Deep Learning for Hydroponic Soybean Growth

Samuel He & Mary Hughes

FINAL REPORT

REVISION – Draft
27 April 2023

CONCEPT OF OPERATIONS
FOR
Deep Learning for Hydroponic Soybean Growth

TEAM 57

APPROVED BY:

_____
Samuel He Date

_____
Prof. Kalafatis Date

_____
Sambandh Dhal Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	12/04/2023	Mary Hughes		Draft Release
1	04/27/2023	Mary Hughes		Revision

Table of Contents

List of Tables

List of Figures

1. Executive Summary

The purpose of this project is to create a deep learning model that will be able to identify growth parameters of soybeans grown with hydroponic techniques. Particularly, the model will use feature extraction to identify the day of growth for any given image of a hydroponic soybean. In this project, we will be annotating images collected during the first 29 days of growth of hydroponic soybeans grown in a controlled lab environment. The samples were cultivated in three different concentrations of magnesium, potassium, or nitrogen. Additionally, there were samples in Plasma Activated Water, and the control samples were in distilled water alone. The annotated images of the samples over this period will be used to train, test, and validate the deep learning model. This will be done by inputting each image, its corresponding dataset being labeled in the annotation, so that the model learns the features of the soybeans that coincide with each day of the growth cycle. In addition to the deep learning model, there shall be a user interface produced over the duration of this project. It shall allow a user to input one or more images and shall display the prediction that was output by the model, along with the calculated accuracy of the prediction. There will be a frontend of the user interface and a backend that are connect to each other remotely. The frontend shall be able to receive user input images and pass them to the backend. Further, the backend shall have the necessary API calls to pass the user images to the deep learning model and return the model's predictions to the user on the front end.

2. Introduction

The purpose of this document is to present Deep Learning for Hydroponic Soybean Growth, a trained software for identifying stages of growth of soybeans in hydroponics. This project has the potential to help soybean growers worldwide and could even be expanded for further stages of soybean growth, soybeans not grown hydroponically, and even to other plants.

2.1. Background

Agriculture is a primary industry in our world today, and it has been for centuries. As food is necessary for sustaining life, we've been reliant on agricultural practices since the beginning of humanity. As population increases, and therefore demand for food increases, the agriculture industry has needed to adapt to be more efficient over time. This has been possible by using new technologies modified for the applications necessary. Not to forget, availability of farmland also continues to shrink for various reasons, furthering the necessity of cutting-edge engineering for the purpose of efficiency and quality in production.

AI, machine learning, and deep learning are employed primarily to process large amounts of data quickly and efficiently. They are able to process a great deal more than any individual or group of people could, and significantly faster as well. Deep learning is a neural network that attempts to simulate the human brain in how it processes information. The models "learn" by taking in significant amounts of data and examples. Typically, the more examples that are used to train the model, the more precise it will be. Deep learning can process more unstructured data, such as images, automating feature extraction. The trained models can identify distinctive features in an image to categorize what is shown. An example that an IBM article gives is sorting images of pets into "'cat', 'dog', 'hamster', et cetera" [IBM]. The algorithm may identify the ears, the nose, or the tail to determine which category each image belongs to.

Although AI is suitable for use in many fields, we would say it is essential in those that are necessary for human life. These are the sectors of healthcare, education, and the food industry as a whole. In healthcare, there are intangible amounts of data. Doctors are well-trained, educated beings, but even they make mistakes, or miss something in a patient's history that could help identify the cause of a current health issue. Many times, there is no room for error. The efficiency and accuracy of deep learning AI models can help with processing patient information to correctly determine the problem, causes, and recommend solutions. Similarly, farmers have a lot of data at their fingertips daily, and deep learning models can aid in processing this data so it can be used to improve various aspects of production.

2.2. Overview

The project will consist of a trained deep learning model and a user interface that will interact with the trained model. The user interface will ideally be a website in which a user can input

multiple images at a time for the model to process and output the specified dataset to the user interface. Below is the block diagram for the overall system.

