AUTOMATED EARLY DETECTION OF PLANT DISEASES USING DEEP LEARNING

FINAL PROGRESS REPORT

A graduate project proposed in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

By

Sree Divya Sudagoni

Student ID: 202617109

Email: sree-divya.sudagoni.238@my.csun.edu

Project Title

Automated Early Detection of Plant Diseases using Deep Learning

Project Goal

The primary objective of this project is to leverage the power of YOLO and Mask-RCNN object detection models, along with a trained CNN, to automate the detection and classification of plant diseases. By enabling early identification and intervention, the project aims to contribute to improved crop health monitoring and effective disease management practices.

Status:

The project's timeline that is proposed is below

Period	Activity	Duration
1st Quarter (January-March)	Surveying existing problems on data-driven and precision agriculture.	1 week
	Obtain remote sensing data and other ground-based observations datasets.	2 weeks
	Preprocess the data to remove any inconsistencies.	3 weeks
	Conduct exploratory analysis to get a deeper understanding of the data.	2 weeks
2nd Quarter (April-June)	Perform feature extraction on the combined dataset.	3 weeks
	Train and evaluate multiple deep learning models using a portion of the dataset.	4 weeks
	Select the best-performing model based on evaluation results.	1 week
	Refine the selected model and prepare for the next stage of testing.	2 weeks
3rd Quarter (July-September)	Evaluate the refined model on the remaining dataset.	3 weeks
	Compare the results with the initial evaluation results	1 week
	Analyze the results and make any necessary adjustments to the model.	2 weeks
	Prepare for the final stage of testing and analysis.	1 week
4th Quarter (October-December)	Apply the model to real-world data and access its performance in a real-world setting.	2 weeks
	Analyze the results and draw conclusions.	2 weeks
	Write up the results, methodology, and conclusion in a final report and thesis paper.	4 weeks
	Prepare for necessary revisions and final submission of thesis.	1 week

The overall project health is Green. I planned to finish training my models by the end of June. The percentage of work that I expected to be completed by today is:

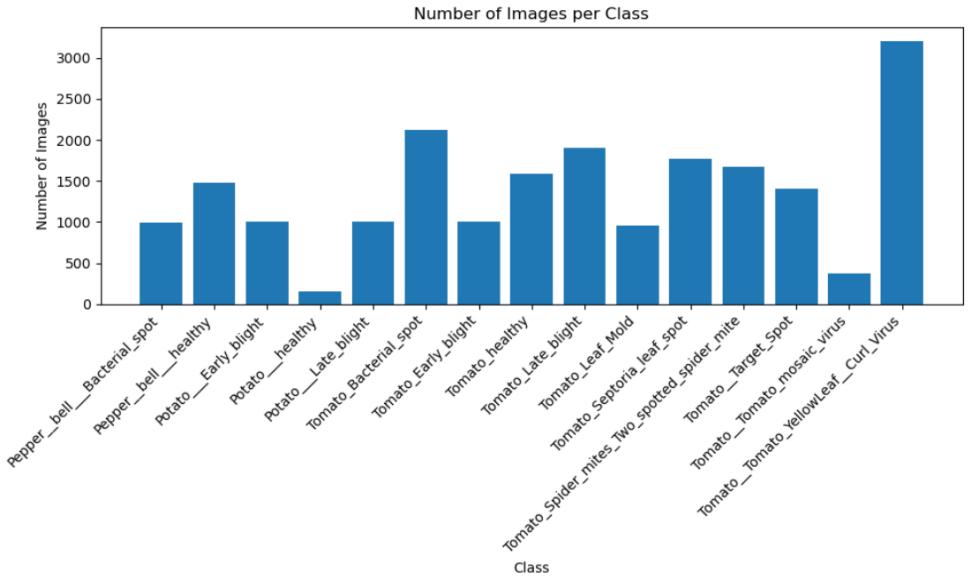
- Understand how YOLO (You Look Only Once), Mask RCNN and Convolutional Neural Networks work.
- Understand how to work with Image data
- Prepare the data according to each algorithm specification before training.
- Start training the models.

