to build a prototype device ○ Cameras ○ Motors (depending on prototype design) ○ Misc building materials (aluminum, plexiglass, screws, wires, etc) ● Plants of certain age (3 week to 5 week maturity)
Qualitative Aesthetics Requirements	● Keep the device as small as possible ● Easy to transport

Economic/Market Requirements	• Create a final device that is affordable for farmers ○ Upper limit around $500 **(Constraint)**
Environmental Requirements	• Must not damage plants in any way • Provide an accurate number of cysts to avoid an overuse of pesticide consumption
UI/UX Requirements	• Farmer should not interact with product during image capture process • No formal training required to use the device

Table 1.4-1 Project Requirements

1.5 Intended Users and Uses

WHO has the problem?

Primary users: Industries and crop researchers.

Secondary users: Farmers.

WHAT is the problem?

Expensive analysis required to gain measurable insight into how many parasitic cysts are on soybean plants. Currently in the U.S, cysts nematodes are affecting about 15~30% of yields of soybean every year. That causes about $1 billion losses annually *[1]*.

Also, there aren't any methods that analyze the number of cysts directly from the root. Other methods require using technical equipment to crush the Cysts and analyze using the microscopic tools.

WHERE is the problem occurring?

The cysts attach to the roots of soybeans underground. In the United States, soybean production is most common in Iowa, Illinois, Nebraska, Minnesota, and Indiana. Also in the research lab, expensive tools are used with technical steps in order to count cysts.

WHEN is the problem occurring?

Before harvesting, during the growth of the plants.

WHY is it important?

Crop researchers can make more educated decisions about while developing cyst-resistant plants. Farmers can make more educated decisions about the proper care of their crops.

HOW will it be solved?

Machine Learning algorithms have made several developments that are tailored to identifying small objects in images. These improvements give a strong indication that a prototype device can be trained to recognize the cysts and provide accurate measurements to farmers.

Use Cases:

1) Uses for farmers
2) Uses for genetecists/plant pathologists
3) Uses for continuing developers/project maintainers

Figure 1.5-1 below shows a use case diagram for our final product:

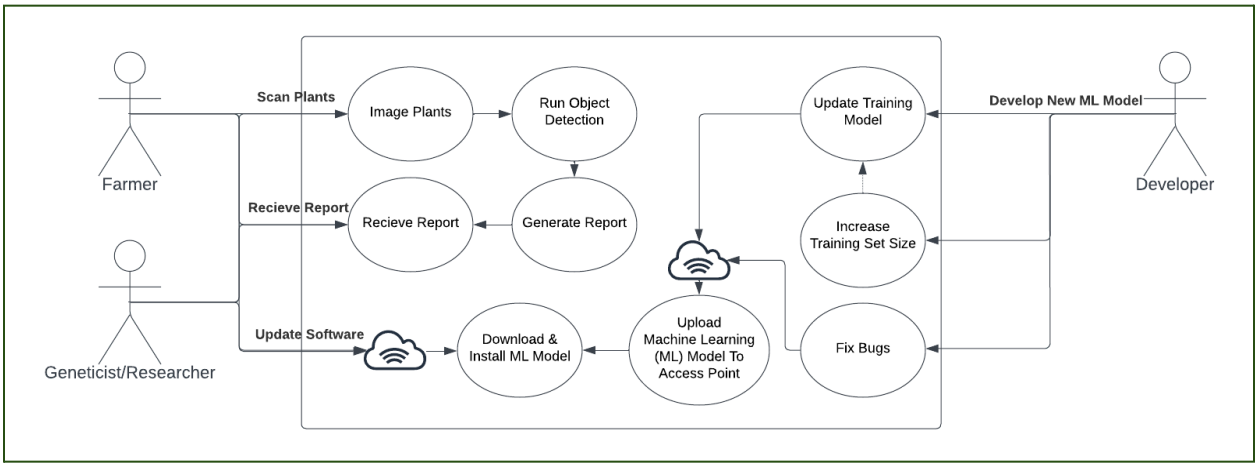


Figure 1.5-1: User Case diagrams, interaction between Researcher, Farmer and Developer

1.6 Assumptions and Limitations

Assumption	Justification
Only 2 pictures per plant will provide enough data for the counting algorithm.	Scanning both sides of the plant should capture every cyst in at least one image.

User will not have to interact with the product during its image capturing and counting algorithm phases.	Product will be mostly autonomous, with some user interaction for setting it up and running it.
Blue is the best background color.	Blue is the most direct contrast to the colorization of the cysts.
Python counting algorithm will be runnable on a Raspberry Pi.	
Raspberry Pi can interface with and control the LCD screen, the camera, and the motor.	Adapters and circuit board (may not be required) connections are available for each product.
Product can be used in the field.	Product is easily transportable and internet free.
Product will be used for soybean parasitic cyst counting only.	Image capture process is trained for parasites unique to soybean plants.

Table 1.5-1 Assumptions and Justifications

Limitation	Justification
1 plant processed at a time.	Counting process is limited to one plant with associated results. Multiple plants would derefence the results.
Cost to produce shall not exceed $500.	
Dimensions will be 12”x15”	Size of the base
Counting algorithm will exceed 50% accuracy but not exceed 80% accuracy.	Faster R-CNN algorithm has repeatedly shown high performance for small object detection, but the research has not indicated over 80% accuracy.
Battery life of 2 hours.	

Table 1.5-2 Limitations and Justifications

1.7 Relevant Standards

Our task is to develop a deep learning model that can find and count the number of specific microscale objects in the image. To address the problem, we have researched various object detection algorithms, including YOLO, Faster R-CNN, and Single Shot Detection. We have compared these algorithms to determine the best fit for our project. Additionally, we have

explored labeling tools and started working on the labeling process. We have also identified existing algorithm implmentations that we may use as part of our project. Based on our algorithm implementation, we will develop hardware tools that are portable, which researchers can utilize outside of their laboratories.

Our project adheres to or is relevant to the following standards:

IEEE 268-1992

American National Standard for Metric Practice

Guidance for the application of the modernized metric system in the United States is given. Known as the International System of Units (SI), the system is intended as a basis for worldwide standardization of measurement units. Information is included on SI, a limited list of non-SI units recognized for use with SI units, and a list of conversion factors from non-SI to SI units, together with general guidance on proper style and usage. [4]

IEEE/ISO/IEC 32675-2021

- ISO/IEC/IEEE International Standard--Information technology--DevOps--Building reliable and secure systems including application build, package and deployment

Technical principles and processes to build, package, and deploy systems and applications in a reliable and secure way are specified. Establishing effective compliance and information technology (IT) controls is the focus. DevOps principles presented include mission first, customer focus, left-shift, continuous everything, and systems thinking. How stakeholders, including developers and operations staff, can collaborate and communicate effectively is described. The process outcomes and activities herein are aligned with the process model specified in ISO/IEC/IEEE 12207:2017 and ISO/IEC/IEEE 15288:2015. [5]

IEEE/ISO/IEC P24748-6

ISO/IEC/IEEE Draft Standard - Systems and Software Engineering -- Life Cycle Management

This standard:

- Provides requirements and guidance for use of the integration process and its relationships to other system and software life cycle processes as described in ISO/IEC/IEEE 15288:2015 and ISO/IEC/IEEE 12207:2017,
- Specifies information items to be produced as a result of using the integration process, including the content and format of the information items.

This standard provides a detailed presentation of system and software integration, considering:

- The related concepts of integration, such as interface, verification, and validation;

- The possible composition of a system comprised of any mix of products and services that can include hardware, software, humans, data, processes (e.g. review process), procedures (e.g. operator instructions), facilities and naturally occurring entities (e.g. water, organisms, minerals),
- The life cycle stages of a system at which integration may occur, and
- The context of the domain in which the system functions. [6]

IEEE/ISO/IEC 14764-2021

ISO/IEC/IEEE International Standard - Software engineering - Software life cycle processes - Maintenance

This International Standard establishes a common framework for software life cycle processes, with well defined terminology, that can be referenced by the software industry. It contains processes, activities, and tasks that are to be applied during the acquisition of a software system, product or service and during the supply, development, operation, maintenance and disposal of software products. [7]

1.8 Expected End Product and Deliverables

The primary end goal for the Cyst Detector project is to develop and implement machine learning algorithms that can accurately detect small-scale objects, specifically cysts. The software will be designed to identify cysts within plant roots without causing any damage or requiring grinding of the roots. This cutting-edge algorithm will lay the foundation for future hardware integration, enabling the creation of a scanner that utilizes the algorithm to count the number of cysts present. Delivery date: Fall of 2023.