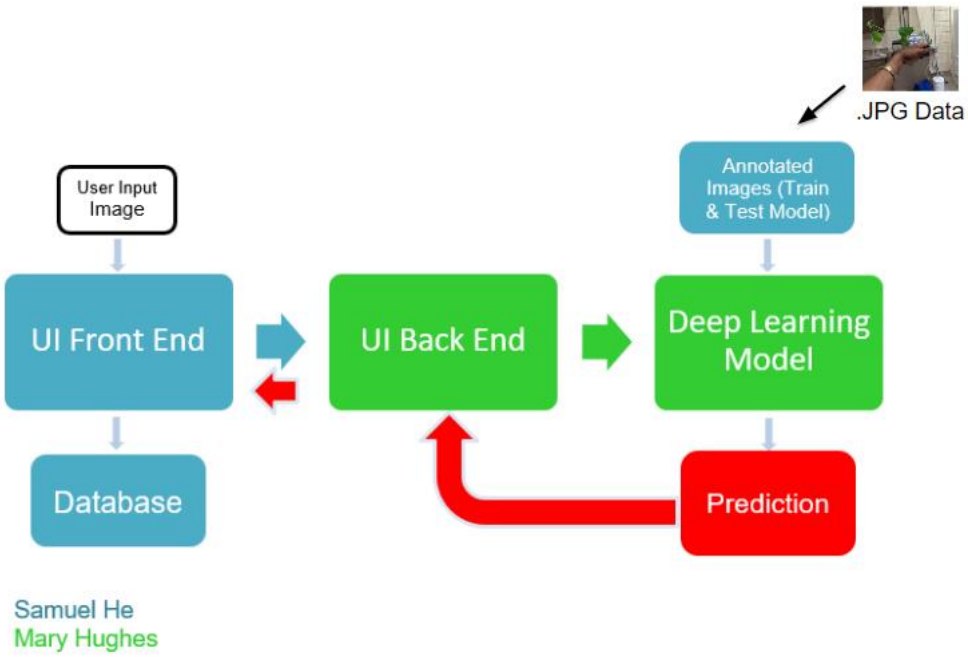

Figure 1: Conceptual System Block Diagram

2.3. Referenced Documents and Standards

[1] *Hydroponic Systems*. (2017, January 26). University of Massachusetts Amherst – Center for Agriculture, Food, and the Environment. *Hydroponic Systems.* Retrieved September 9, 2022 from https://ag.umass.edu/greenhouse-floriculture/fact-sheets/hydroponic-systems

[2] *Benefits of Hydroponics: The Future of Farming.* (2022, February 11). Green Our Planet. Retrieved September 9, 2022 from https://greenourplanet.org/hydroponics/benefits-of-hydroponics/.

[3] IBM Cloud Education. (2020, May 1). IBM. *What Is Deep Learning?* Retrieved September 12, 2022 from https://www.ibm.com/cloud/learn/deep-learning.

[4] Liu, Simon Y. "Artificial Intelligence (AI) in Agriculture." *IT Professional*, vol. 22, no. 3, 2020, pp. 14–15., https://doi.org/10.1109/mitp.2020.2986121.

3. Operating Concept

3.1. Scope

The Deep learning software for hydroponic soybean growth will be utilized within any situation requiring the user to identify lab-grown soybeans. The model will be trained with image data provided by the agricultural department, and thus will be used in similar situations in which the model identifies the stage of growth. For scale purposes, this will only be used with soybeans grown hydroponically in labs, and not any soybean plant in any condition.

3.2. Operational Description and Constraints

Our project will be used to identify the day of growth of a hydroponic soybean plant. To achieve this, our deep learning model will take an image as input and use its knowledge to identify growth parameters. It will then output the information specified above. Some constraints must be taken into consideration:

- There may be overlap between plants, so the model must be able to handle those situations.
- The model must be able to handle different lighting conditions.
- The model must take into consideration that the soybeans are grown in distilled water alone from days 1-8, and a nutrient solution from days 9-28.

3.3. System Description

Deep Learning for Hydroponic Soybean Growth is separated into three phases: image processing, model training, and user interface development. Both software-based phases are described below.

