EE/CprE/SE 491 - sddec23-10

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

Week 5 Report

02/27/2023 – 03/5/2023
Client : Professor. Santosh Pandey
Group number: 10

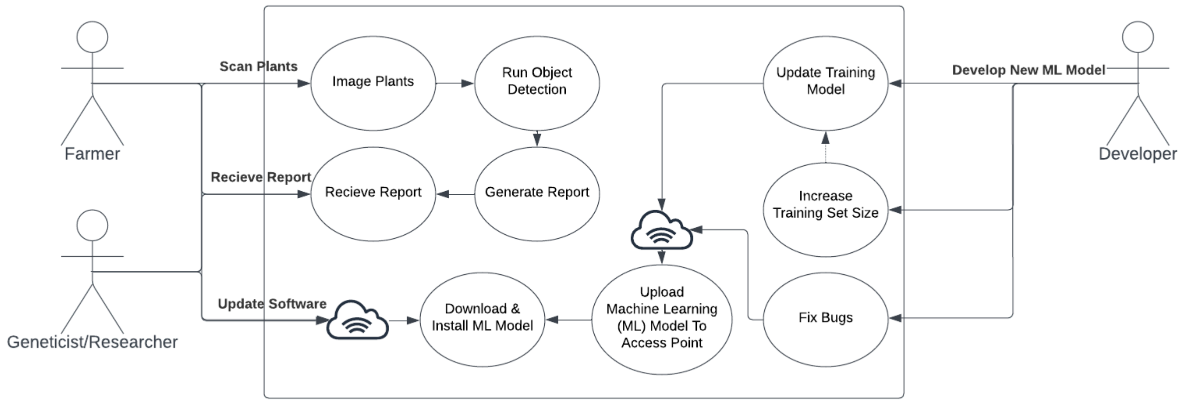
Team Members:

Katherine Moretina
Ethan Baranowski
Chris Cannon
Matthew Kim

Past week Accomplishments

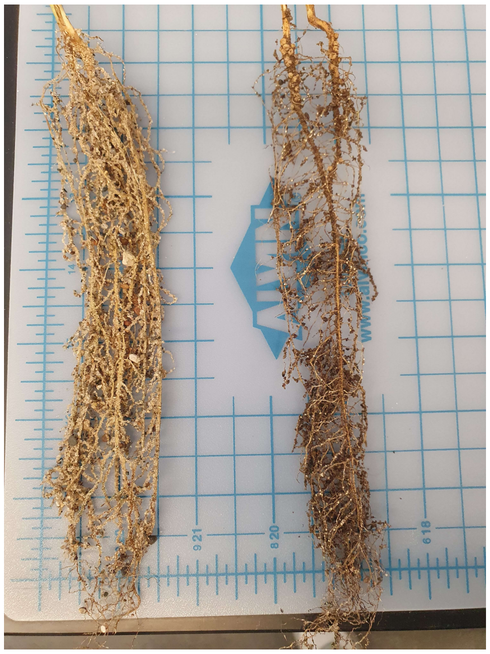
- Group Discussion with client outlining and documenting the system specifications (Hardware and Software).
- Image collection process demonstration by Yunsoo Park.
- Additional team member gained as an undergraduate researcher assigned to the project (introductions will be soon)
- In depth analysis of YOLO vs SSD vs Faster R-CNN algorithms yielding selection of Faster R-CNN as algorithm of choice.
- Began code practice tasks where we are looking through Faster R-CNN implementations for understanding and translation into our program.
- Compiled multiple research documents that outline the various phases of our research investigations and reasoning selection for that algorithms.

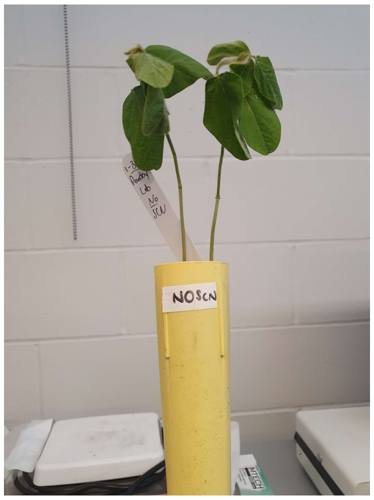
Use Case Diagram

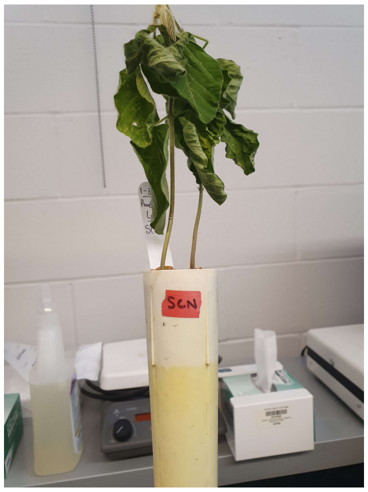


Scanning Stages (Dataset preparation) - Everyone

Get Samples. Half of them are without nematodes and the other half are controlled.