As an extension of the project, we will develop a hardware solution that acts as a scanner to utilize the machine learning algorithm for cyst detection and counting. The hardware will be designed with simplicity, cost-effectiveness, and automation in mind, allowing users to focus on other tasks while the device processes the count. Furthermore, the hardware will be suitable for outdoor use, portable, and able to operate without an internet connection.

A user-friendly software interface will be developed to facilitate seamless interaction between the user and the cyst detection system. This interface will allow users to scan plant roots with the hardware device and obtain cyst count results generated by the machine learning algorithm. The software will be designed to work efficiently both with and without an internet connection, ensuring flexibility and reliability in various environments.

Comprehensive documentation will be provided for both the cyst detection machine learning algorithm and the hardware scanner which clients can update and maintain afterwards as well. Documentation will include guidelines for installation, setup, and operation, along with

troubleshooting tips and maintenance instructions. These resources will ensure that users can effectively deploy and manage the Cyst Detector system in their operations.

2 Previous Work

2.1 Previous Work And Literature

The issue of soybean nematode cysts has been studied since the 1950s, with current processes very similar to the ones developed back then. The two current processes are discussed below, along with relevant technology and machine learning algorithms.

2.1.1 Existing Processes

From talking to Dr. Greg Tylka in Plant Pathology and Dr. Santosh Pandey in Electrical Engineering, we know there are a couple of processes for counting these nematodes that already exist.

1. In the **corporate** research environment, soybean samples are visually scanned by experts to get an estimate of how many cysts are present. This method is imprecise and requires a subject matter expert, i.e. lots of investment in training or finding appropriate personnel.
    - The strength of this method is primarily the simple workflow. A single person takes a sample, examines it, reports their finding, and then disposes of the sample
    - However, this process requires highly skilled labor, which could potentially be cost-prohibitive, as well as limit the quantity processed.
2. In the **academic** research environment, soil samples are sieved to filter out the appropriate size particles to capture cysts. Then, the cyst-sized particles are ground down to release eggs and create a sample. The sample must then be dyed and a prepared on a microscope slide, and finally visually examined by either an algorithm or human researcher to count the number of eggs, discounting any dirt that remains in the sample.
    - The advantages of this process are consistency and accuracy. The methods employed in this environment are known and frequently used, and the accuracy is significantly higher since they are counting the individual eggs.
    - The primary drawbacks of this method are that it is both time-consuming and requires specialized equipment. This means that it isn't ideal for high-volume use cases.

Context: Eggs vs. Cysts

Eggs are different than cysts - each cyst contains 200-250 eggs. In process (2) above, cysts are ground down to release the eggs, which are then counted under a microscope.

This project will be detecting cysts.

2.1.2 Existing Technology

1. There was a company founded (Creative AI) to assist in process (2) above, which created the algorithm referenced in that process. A plant pathology professor, Dr. Tylka currently uses Creative AI to assist in his lab research. However, that company went out of business, and its founders are now working for other companies on unrelated projects.
2. NVidia is a big name in the AI market, and with a product line known as Jetson designed for embedded devices and usecases, it provides hardware and software designed for embedded systems using AI and deep learning. [8]
3. The Detectron2 project, published on GitHub as an open source project by Facebook's research division, contains an implementation of Mask R-CNN, an improved version of Faster R-CNN, discussed below. Included in their project are details on how to train a Mask R-CNN model with a custom dataset, and we are exploring using this implementation as the machine learning algorithm in our project.

2.1.3 Existing Algorithms

We did extensive background research on object detection algorithms in order to choose an appropriate algorithm for our use-case. In general, object detection algorithms have two functions: object detection, and object classification. Object detection is used to find the presence of distinct objects in an image, and a classification model is used to predict *what* an object is. Classification models have to be trained on certain object classes.[9]. In our case, we will be using an existing implementation of an object detection & classification algorithm to detect and count cysts in a image of a soybean root.

We are following the Faster R-CNN process for machine learning object detection. This process is slower in counting the cysts than the state-of-the-art for real-time object detection. However, with a slower process comes higher accuracy. We discovered that Faster R-CNN is a top performing algorithm for classifying objects, especially small objects[3]. Upon successful implementation, we expect to exceed our clients accuracy expectations by a considerable margin.

Below is a table comparing our chosen algorithm, Faster R-CNN, with other contemporary deep learning algorithms. Notably absent is the fact that Faster R-CNN is known to have a better accuracy with small objects in images.

Faster R-CNN [3]	You Only Look Once (YOLO) [10]	Single Shot Detector (SSD) [11]
5 FPS	20-150 FPS	20-50 FPS
Uses Region Proposal Network to generate regions containing objects for classification.	Uses Anchor Boxes to generate regions containing objects for classification.	Uses Anchor Box Pyramids to generate regions containing objects for classification.
Uses Convolutional Neural Network to classify objects.	Uses Convolutional Neural Network to classify objects.	Uses Convolutional Neural Network to classify objects.
Algorithm Training takes a considerable amount of time.	Algorithm Training takes a moderate amount of time.	Algorithm Training takes a moderate amount of time.
2 stage network- Additional stage is Region of Interest Pooling layer which filters out bad predictions improving accuracy and speed later.	1 stage network	1 stage network
73% mean Average Precision (mAP)	50-65% mean Average Precision (mAP)	75% mean Average Precision (mAP)
Capable of handling high resolution images.	Capable of handling high resolution images.	Capable of handling high resolution images.

Table 3.1-1 Object Detection Algorithm Comparison

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	~ 1000 × 600
Fast YOLO	52.7	155	1	98	448 × 448
YOLO (VGG16)	66.4	21	1	98	448 × 448
SSD300	74.3	46	1	8732	300 × 300
SSD512	76.8	19	1	24564	512 × 512
SSD300	74.3	59	8	8732	300 × 300
SSD512	76.8	22	8	24564	512 × 512

Figure 3.1-1 Research illustrating the results of running several machine learning algorithms. Note that the algorithms were not run on the same data set for each case and are not indicative of small object performance.

Figure 3.1-2 Model architecture of the Faster R-CNN model.

3 CprE 491 Design

3.1 Original Hardware Conceptual Sketch

Our intended design is shown in the sketch below (Figure 3.1-1). It is a 3-D representation of the prototype we intend to develop. The description is broken down as:

Front: handlebar for user to carry the product.

Side: LCD screen for displaying parasitic cyst count. Rotating attachment point for arms. The rotating attachment point enables the user to fold the arms down for transportability. Additionally, the rotating attachment contains a hole through which the wires connecting the Raspberry Pi underneath the platform can be fed to the rest of the components.

Top: On the far end, there is a motor box which may be geared as necessary to produce enough torque to rotate the plant 180 degrees, flipping the plant. The motor uses a binder clip-like clamp. Binder clips have highly desirable properties that would make flipping the plant easy. These properties are a high pressure clamp which requires applied force to release. With this type of clamp, we will have full control over the orientation of the plant during operation with very little risk of losing our clamp grip during normal operation.

Bottom: The bottom is not portrayed in the figure. However, the bottom will be partially hollow exposing the Raspberry Pi (while keeping it off the ground) for the user to connect to later for data collection/management and charging the power source.

Note: The plant is portrayed with its stem in the clamp and the roots clearly unobstructed from the overhead camera.

Arms: The arms are PVC arms attached to the sides as well as attached to each other via a cross bar piece of PVC. PVC is an ideal material for this because PVC can be machined easily while having secure joint connections. This enables us to attach a camera mount to the underneath side of the crossbar in the ideal overhead position for image capture of the plants.

Blow-up Circle: The Blow-Circle in the figure illustrates that the Raspberry Pi can be wired to the rest of its components by sending the wires through the rotatable attachment point of the arms.