I currently finished all the above tasks that I have to complete by now. I am not ahead or behind according to the time-line, everything is aligned to the initial plan.

WORK DONE

CNN model**:**

- Imported the dataset consisting of 15 classes which belong to 3 species (Tomato, Pepper, Potato).
- The below is the visualization of all classes and the number of images in each class.



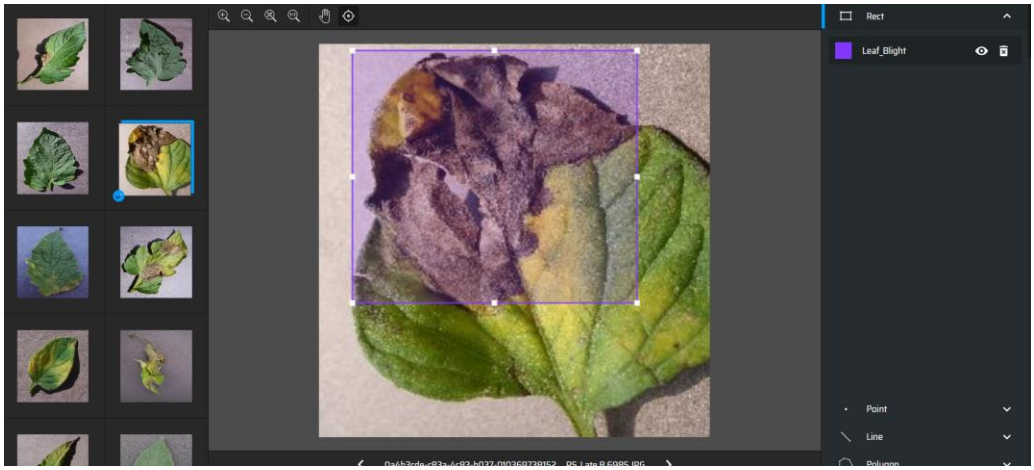
- Saved the images path and labels path in two lists separately
- Encoded the labels which are in categorical format into binary using LabelBinarizer()
- Preprocessed the images by resizing them and normalizing where are the pixel values are scaled down to 0 and 1.
- Converted the images to numpy array format before training.
- Created the CNN model using 3 convolutional layers, 3 pooling layers, 1 flatten layer, 1 dense layer and 1 output layer.
- Trained the model using batch size of 32 and for 5 epochs.
- The Test accuracy achieved was 89%

YOLO:

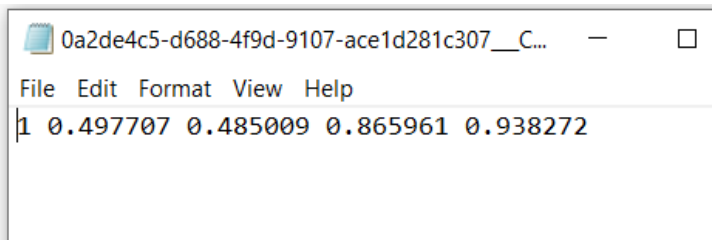
The YOLO (You Only Look Once) object detection algorithm works by dividing the input image into a grid and predicting bounding boxes and class probabilities for objects within each grid cell. It simultaneously performs object detection and classification in a single pass through the neural network. YOLO utilizes a convolutional neural network (CNN) architecture to process the input image and extract relevant features. It uses 24 convolutional layers, four max-pooling layers, and two fully connected layers. The network predicts bounding boxes by regressing the coordinates of the box relative to the grid cell and simultaneously predicts class probabilities for each box. This is achieved by utilizing anchor boxes, which are pre-defined bounding box shapes of different scales and aspect ratios.

During inference, YOLO applies non-maximum suppression to filter out overlapping bounding boxes with lower confidence scores, keeping only the most confident and accurate predictions. I am currently using the latest version which is YOLOv7 which is available here https://github.com/WongKinYiu/yolov7

I have downloaded all the necessary files from the GitHub repository and required libraries. Modified few files in the directory according to my dataset requirement. Before training the model, we have to annotate the images in YOLO format.