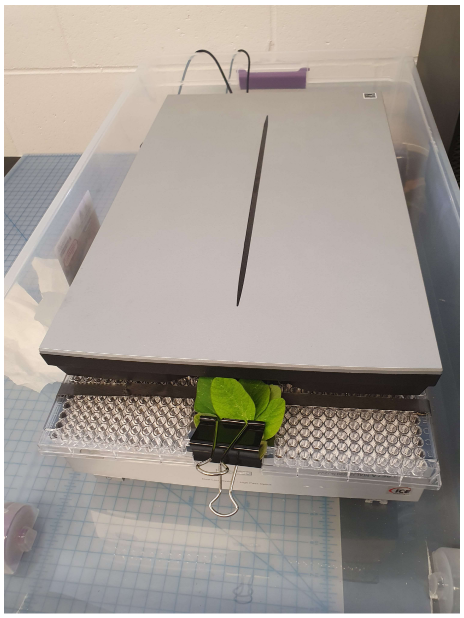


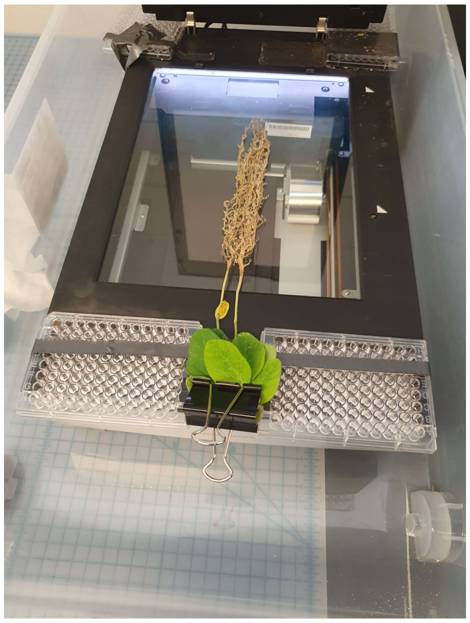




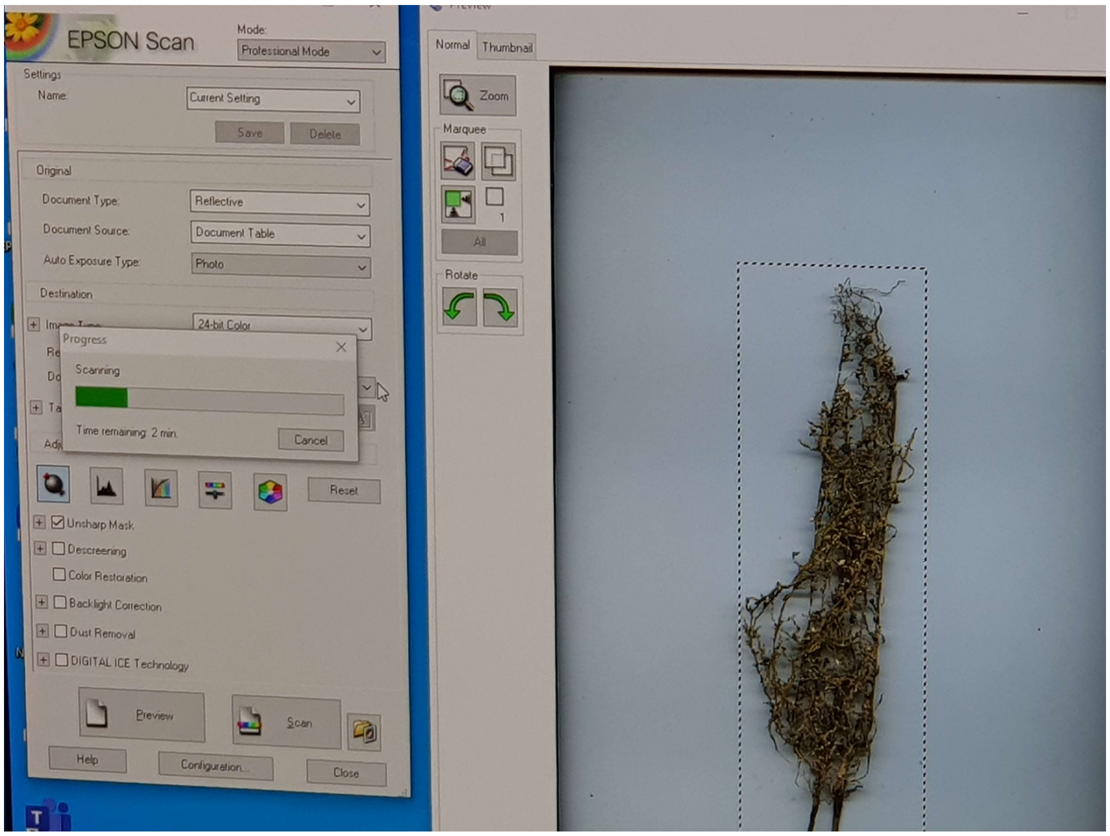
Left without / Right with

Use scanner to scan four sides (Front, Back, Left, Right)





Select area to be scanned



EPSON Scan

Mode: Professional Mode

Settings

Name: Current Setting

Save Delete

Original

Document Type: Reflective

Document Source: Document Table

Auto Exposure Type: Photo

Destination

Image Type: 24-bit Color

Progress

Scanning

Time remaining: 2 min.