Finally, Figure 2.4-2 illustrates the block-diagram connections and interactions between the different components. The blue background illustrates mechanical components, while the green background illustrates electrical components.

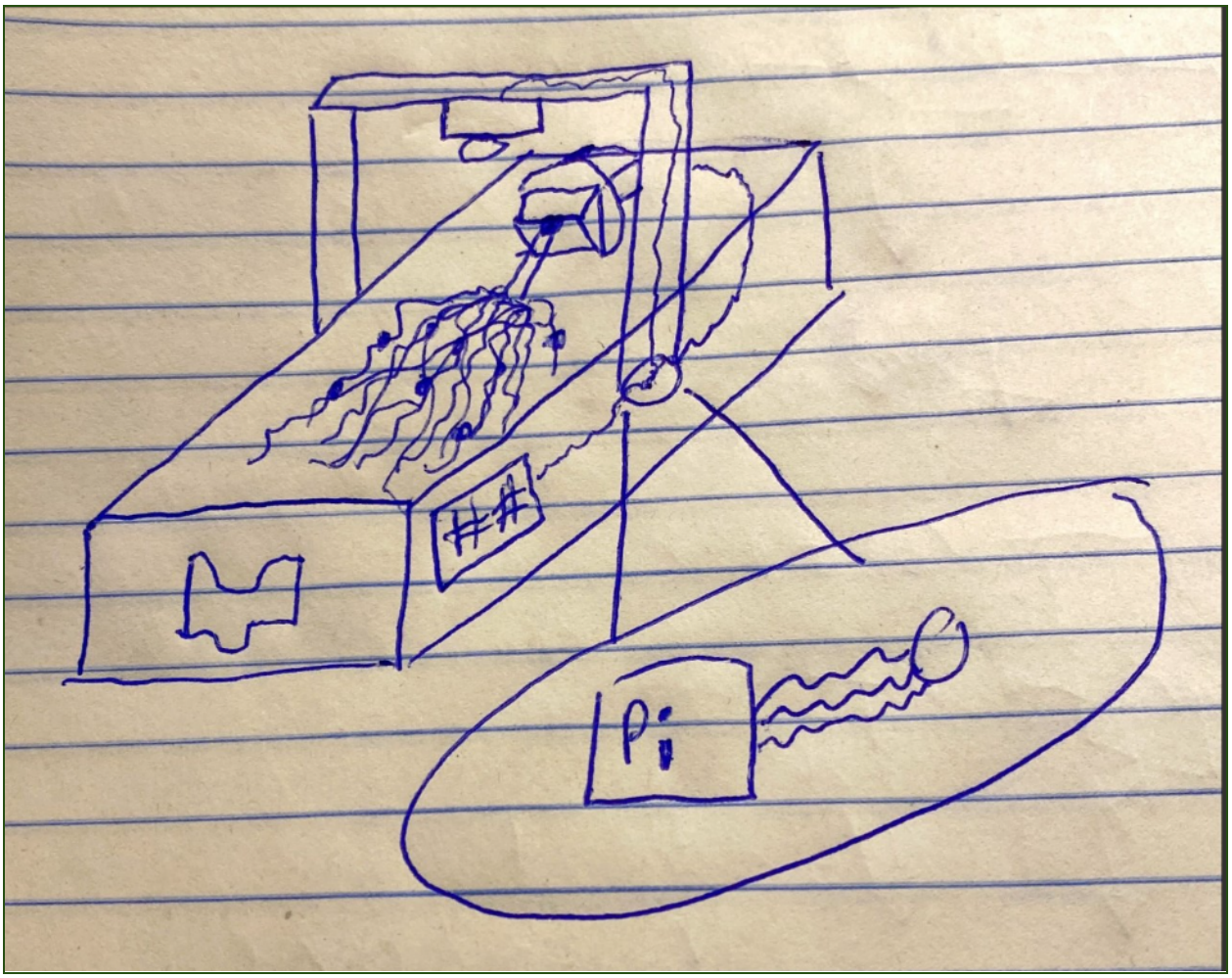


Figure 3.1-1: Conceptual Sketch of Soybean Root Scanner

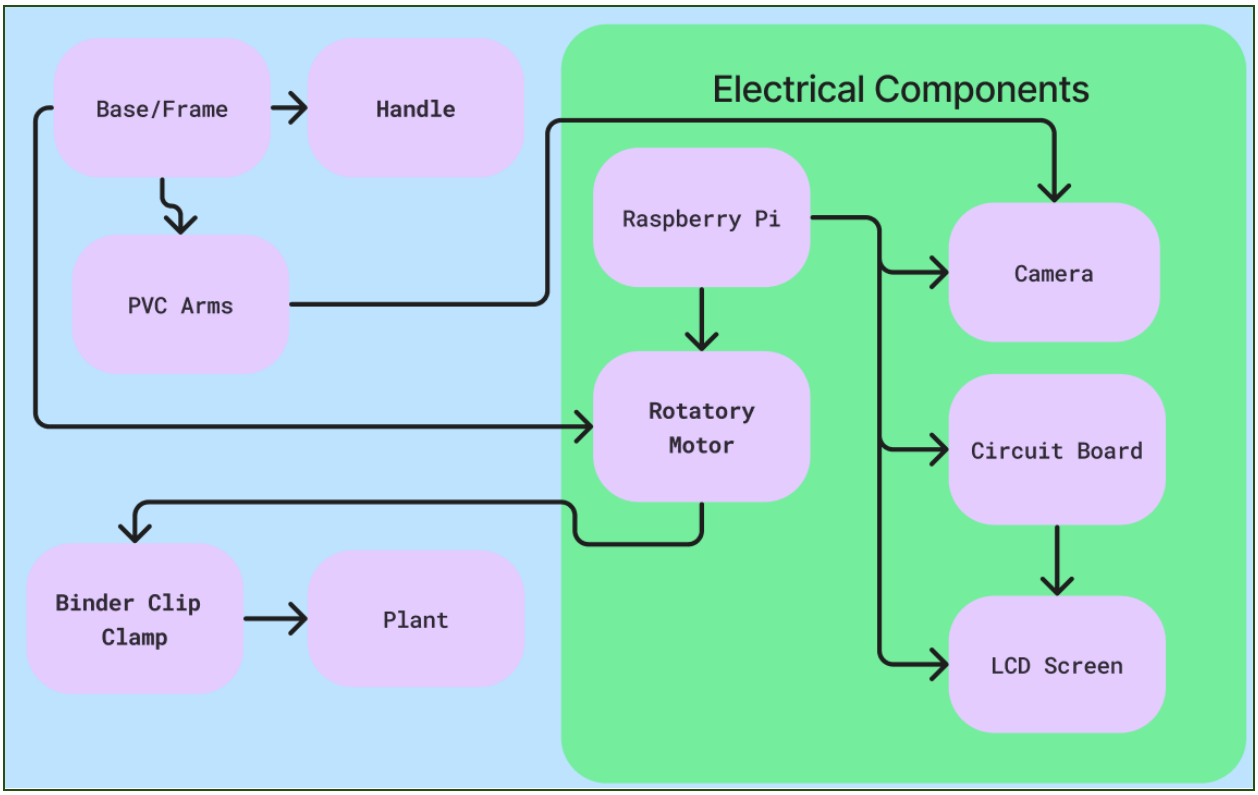


Figure 3.2-2 Systems-Level Diagram of Parasitic Cyst Detector

3.2 Detectron2 Implementation of Faster R-CNN

After researching several object-detection algorithms and deciding on the Faster R-CNN object detection algorithm, we had the task of finding and evaluating implementations. After coming across several GitHub pages referenced in scholarly articles, we came across a Facebook Research project by the name of Detectron2 [9].

Detectron2 is an open source Python library developed by Facebook's research team with several implementations of various object-detection algorithms, including Faster R-CNN. Since they provided high-quality tutorials on how to get started with their library and have a dedicated documentation repository, we decided to use and customize their tutorial to apply it to small object detection.