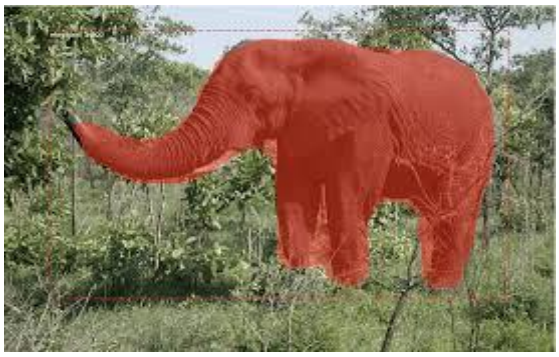


When we finish the annotation, the results will be saved in txt format for each image, saved everything in the labels folder. Each text file contains the below information, the 1 represents the class, following the four coordinates of the bounding box we created. In this way, annotated 200 images from the dataset to train the model in the first phase.

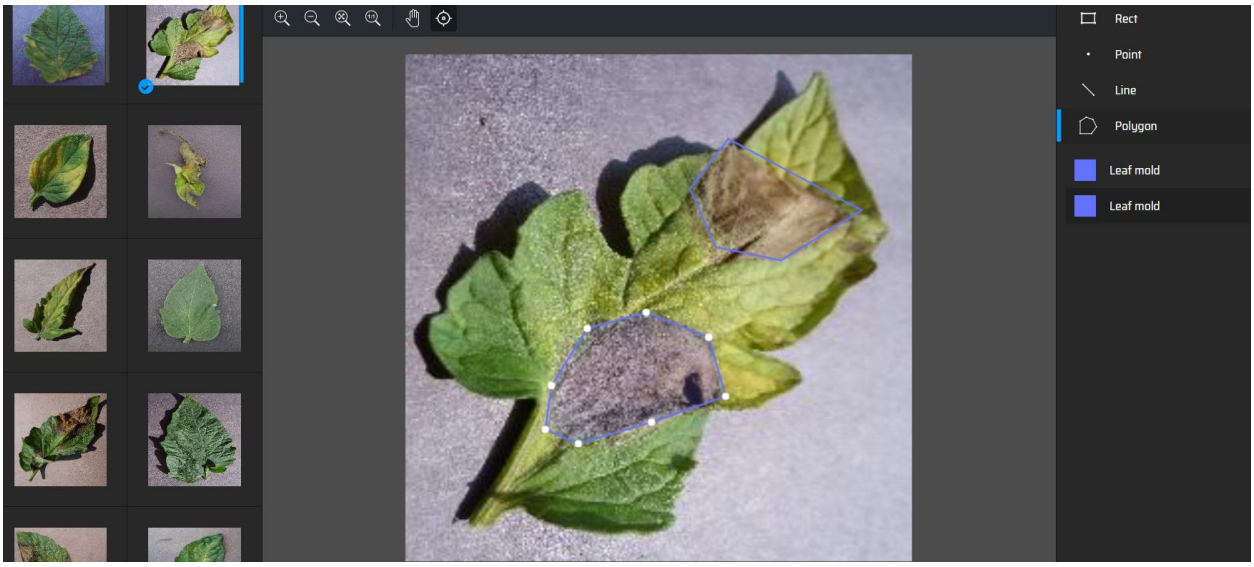


Mask – RCNN

The Mask R-CNN (Region-based Convolutional Neural Network) object detection algorithm works by extending the Faster R-CNN framework with an additional branch for pixel-level segmentation. It uses a two-stage approach, where the first stage generates region proposals. It identifies potential areas in image where objects might be using the region proposals method and the second stage classifies and refines those proposals while simultaneously predicting pixel-level masks for object instances. Once the bounding boxes are determined, it classifies the objects into different categories and creates a detailed outline or mask that shows the exact shape of the object.



In order to train this model, we need to prepare the images using polygon shape. The result of this is a json file with coordinates of each line in the image.