Cancel

Reset

☑ Unsharp Mask

☐ Descreening

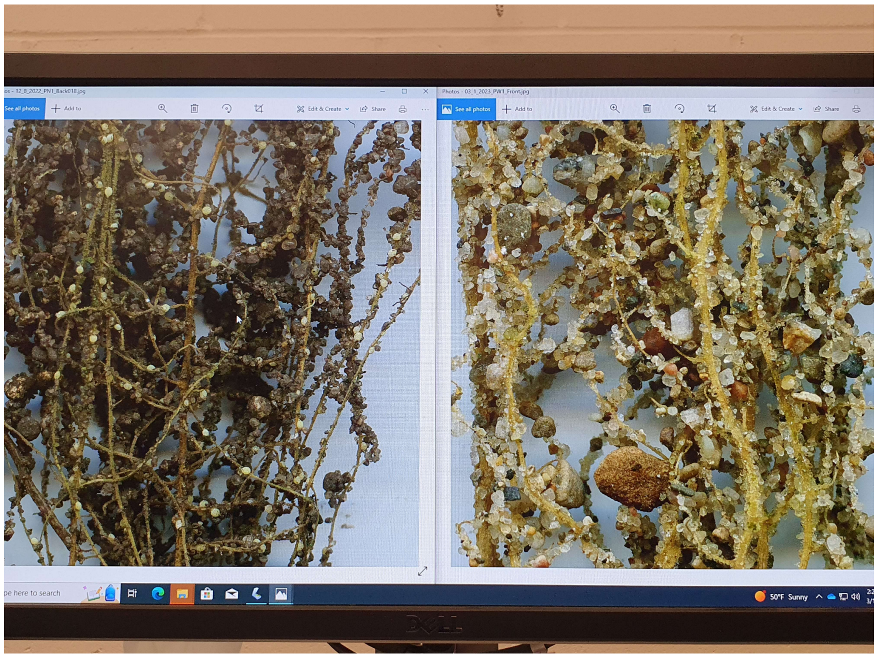
☐ Color Restoration

☐ Backlight Correction

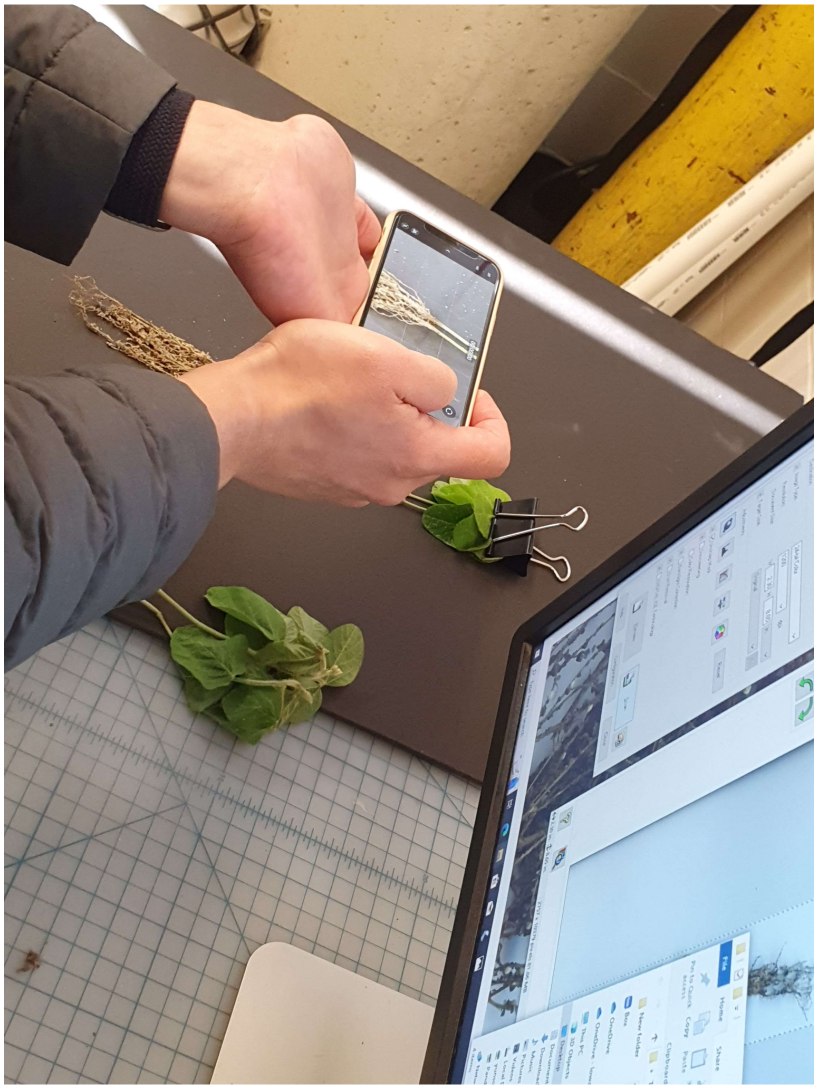
☐ Dust Removal

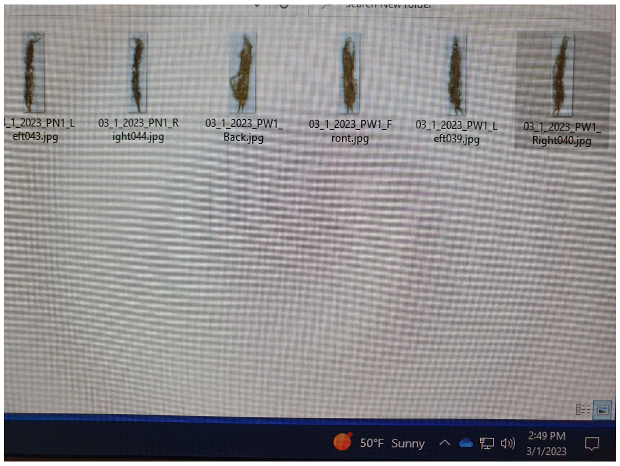
☐ DIGITAL ICE Technology

Left-With cysts / Right - without



Also use smartphone to take picture and video





_1_2023_PN1_Left043.jpg
03_1_2023_PN1_Right044.jpg
03_1_2023_PW1_Back.jpg
03_1_2023_PW1_Front.jpg
03_1_2023_PW1_Left039.jpg
03_1_2023_PW1_Right040.jpg
50°F Sunny
2:49 PM
3/1/2023

Faster R-cnn focused research - Matthew

Examples of Faster R-CNN Process
Bring data and separate that data into training and testing.