Image Processing: In the image processing phase, first we will take the images collected from the growth process of the plants and annotate them. One of the original images is *Figure 2.* This is to collect essential information about the plants and identify this manually, giving each image a label that is consistent throughout the set of images. Then, these labeled images will be fed into a computer program that will output a dataset corresponding to each given image. These dataset outputs are used as inputs for the model training phase.

Model Training: In the model training phase, we will be inputting images and the datasets associated with each image. Two of the images that will be used to train our deep learning model are shown in *Figure 5* and *Figure 16*.

User Interface Development: For this subsystem, we will be creating a website for the usage of the deep learning model we have trained.

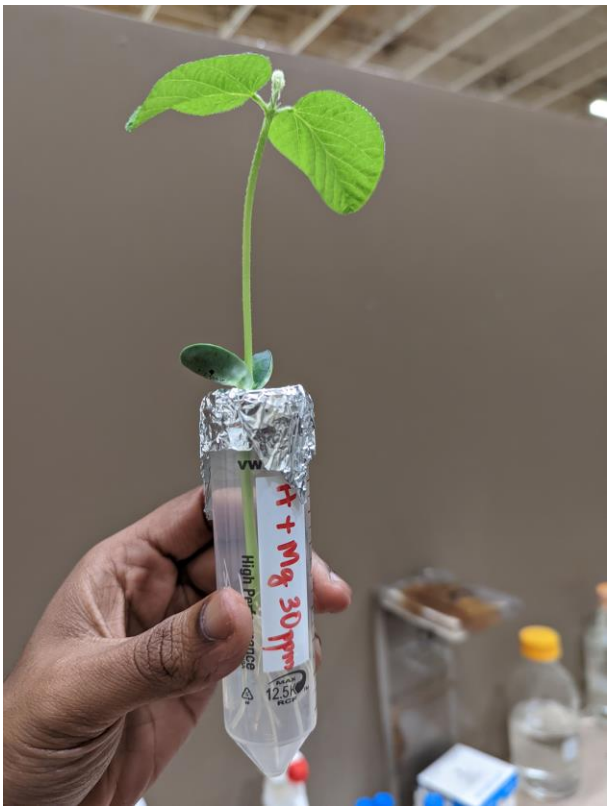

Figure 2: Image of Soybean with Magnesium Concentration of 30ppm on Day 15

3.4. Modes of Operations

The Soybean Hydroponics system will have two primary modes of operation in which it will be used. As the system is simply software to be used within a larger scoped project, the two main modes of operation are:

Active
The active mode of operation refers to when the user inputs an image into the software. The user is given the ability to input an image as the input, and receive the output day of growth, as well as the output hydroponic solution for growth.

Idle
Idle refers to the mode of operation in which the user is not inputting an image into the machine learning model, and the system is not being used.

3.5. Users

The proposed system will be utilized as a subsystem for a much larger system design. Potential users include researchers within the agricultural department within and outside of Texas A&M University. As these groups of people both deal with tracking soybean growth within different environments, our software will automate the process, and in turn, reduce the execution time of research tasks. As the software will input an image and output a day of growth and a solution of growth. Some software literacy is required to use the system, so the researchers must have some knowledge of using it.

3.6. Support

After creating this product, we plan to provide an instruction/user manual on how to use the software in the system, as well as documentation behind the code itself, and how the deep learning model works. There will be instructions on how to operate the user interface and guidelines for the type of input the model can accept and process.

Additionally, the deep learning model portion of this project was created, and the code written, in such a manner that if supplementary data is collected, the model can be retrained on the larger dataset to increase the performance of the model itself. Details for this are include in Section 21.4.

4. Scenario(s)

Since our project will not produce an end-user product, but is an intermediate stage of a larger project, there are limited uses for what our team will produce by the end of this course. Eventually, our trained model could be used to create an end-user product that may be sold or otherwise marketed for uses that will not be listed here. At this time, we do not know what the end goal of this overarching project is; therefore, in this section, we can only describe how the system can be used separate from any additional systems, applications, or data.

4.1. Agricultural Research Use

Researchers in agriculture departments would be able to use our model to expand the depth of their research.