One json file is created for all the images and it consists of information of coordinates of each object in each image. I have annotated 200 images for the training Mask -RCNN algorithm.

```
"347b.jpg": {
        "fileref": "",
        "size": 35290,
        "filename": "347b.jpg",
        "base64_img_data": "",
        "file_attributes": {},
        "regions": {
                "0": {
                        "shape_attributes": {
                                "name": "polygon",
                                "all_points_x": [
                                        80.13640238704174,
                                        85.49506759225426,
                                        91.82803556205086,
                                        101.32748751674578,
                                        112.04481792717081,
                                        121.78784557301175,
                                        131.5308732188527,
```


Blocking issues:

While training the YOLO model, I encountered an issue where it successfully ran without any errors, but the output did not include proper classification with bounding boxes for the test images. Initially, I suspected an error with the annotation process of the training images, so I diligently re-annotated them. However, the problem persisted despite my efforts. The below is the resultant image stored in runs folder after the detection.

To further investigate, I decided to use a different dataset consisting of images of Johnny Depp as Jack Sparrow, focusing on a single class for Jack Sparrow. My intention was to test the model's ability to detect Johnny Depp's face as Jack Sparrow. After annotating a set of training images, I performed a preliminary check before training the model. Surprisingly, when I uploaded an image of Johnny Depp, the model successfully created a bounding box and identified the object as a tie within the image.

Encouraged by this result, I proceeded to train the model using my own bounding boxes, specifically targeting a person's face as the object, with the expectation of detecting Jack Sparrow. However, upon testing the trained model, it still detected the tie and recognized the face as a person, but it failed to identify it as Jack Sparrow.

Normal Issues:

Prior to training the Mask-RCNN model, I proceeded to download the required files from the official GitHub repository. However, I encountered a challenge when attempting to install the necessary libraries, as I encountered the version of python libraries that I'm working with YOLO are slightly different from Mask-RCNN.

Milestone review

Milestone name	Percentage completed	Planned start - finish	Actual start - finish
Literature review and Data collection	100%	January - March	January - March
Data preprocessing and Training models	70%	April - June	April - June
Refining the model and building an application	-	July - September	July - September
Testing and Final report	-	October - December	October - December

Issues and Risks

Issue 1:

Issue name	Failure of Object Classification with Bounding Boxes in YOLO Model Testing.
Date and time reported	April 18 2023, 5:00pm
Priority or severity of the issue	This issue is of high priority as this classification is one of the important goals of the project
Estimated time to resolve the issue	3 weeks from May 20, 2023. I will try to resolve it by June 10, 2023
Current activity to resolve the issue	-Going through the documentation from official GitHub repository of YOLOv7 to train the images. -Understanding how the directory structure and annotation effects the training.

Issue 2:

Issue name	Errors while installing the libraries for Mask-RCNN
Date and time reported	May 1 2023, 11:00am
Priority or severity of the issue	This issue is of priority two. The problem is with requirements setup.
Estimated time to resolve the issue	I will try to resolve in two weeks.

Current activity to resolve the issue	-Consider creating a virtual environment separately for each algorithm in order to avoid the conflict.

Expected Results

I am currently in the process of training the object detection models. Following this milestone, I will proceed to develop an application that enables users to upload leaf pictures, providing them with the corresponding disease name, description, and recommended solutions. This application is targeted to be completed by September 30th, marking the achievement of my third milestone.

The tasks that I'm facing issues with are Bounding box classification using YOLO algorithm and an issue with installation of necessary libraries for Mask-RCNN. I will be able to rectify these two issues by the end of June. I hope the above two issues will not affect my time line as I have some more time to work on them.

The plan of action for resolving the issue 1 is Going through the documentation from official GitHub repository of YOLOv7 to train the images and understanding how the directory structure and annotation effects the training process. The current activity to resolve issue 2 is to consider reinstalling the libraries using a virtual environment for python and using GPU from Google Colab or Kaggle to work on training the model.

Once I fix the above two issues, I will have 3 working models to detect plant leaf diseases. The CNN will be able to classify the disease of the leaf, YOLO and Mask – RCNN will create the bounding box of for the diseased leaf and classify it.