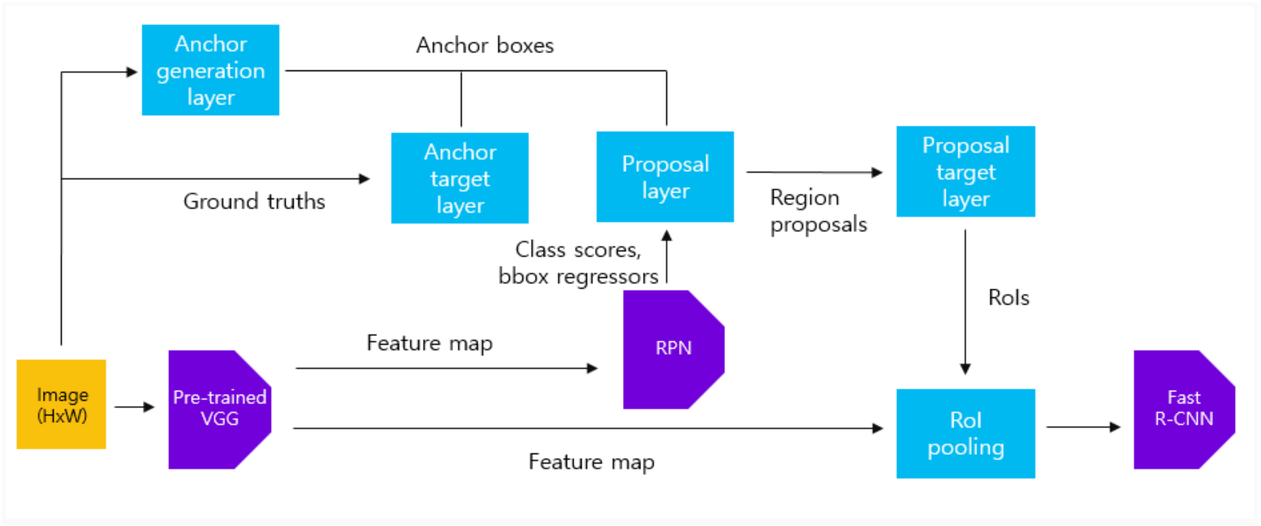
How to setup Faster R-CNN using anaconda
https://pub.towardsai.net/training-faster-r-cnn-using-tensorflow-object-detection-api-with-a-custom-dataset-88dd525666fd

1. Create virtual environment and active anaconda
2. Install tensorflow GPU
   a. conda install tensorflow-gpu==1.15.0
3. Upload the Tensorflow model file. (Soybean files)
4. Put the Faster R-CNN Inception V2 model in the object detection folder

How FasterRCNN works and step-by-step PyTorch implementation
https://www.youtube.com/watch?v=4yOcsWg-7g8

Example faster R-CNN implementation
https://github.com/herbwood/pytorch_faster_r_cnn/blob/main/faster_r_cnn.ipynb

This example uses Pytorch to implement the R-CNN.
Single image example



1. Pre-training.
   a. Define a pre-trained model. Then from all models, get a sub-sampling ratio of 50x50 sized layers.
2. Anchor generation layer
   a. In the anchor generation layer, the author generated an anchor box.

   b. Image size was 800x800, therefore sub-sampling ratio is 1/16, which total of 22500 anchor box needs to be created.
   c. In anchor_boxes coordinate of the anchor box was saved. (x1, y1, x2, y2)

```
anchor_boxes[index, 1] = ctr_y - h / 2.
anchor_boxes[index, 0] = ctr_x - w / 2.
anchor_boxes[index, 3] = ctr_y + h / 2.
anchor_boxes[index, 2] = ctr_x + w / 2.
```

3. Anchor Target layer
   a. Selects anchor box to train RPN.
   b. Only selects image that are inside boundary
4. Get Ground truth box and IoU for the anchor box.
   a. For each ground truth box and iou, if their value is greater than 0.7, save that anchor box to positive sample, and if less than 0.3, set them into negative sample.
   b. Save positive sample into 1, and negative sample into 0.
5. RPN(Region Proposal Network)
   a. Define RPN.
   b. Calculate the loss.
6. Proposal layer
7. Proposal Target layer
8. RoI pooling
   a. Use feature map and the proposal target layer from the feature extractor.

Faster RCNN Github Research - Chris Cannon

Code Analysis

https://github1s.com/rbgirshick/py-faster-rcnn

- This implementation uses pycaffe and Caffe instead of Pytorch.
- It has lots of examples, documentations, and tutorials, under caffe-fast-rcnn/examples
- Has two modes - classify and detect, with their own files
  - caffe-fast-rcnn/python/classify.py
  - caffe-fast-rcnn/python/detect.py
- Focusing on detect, since we don’t necessarily need to classify for our project.
- detect.py
  - creates a detector from ./caffe/detector.py
  - handles i/o
  - Calls the detector’s relevant detect function
    - detector.detect_windows()
    - detector.detect_selective_search()
  - Produces output based on detector’s behavior
- detector.py