4.2. Engineering Research Use

Research in engineering fields shall be able to use the model as a basis to build more complex models, or to widen the capabilities of this one.

4.3. Use by Farmers and Others in Industry

In farming/growing, it can be useful to offset planting dates for a variety of reasons, and this model could in the process to ensure that all plants are grown to maturity.

4.4. Plant Rehabilitation Efforts

For those who rehab plants, this tool could be useful in identifying the age of the plant, so they can determine the care necessary moving forward with their rehabilitation efforts.

5. Analysis

5.1. Summary of Proposed Improvements

- The system will automate the workflow of identifying the day of growth of a soybean plant.
- The software utilizes open-source deep learning libraries, and therefore is easily modifiable and can be re-trained.
    - If additional data were to become available that is equivalent in format to the original images, details on how to incorporate this data, and re-train the model accordingly, are included in Section 3.6 of this paper.
- This system provides a solution to re-label subjects that have lost any or all methods of identification.

5.2. Disadvantages and Limitations

Because our project is specifically based on hydroponically grown soybeans, the final model will only be trained to this, and therefore is limited by it. The model would not necessarily be able to be employed for soybeans grown conventionally or otherwise and will not support any other plant. Not only this, but as our model will be trained by data specifically with a lab background, even if the plants were grown hydroponically, if the image had a

different background, it would be difficult for the model to accurately classify the image. The image quality also determines the accuracy of the classification.

5.3. Alternatives

There are a few alternatives to our proposed solution. The first alternative consists of creating a manual workflow to label images based off recognizable plant attributes. There are many issues with this approach. The first is the copious amounts of time that it will take to recognize and label the images. As there can be any amount of or types of data to be processed, this method will result in uncertainty in the recognition of the stage of growth for the dataset, as the individual would need to label each plant individually, in comparison to our solution, which would label all the plants in the image at once.

Another alternative solution to utilizing deep learning for recognizing hydroponically grown soybeans at different stages in time with different solutions would be to hard code parameters in the image recognition process with specific image processing techniques. This alternative is not nearly as effective as utilizing deep learning techniques, specifically image classification, because there are many variables that exist within the dataset, and thus there would have to be a disproportionate number of parameters to satisfy those variables.

5.4. Impact

In terms of the environment, sustainable and ‘green’ solutions are very important these days. Hydroponics itself is a very efficient and eco-friendly way to grow a variety of plants. As opposed to conventional, soil-grown plants, hydroponics allows crops to be grown in places that do not have proper environments or soil suitable for growth. In addition, hydroponics reduces water consumption, and many growth systems recycle water, which is a plus on the sustainable side of things. Another substantial advantage of hydroponics that is ecologically sound, is the space component. There is limited land on this planet, and the volume of land available for farming is only shrinking over time. Hydroponics offers the capability to expand vertically, like the adaptation of New York City and all its skyscrapers: there was no more room to grow outwardly, so they just went upwards instead.

Deep Learning for Hydroponic Soybean Growth

Samuel He & Mary Hughes

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – 2
27 April 2023

Change Record

Rev.	Date	Originator	Approvals	Description
-	09/21/2022	Mary Hughes		Draft Release
1	12/04/2022	Mary Hughes		Revision
2	04/27/2023	Mary Hughes		Revision

6. Introduction

6.1. Purpose and Scope

Deep Learning for Hydroponic Soybean Growth is a deep learning model trained for use with hydroponic soybeans to detect the day of growth and the nutrient solution its being grown in. Figure 3 shows the system block diagram for our project.

6.2. Responsibility and Change Authority

The team leader, Samuel He, will be responsible for ensuring that both subsystems meet specifications. Any changes made to the project must be cleared by Samuel He and the project sponsor, Sambandh Dahl.