4 Evolution Since Cpr E 491

4.1 Hardware Evolution

Our final design is shown in the sketch below in figure 4.1-1. This design reflects more knowledge of the needs of the final user, as well as a better understanding of the feasibility of the hardware.

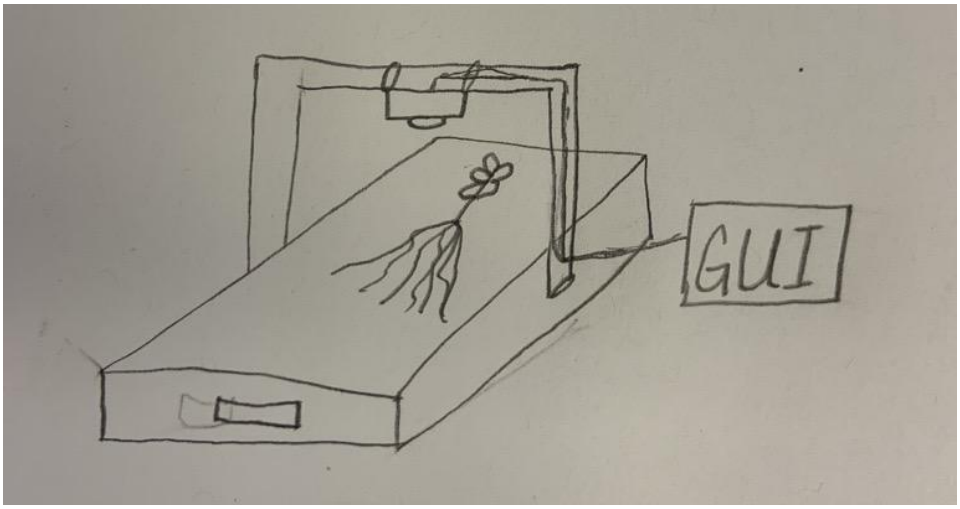


Figure 4.2-1 Updated Hardware Conceptual Sketch

Front: handlebar for ease of carrying the device around a field.

Bottom: The base of the device is a tray table for the soybean roots to lay on.

Arms: The PVC Arms are connected to the tray table to give a birds-eye view of the soybean plant. These arms are also extendable, so the user has the ability to zoom in to get a better view of the cysts or zoom out to see the entirety of the plant.

Top (connected to arms): Directly attached to the PVC pipe arms is a 3d printed custom camera mount. This camera mount will house the camera as well as the Raspberry Pi.

Side: A monitor is connected to the Raspberry Pi. This monitor displays a graphical user interface (GUI) to easily communicate with the machine learning model. The GUI will display a live feed from the camera so the user knows if the soybean roots are in frame. It will also display a button that takes a picture and sends it to the machine learning model. After the cysts are counted, the final number will be displayed on the GUI.

An updated block diagram is seen in figure 4.1-2 to illustrate the interaction between each mechanical and electrical component.

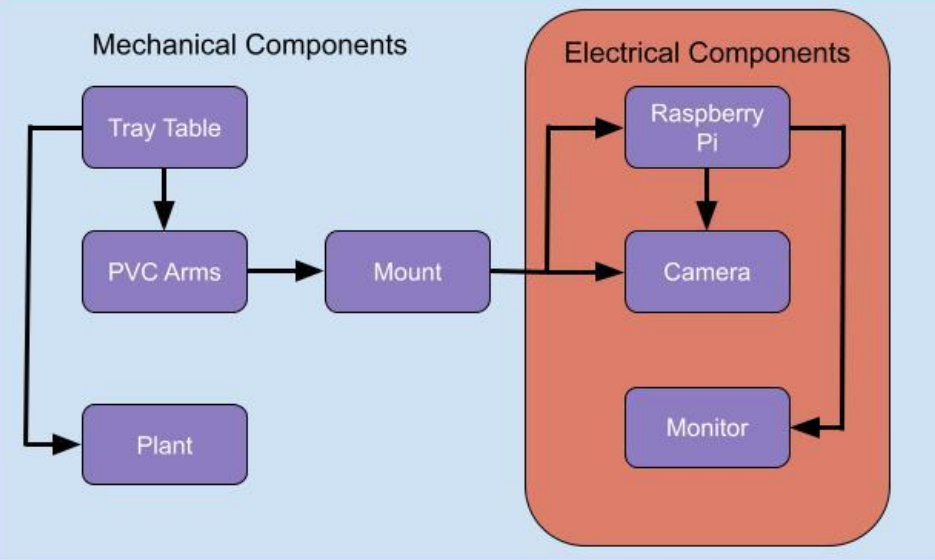


Figure 4.2-2 Updated Block Diagram

4.2 Software Tools Developed

We have developed several tools this semester to solve the issues we ran into along the way. The first, chronologically, was that the labelled data from Labelme, our dataset labelling software, was incompatible with the Detectron2 library. We wrote a script to consolidate the Labelme dataset into a single file (rather than individual label files per-image), and another script to transform that data into the correct format for loading into Detectron2, per their specifications [9].

The next challenge we encountered was actively loading our dataset into Detectron2 and running their training algorithm on it. To do this, we wrote a Python function that "registered" the dataset with the Detectron2 library, according to their documentation.

Finally, we wrote two more Python programs to train and test the algorithm. This involved installing dependencies on our dedicated Linux machine, loading the dataset into the Detectron2 library, registering it with the library, configuring hyperparameters, creating a trainer, and outputting checkpoint and final model weight files. To test, we developed a program that would load a trained model and run a predictor on our validation dataset.

5 Testing and Optimization

We developed a framework for testing and fine-tuning the computer-vision model and identified several hyperparameters we believe are key to adjusting this algorithm to be suitable for small-object detection.

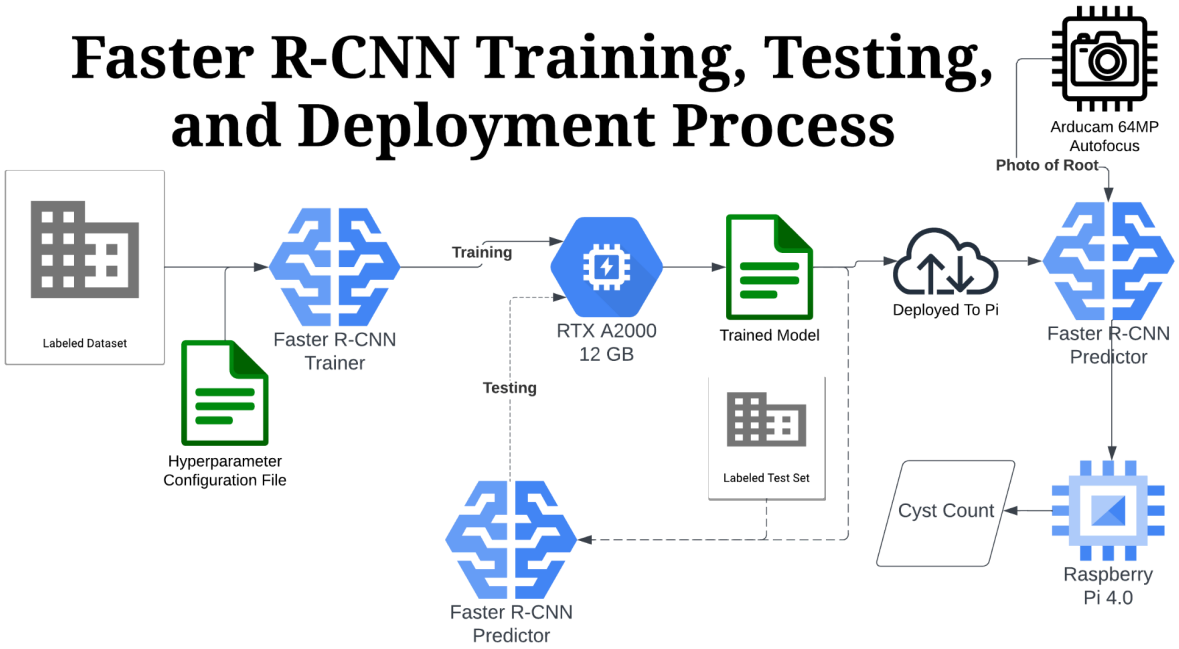


Figure 5.0-1 - Process diagram for fine-tuning hyperparameters and deploying the model

5.1 Notable Hyperparameters

A hyperparameter is a configuration setting used to control the training process of a machine learning algorithm. We determined these categories of hyperparameters to cause noticeable impact for small object detection through both research and testing. Due to lengthy training sessions, we did not have time to isolate and iteratively test each hyperparameter, so the hyperparameters can still be refined. For maximum impact, we adjusted a single category of hyperparameters for each training session and observed the effects on the resulting model.