    - detect_windows()
        - Do windowed detection over given images and windows. Windows are extracted then warped to the input dimensions of the net.
        - Parameters:
            - images_windows: (image filename, window list) iterable.
            - context_crop: size of context border to crop in pixels.
        - Returns:
            - detections: list of {filename: image filename, window: crop coordinates, predictions: prediction vector} dicts.
        - Process:
            - Loads each image in images_windows as a matrix/array of floats
                - For each image, for each associated window, crops the image to the window, and appends that result to an array called window_inputs[]
                - Preprocesses it to change dimensions to match what is needed
                - Does some magic to create predictions, I couldn't find where it happened.
            - Packages predictions with images and windows
            - Returns that list
    - detect_selective_search()
        - Essentially just runs detect_windows() on a list of images with a specific list of windows.

Packages Research

Argparse

- The argparse module makes it easy to write user-friendly command-line interfaces by allowing the program to define arguments, and argparse will handle finding them from sys.argv. Also automatically generates help and usage messages.

Caffe

- Caffe is a deep learning framework from Berkely AI Research (BAIR)
- pycaffe is a python library that makes use of Caffe after using `import caffe`:
    - `caffe.Net` for loading, configuring, and running models.
    - `caffe.Classifer` and `caffe.Detector` provide convenience interfaces for common tasks
    - `caffe.io` handles i/o with preprocessing and protocol buffers
    - `caffe.draw` visualizes network architectures
    - Caffe blobs (Binary Large OBjects) are exposed as numpy ndarrays for ease-of-use and efficiency.

Selective_search_ijcv_with_python

- This is a package that wraps a matlab functionality described here for use in Python.
- The goal of selective search is to intelligently generate potential windows, instead of exhaustively generating windows over the entire image.
    - I believe that it uses pixel values to detect roughly homogenous regions quickly

Faster RCNN Implementation and Python Package Research-Katherine

Code Analysis

https://github.com/jwyang/faster-rcnn.pytorch

- Uses Pytorch instead of Jupiter Notebook
- Supports 3 pooling methods- Roi pooling, roi align, and roi crop
- Prerequisites
    - Python 2.7 or 3.6
    - Pytorch 0.4.0
    - CUDA 8.0 or higher
- Has a train, demo, and test set
- 3 main steps
    - Generating region proposals
    - From each region proposal, a fixed-length feature vector is extracted using image descriptors
    - The feature vector is used to assign each region proposal to either background or object classes
- Went through demo.py to find the stages
    - Creates an array for different classes line 168
    - Creates architecture line 188 and loads model after
    - Everything is in faster-rcnn.pytorch/lib/model/faster_rcnn/faster_rcnn.py
    - Define region proposal on line 30
    - Find weights on line 116- feature extractor
    - Evaluates Detections on line 325 of demo.py

Package Research

- Researched 3 different python packages
    - Import torch
        - The package allows code to use PyTorch
    - Import sys
        - Provides functions and variables that manipulate the python runtime environment

- Import pdb
    - Python debugger- provides an interactive source code debugger

Individual Contributions

Member	**Tasks Completed**	**Hours This Week**	**Total Hours**
Katherine Moretina	Went to all required meetings- learned how to get images of soybean roots, general meetings. Researched RCNN Implementation and three python packages.	5	19
Matthew Kim	Attended regular meetings to discuss which algorithm to choose. Also learned the process of preparing datasets. How to scan soybeans. Researched on R-cnn focused.	5	15
Chris Cannon	Attended regular meetings, did in-depth research on an RCNN implementation and three python packages. Created use-case diagram.	6	18
Ethan	Continued task development and deployment for iterative progress on algorithm selection and development. Attended group meeting for discussion on system specifications and requirements for both the algorithm (software) and application implementation (hardware). Attended lab meeting where Yunsoo Park walked us through the image collection process and discussed possible optimizations. Talked with Prof. Forrest Bao (Machine Learning Expert) for feedback on project progress and development. He clarified we have enough features across the 1000 nematode images to loosely train a CNN or properly train a lesser algorithm called SIFT (to be investigated).	10	23

Plans for Coming Week

- Investigate R-CNN implementations and create a baseline algorithm for us to modify for our purposes.
- Start labeling data with Label Studio software for training set. (reduction from 2000 images to 1000 images as only the cysts are required for feature training).
- Have Yunsoo Park walk us through coding on the lab computer.
- Setup Jupyter Notebooks server for student collaboration.
- Investigate SIFT machine learning algorithm for possible simplified object detector that will help simplify algorithm training and implementation.