7. Applicable and Reference Documents

7.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
IEEE 2941-2021	3/18/2022	IEEE Standard for Artificial Intelligence (AI) Model Representation, Compression, Distribution, and Management
IEEE 3652.1-2020	3/19/2021	IEEE Guide for Architectural Framework and Application of Federated Machine Learning
IEEE P3123	11/9/2021	Standard for Artificial Intelligence and Machine Learning (AI/ML) Terminology and Data Formats
IEEE P2841	9/21/2022	IEEE Approved Draft Framework and Process for Deep Learning Evaluation

Table 1: Applicable Documents

7.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
IEEE P3123	11/9/2021	Standard for Artificial Intelligence and Machine Learning (AI/ML) Terminology and Data Formats

Table 2: Reference Documents

7.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions

All specifications, standards, exhibits, drawings, or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

8. Requirements

This section defines the minimum requirements that the development item(s) must meet. The requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system, are covered.

8.1. System Definition

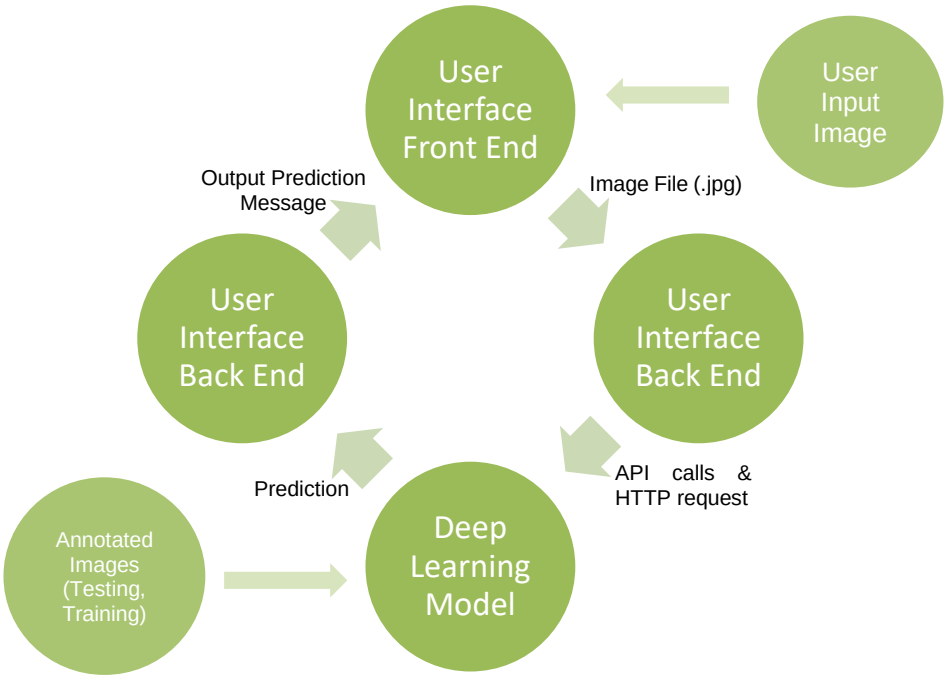

Figure 3: System Block Diagram

8.1.1. User Interface Front End

The User Interface Front End shall be built to take in .jpg images as the input and output the message response given by the backend. It shall restrict the user from inputting any other file type and send an error flag when it does not receive the right file type.

8.1.2. User Interface Back End

The User Interface Back End shall be built using the Python Flask framework in tandem with TensorFlow. It shall take inputs as an API via HTTP requests and restrict access to the API to just the web application's front end, while excluding all other requests. It shall return both the day of growth and the accuracy of a prediction by sending the input data to the Deep Learning Model.

8.1.3. Deep Learning Model

The Deep Learning Model consists of a Convolutional Neural Network created using the Python Library known as TensorFlow. The model will take raw image input data and output a list of percentage probabilities of the image being associated with each category (day of growth).

8.1.4. Image Processing & Annotations

The Image Processing and Annotations shall be a process in which images will be segmented using an Annotation tool and converted into .jpg files with the backgrounds removed.

8.1.5. Database

The Database shall keep record of image locations, as well as send the data to an S3 bucket. This database shall keep track of any solution, water uptake, and dry weight data as well. It shall support select and insert queries.