Learning Rate

Controlling the learning rate affects how quickly and how close to convergence the model will become. We used a consistent learning rate of .002, slowing down after 20,000 iterations. This enabled us to observe the full training of the model from its early stages to its post trained stages. We did attempt changing the learning rate to .05 and stepping down regularly every 5,000 iterations. However, this made our results from that training session irregular and did not indicate improvements of convergence in the model. From our observations, the learning rate requires

consistent, expected behavior to be optimized. Thus, this hyperparameter should be independently refined for the best results.

Max Iterations

Max iterations is useful for determining overfitting and underfitting of the model for a specific learning rate. With small objects proving difficult to train a model on, we immediately tested when the model converged. Using a consistent learning rate of .002 for a higher number of iterations allowed us to determine underfitting before 5,000 iterations and overfitting after 30,000 iterations. The highest number false positives occurred consistently at 15,000 iterations. The best performance occurred consistently in between 20,000 and 30,000 iterations.

Detections Per Image

Detections per image is how many predictions the model is allowed to make per image. We found the default number of detections per image was too low for our use-case at 100. Consequently, we tested how increasing this value to 200, 400, and 1000 affected performance. We found that the higher the better. This enables the model to make many predictions and consequently catch more cysts. However, this coincided with a higher number false positives in every case. The alternative, lower predictions, is much worse as the model did not “see” many obvious cysts. This is likely due to false positives maxing out the prediction cap preventing the model from “seeing” all the cysts. For best performance, more than 1000 detections per image should be allowed.

Anchor Box Sizes & Aspect Ratios

Anchor boxes are pre-sized bounding boxes used to interpret regions of interest, or areas that likely contain a cyst. They are used to create a better fit to the objects being detected by scaling the region of interest bounding boxes in a certain direction. Standard object detection allows a minimum of 32x32 pixels for anchor box sizing. This posed problematic for us as our cysts were often 16x16 pixels on a high resolution scanner. Consequently, we attempted to shrink the anchor boxes down to 16x16 and even 8x8 pixels. The results were misleading. This lead to higher detections and better performance but only psuedonamously. The anchor boxes ended up stacking (see Fig 5.1-1) on top of the same cyst creating multiple predictions for the same cyst. With 8x8 sizing, this problem was egregious. With 16x16 sizing, the problem was rare and limited to 2 predictions for the same cyst. In addition to having more square aspect ratios of 0.75, 1.0, and 1.5, this ended up being a net improvement to the performance of the model.

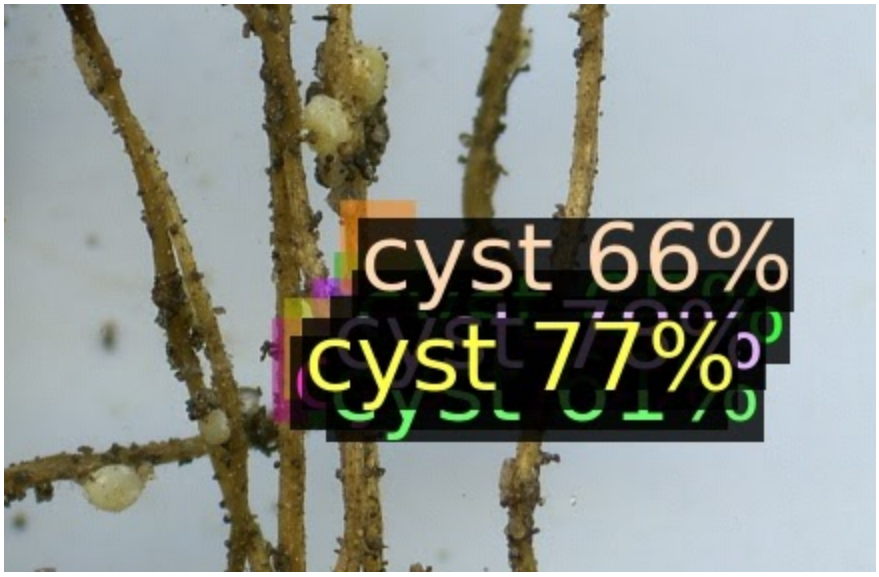


Figure 5.1-1 8x8 Anchor Box Stacking

Maximum Feature Size
This is a conglomerate of several hyperparameters that all serve the same purpose: how large of a feature can be passed into the region proposal network. Essentially, the feature extraction pyramid network breaks the image into a grid of cells allowing one prediction per cell in the image. By increasing the hyperparameters, we effectively shrink the size of the cells allowing densely populated areas of cysts to have more distinguishment between neighboring cysts. Consequently, performance improved using a smaller maximum feature size.

6 Results

6.1 Hardware Results

The first iteration of a prototype was created. The prototype has most of the functionality requirements specified in section 1.4 of this document. The device is small, easy to transport, and has the capability to be wirelessly powered. It is affordable, with a cost of around $450 to produce. However, the end user is required to interact with the soybean plant by flipping the plant 180◦ to capture both sides. Also, the camera is not a high enough resolution to see the cyst nematodes without a zoom feature. A physical zoom in and out feature was put in by having extendable arms.

As mentioned in section 4.1 of this document, a custom camera mount was needed to complete the hardware prototype. The part below was created in Solidworks and 3-d printed:

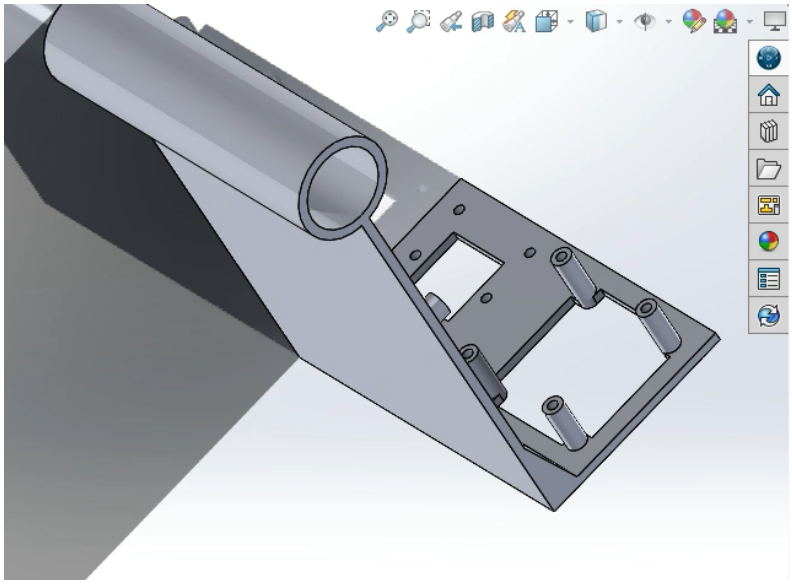


Figure 6.1-1 3-d Printed Camera Mount

A first iteration prototype is show below in figure 6.1-2

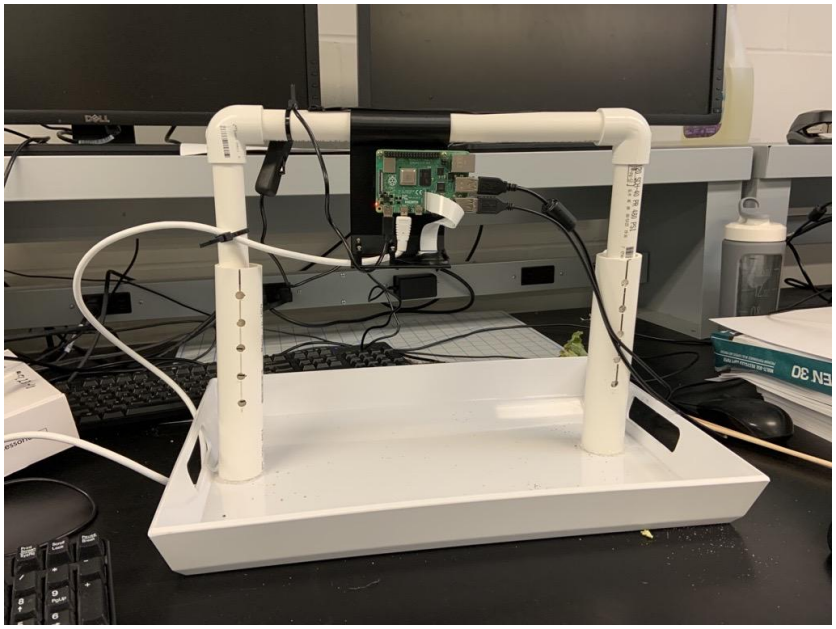


Figure 6.1-2 Hardware Prototype

6.2 Graphical User Interface Results

The GUI was designed to meet a requirement described in section 1.4 of this document stating the end user should be able to use this product with no formal training. The GUI pictured in figure 6.2-1 allows the user to get a cyst count with the push of a button.

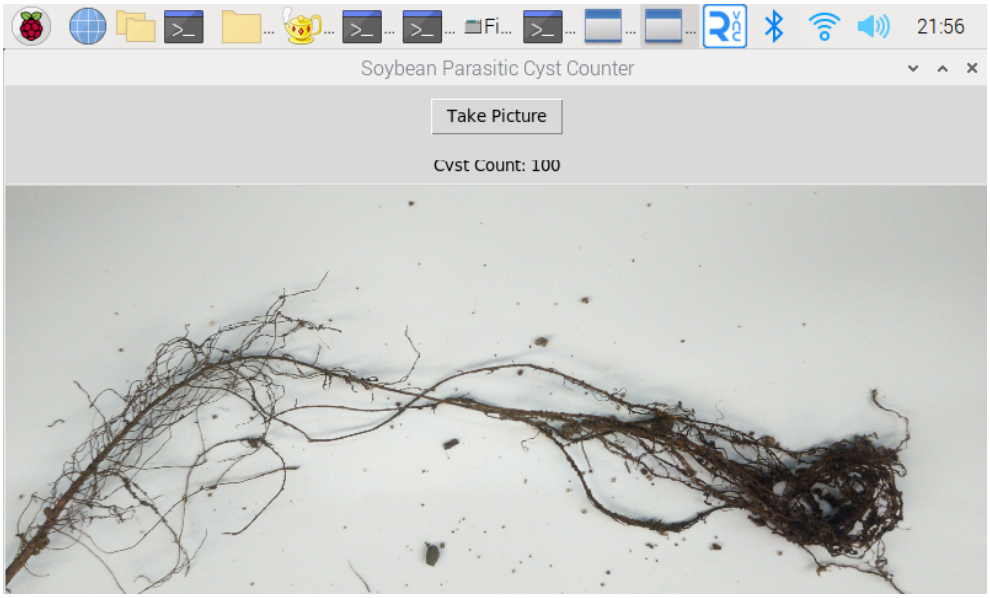


Figure 6.2-1 Graphical User Interface for Hardware Prototype

An operations manual has been created for this product and is linked in the Appendix.

6.3 Integrating Detectron 2 to Raspberry Pi 4 Results

Integration was done on the Raspberry Pi 4 by using dependencies of the Detectron2 library. Pytorch, Torchvision, and OpenCv were required in order to run Detectron2. In order to successfully download all the required libraries, Raspberry Pi had to be in the Bullseye version with 64 bit, and aarch64 which is AMD. Piwheels were used to pull files from the githubs and download in the Raspberry Pi. After the environment was successfully built, it was possible to move the trained model (pth) file and configuration files (yaml) file from our local machine to the Raspberry Pi using the sd card reader. With the sample cysts image from the Scanner, our machine was able to return cysts count.

However, it was possible to observe that usage of the CPU was intense while processing the predictor function from the Detectron 2 library. The process took more than 10 seconds per image, and I was able to observe that the CPU of the Raspberry Pi gets warm.

6.4 Detectron2 Optimization Results

This section details the results of changing the hyperparameters covered in section 5.1. But first, it is important to discuss how to evaluate a computer vision models. Standard analysis techniques use precision and accuracy for analyzing models. Accuracy is the ratio of correct predictions to

the total number of predictions. Precision is the ratio of true positives to ratio of total positive predictions. A true positive (see Fig 6.4-1) is a correctly predicted cyst while a false positive (see Fig 6.4-2) is a an incorrectly predicted cyst. In our analysis, we focused more on precision rather than accuracy.

Figure 6.4-1: True Positive Cyst Prediction

Figure 6.4-2: False Positive Cyst Prediction

Unfortunately, the standard analysis techniques showed our initial results were quite unsuccessful. This is seen by the near 0% Average Precision (AP) score using the default config shown in Table 6.4-1. So, using standard analysis was less useful in determining whether the hyperparameter changes were having a positive effect. Consequently, more manual analysis of looking at each prediction image versus the original to determine how successful each prediction was became necessary. This was combined with analyzing the total predictions shown in Chart 6.4-1. By using this analysis, we were able to determine the hyperparameters from section 5.1 that contributed to the low AP score and how to improve them.

	5k iter.	10k iter.	15k iter.	20k iter.	25k iter.	30k iter.	35k iter.	40k iter.
Default Config	0.002	0.002	0.001	0.001	0	0	0.008	0
SmallerAnchorsConfig	0.025	0.002	0	0.001	0.001	0.006	0	0
SmallerTiling	0	0.001	0	0.002	0.001	0.001	0	0
EvenSmallerTiling	0.005	0.005	0.005	0.004	0.002	0.002	0.002	0.002
SmallerAnchorsW/SmallTiling	0.002	0.003	0.006	0.003	0.119	0.017	0.025	0.025
SmallAnchorsW/SmallTilingLearningRateEnlarged	0.005	0.008	0.011	0.01	0.009	0.009	0.009	0.009

Table 6.4-1 AP Scores for Various Training Configurations.

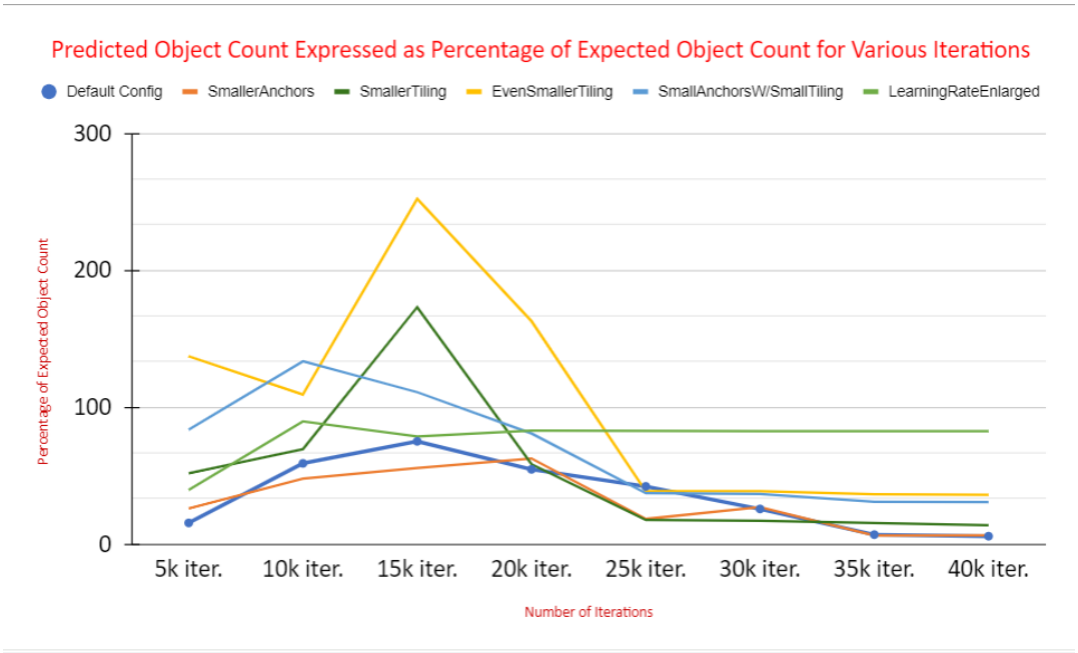


Chart 6.4-1 Predicted Object Instances versus Expected Object Instances.

From Chart 6.4-1, we observe that underfitting occurs at less than 5,000 iterations, while overfitting regularly occurs at 30,000 iterations. Additionally, we observe the hyperparameters that increase the false positive count. Smaller tiling, or reducing the maximum feature size, combined with increased maximum predictions per image yielded the largest increases in false positives giving 150% and 250% predictions over the expected for two different configurations. Manual analysis of the small anchors configuration was used to determine that many of the predictions were stacked predictions, hence the overall low AP scores. But interestingly, combining these two configurations yielded a moderate prediction ratio with the highest AP score of 11.9% from Table 6.4-1.

This perfectly illustrates our difficulty with finding the optimal hyperparameter values as the largest increase came from a combination of hyperparameters rather than increasing or decreasing one to find the perfect value. Thus, our efforts went to identifying key hyperparameters over optimizing their values. Consequently, the results illustrate the importance of the chosen hyperparameters to small object detection and that with optimal values, small object detection is possible using Faster-RCNN.

Closing Statements

7.1 Conclusion

During the course of this project, we were able to accomplish most of the goals that were given at the beginning of this project. The goals of this project were to have a Machine Learning Model, a working hardware prototype, and successful integration between the two portions of our project. The three main goals were accomplished to some degree throughout the duration of the year. We also met many of our functional requirements. The software does not require the use of the internet, and the hardware is portable and does not damage the plant. Also, every non-functional requirement given to us by our client. The product requires no formal training, could be sold for less than $500, has no fatal errors during operation, is durable and can be operated outdoors, and is semi-autonomous.

Regarding the portions that have not been completed, a framework has been set up to optimize the project further in the future. The framework includes a labeled data set, and documentation describing goals moving forward with the project. With more trials with different configurations of hyper-parameters, the accuracy of the model should increase above the 50% accuracy goal. Also, more market research can be conducted to find a higher resolution camera that matches the 100 megapixel scanner that was to create the dataset.

Our device allows for easy and fast image capturing to find the cysts soybean roots. Our method provides a way to automate the process of counting cysts on a soybean plant and will help researchers, and eventually farmers, learn how to prevent soybean cyst nematodes.

7.2 References

[1] Brooks, Rhonda. “New Program Tackles the $1.5 Billion Bite SCN Takes out of U.S. Soybeans Annually.” *AgWeb*, 18 July 2022, https://www.agweb.com/news/crops/soybeans/new-program-tackles-15-billion-bite-scn-takes-out-us-soybeans-annually.

[2] Gamage, Omega. “Car Detection from a Drone: A New Angle.” *Medium*, ACCELR Blog, 10 Sept. 2021, https://medium.com/accelr-blog/car-detection-from-a-drone-a-new-angle-821cc6fff422.

[3] S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” Cornell University ArXiv, 2016 [Online]. Available: https://arxiv.org/abs/1506.01497v3

[4] "American National Standard for Metric Practice," in ANSI/IEEE Std 268-1992 , vol., no., pp.1-80, 28 Oct. 1992, doi: 10.1109/IEEESTD.1992.114377.

[5] "IEEE Standard for DevOps:Building Reliable and Secure Systems Including Application Build, Package, and Deployment," in IEEE Std 2675-2021 , vol., no., pp.1-91, 16 April 2021, doi: 10.1109/IEEESTD.2021.9415476.

[6] "IEEE Guide--Adoption of ISO/IEC TR 24748-1:2010 Systems and Software Engineering--Life Cycle Management--Part 1: Guide for Life Cycle Management," in IEEE Std 24748-1-2011 , vol., no., pp.1-96, 3 June 2011, doi: 10.1109/IEEESTD.2011.5871657.

[7] "ISO/IEC/IEEE International Standard - Software engineering - Software life cycle processes - Maintenance," in ISO/IEC/IEEE 14764:2022(E) , vol., no., pp.1-46, 21 Jan. 2022, doi: 10.1109/IEEESTD.2022.9690131.

[8] “Nvidia embedded systems for next-Gen Autonomous Machines,” *NVIDIA Jetson: Accelerating Next-Gen Edge AI and Robotics*. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/. [Accessed: 22-Apr-2023].

[9] Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., & Girshick, R. (2019). Detectron2. Retrieved from https://github.com/facebookresearch/detectron2

6.3 Appendix

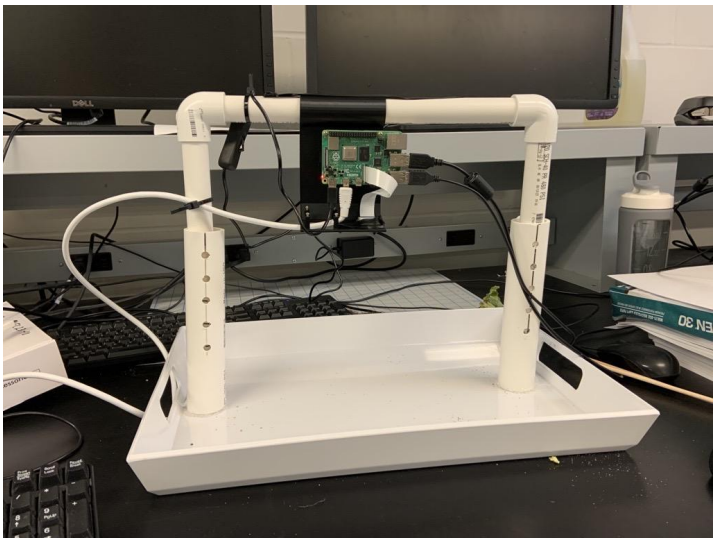
Operations Manual
Overview:

This appendix contains the steps necessary to operate the root scanning device and deploy the GUI to get a count of parasitic soybean cyst nematodes. This process can be done indoors in a lab setting, or outdoors on a soybean plant farm.

Step 1: Setting Up Hardware

Before deploying the GUI, you must set up the hardware.

1.1: First, connect your Raspberry Pi and 7 inch monitor to a power supply. This could be via wall outlet if you are indoors, or if you require a cordless setup, a USB battery pack will supply enough power to operate the device. The Raspberry Pi requires a USB-C connector, and the monitor requires a micro-USB connector.



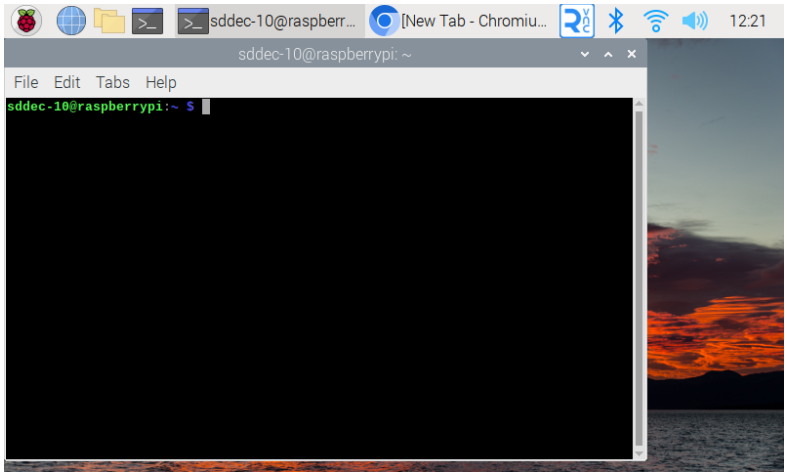
You will know everything is connected properly when a red LED on the Raspberry Pi lights up and the monitor displays the desktop below:

1.2: Lay the soybean roots on the tray

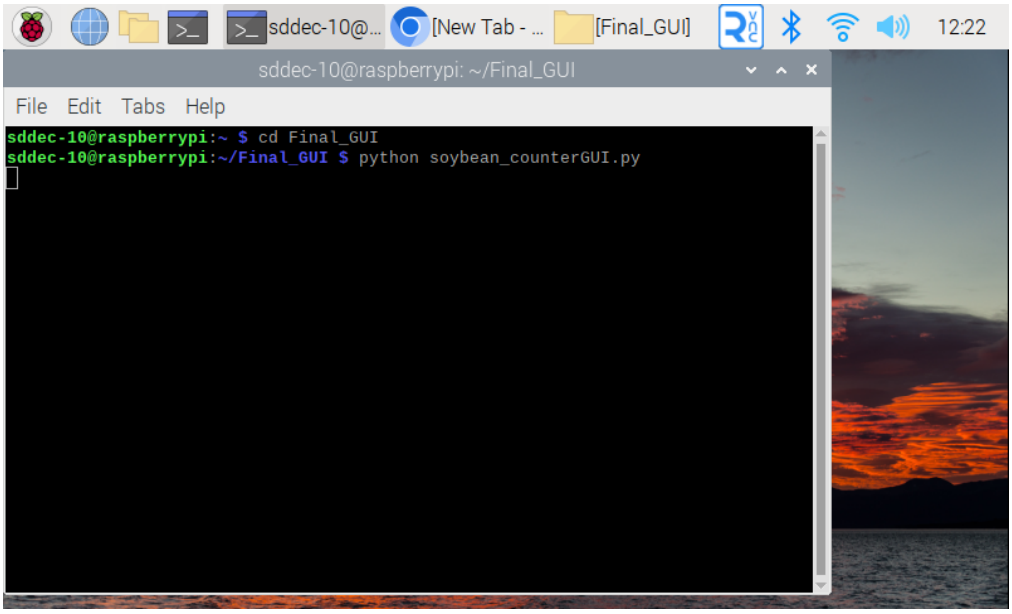
Step 2: Deploy the Graphical User Interface

The Graphical User Interface (GUI) will allow you to communicate with the machine learning model. The deployment of the GUI requires some interfacing with the Linux command line.

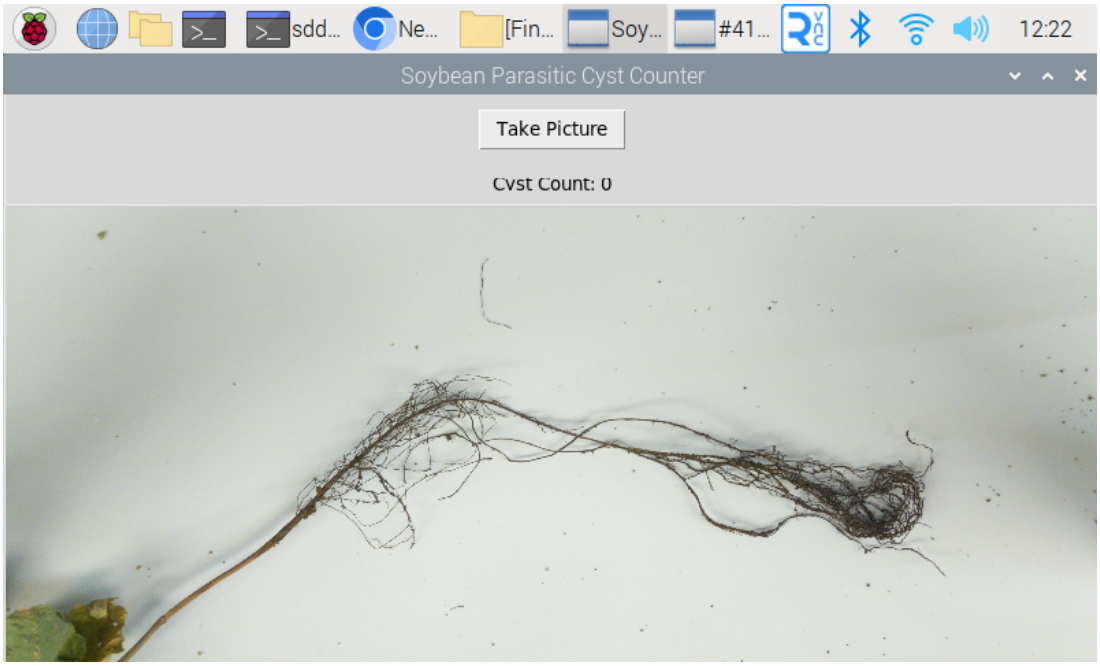
2.1: Open the Linux command window in the upper left portion of the desktop



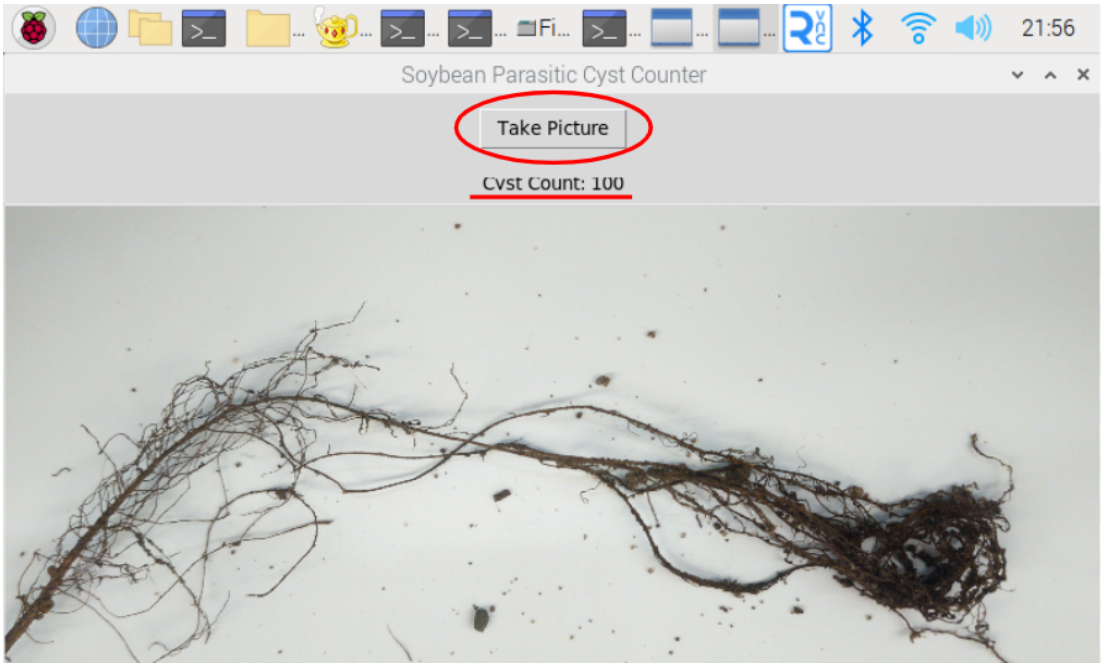
2.2: Run the command “cd Final_GUI” without the quotation marks. This will redirect you to the folder that contains the cyst counter GUI. Then, run the command “python soybean_counterGUI.py” without the quotation marks. This will launch the graphical user interface. An example of the



2.3: Ensure the soybean roots are visible to the camera. There is a live feed that will be shown upon launching the GUI. There is a motor in the camera that will automatically focus the camera



2.4: Once the photo is in focus, press the “Take Picture” button and wait for the number of cysts to be reported next to “Cyst Count: “



Background Research Compilation Document:
https://docs.google.com/document/d/1mITfIIbpWDMYVwMXOBxhz1OOZPWPzlSX5o5VofkQcLU/edit?usp=sharing

Algorithm Selection Research Compilation Document:
https://docs.google.com/document/d/1RJ1XpO6BM6Xi7iS9vB-CeDnzz7i8VdGND5rzunaQNCM/edit?usp=sharing

Faster R-CNN Research Compilation Document:
https://docs.google.com/document/d/1BST4wBlz1tL_HMQkdW3w6Xb2519I7nstuGS6UsmCbjQ/edit?usp=sharing

You must be logged in to your Iowa State account to access these documents.