8.1.6. Data Analysis

Data analysis shall be performed on numerical data provided by the sponsor, which shows water uptake, nutrient solution contents, and biomass information for samples throughout the growth process. This analyzed data will be displayed on the UI, with knowledge gathered during the analysis that may be helpful to users. Also done with the data analysis was the creation of a heat map for each of the three main nutrient solutions, which were Magnesium, Potassium and Nitrogen. The Potassium heat map is shown below. All heat maps for this project show the correlation between all features from the nutrient solution numerical data provided by our sponsor, where the growth indicators are pH, conductivity, alkalinity, hardness, and TDS. Within the UI backend, the heat map is updated and displayed for the day of growth a user requests, and this updated heat map is displayed for the user to access.

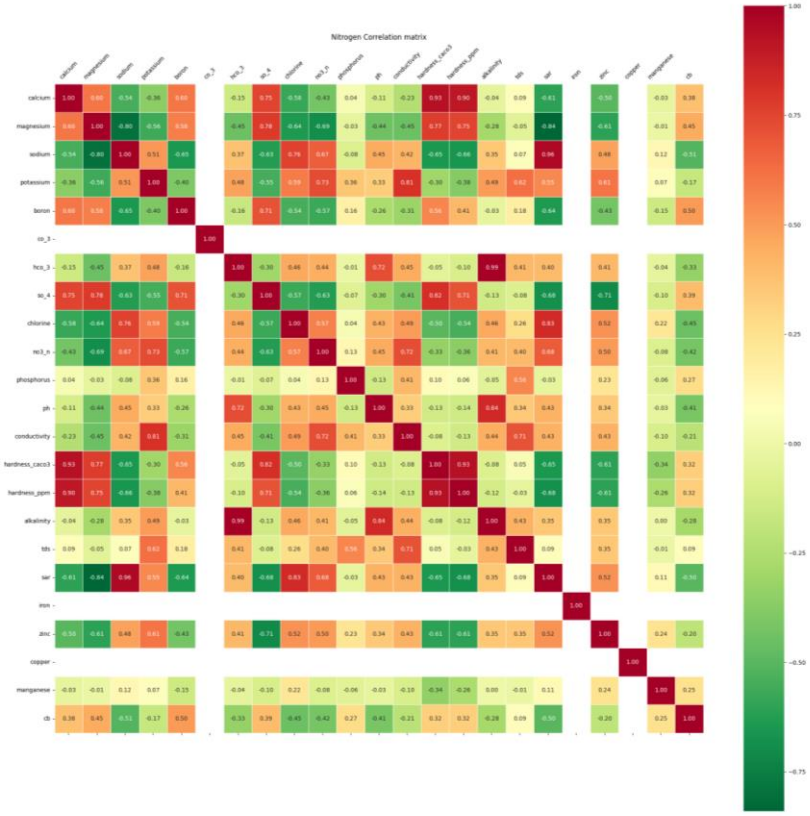

Figure 4: Potassium Heat Map Available on UI

8.2. Characteristics

8.2.1. Functional / Performance Requirements

8.2.1.1. Day of Growth Identification

The SHDLM shall be able to make a prediction of the day of growth of a soybean from an input image based on its training.

Rationale: This is the 1st of two core system performance requirements.

8.2.1.2. UI Input Acceptance

The SHDLM shall have the ability to accept a user upload in the form of a JPEG image. The SHDLM shall have the ability to accept user inputs for solution, image, water uptake, and dry weight.

Rationale: This is a fundamental operation of the user interface and communication with the model.

8.2.1.3. UI Output Delivery

The SHDLM shall deliver an output to the UI in a readable manner after an input is processed by the deep learning model.

Rationale: An intuitive and readable UI is necessary for the deep learning model output to be effectively interpreted by the user.

8.2.1.4. Database

The SHDLM shall store all predictions and images on the database and file storage. The database shall have unique tables for each type of data, as well as links to the file storage within the table entries.

Rationale: A database is required to record all data that the user wants to save for later access.

8.2.2. Physical Characteristics

N/A as project is entirely software.

8.2.3. Electrical Characteristics

8.2.3.1. Inputs

a. While there are no electrical inputs, this SHDLM does require a variety of inputs for the software being used to create the deep learning model itself, and for the software used to deploy the model and the UI.

b. It follows that the use of the model also demands inputs to serve its purpose. Through the UI, anyone using the model must have inputs to give the model.

Rationale: By design, SHDLM will require input to use the trained model.

8.2.3.1.1 Deep Learning Model Training

The SHDLM shall be trained using annotated images. These annotations will be done on the original images provided by the project sponsor using CVAT. As CVAT does not export .jpg annotated images, the annotated images are exported in the COCO data format, and that data must be then processed into raw .jpg images through python script processing, creating images showing only the plant and black otherwise, as shown in *Figure 5.*

8.2.3.1.2 Inputs to Heroku for Back End

The SHDLM shall have Cloud Dependencies for the Project to run, which will configure the Heroku Stack in Heroku Terminal to include Ubuntu 20.04 LTS, Python Version (3.8.6), and have all the necessary Python Libraries will be defined in a requirements.txt to include all necessary dependencies that Heroku must install on the cloud server for the project. There will also be a Procfile telling the cloud server to run specific commands before running the API.

8.2.3.1.3 Input to Heroku for Front End

The SHDLM shall have Cloud Dependencies for the Project to run, which will configure the NodeJS dependencies, which will be defined in a package.json to include all necessary dependencies that Heroku must install on the cloud server for front-end web application.

8.2.3.1.4 External Commands

The SHDLM shall document all external commands in the appropriate ICD.

Rationale: The ICD will capture all interface details from the low-level subsystem to the high-level usage format.

8.2.3.2. Outputs

8.2.3.2.1 Data Output

The SHDLM shall include an interface compatible with the deep learning model that outputs the prediction the model produces.

Rationale: The UI will pass information directly to the deep learning model and receive outputs from output directly from the model to the interface.

8.2.3.2.2 Diagnostic Output

The SHDLM shall not include a diagnostic interface.

8.2.3.2.3 Raw Video Output

The SHDLM shall not include a raw video output component.

8.2.3.3. Connectors

N/A as project is entirely software.

8.2.3.4. Wiring

N/A as project is entirely software.

8.2.4. Environmental Requirements

There are no environmental requirements necessary.

8.2.5. Failure Propagation

The SHDLM shall not allow failures beyond the failure detection systems implemented.

8.2.5.1. Failure Detection

8.2.5.1.1 Application Failure Detection (AFD)

The SHDLM shall have an internal subsystem that will generate test queries from the front-end application to the back-end application to test the functionality of communication between the two.

Rationale: As the application is hosted on a third-party web-hosting service (such as AWS or Heroku), the reliability of the web application is heavily influenced by the reliability of the web-hosting service.

8.2.5.1.2 Model Failure Detection (MFD)

The web-application should have a method of detecting whether the model returns a valid output or not.

Rationale: If the model outputs some sort of invalid output that is not useful to the user, then the web-application should either communicate the lack of output to the user or re-run the model on the input until a valid output is given.

Figure 5: Annotated Soybean Image Used for Model Training

9. Support Requirements

As the web application will be hosted on a third-party web-hosting service, extensive information on how to contact the help desk for the application will be provided within our technical support documentation. However, as our clients are well-informed on the process of creating/synthesizing data and training the model, there is a lack of need for technical support for that portion.

Deep Learning for Hydroponic Soybean Growth

Samuel He & Mary Hughes

INTERFACE CONTROL DOCUMENT

REVISION – 2
27 April 2023

Change Record

Rev.	Date	Originator	Approvals	Description
-	09/30/2022	Mary Hughes		Draft Release
1	12/04/2022	Mary Hughes		Revision 1
2	4/27/2023	Mary Hughes		Revision 2

10. Overview

The Deep learning software for hydroponic soybean growth will be utilized within any situation requiring the user to identify lab-grown soybeans. The model will be trained with image data provided by the agricultural department, and thus will be used in similar situations in which the model identifies the stage of growth and solution in which the soybean is grown. For scale purposes, this will only be used with soybeans grown hydroponically in labs, and not any soybean plant in any condition.

11. References and Definitions

11.1. References

IEEE P3123
Standard for Artificial Intelligence and Machine Learning (AI/ML) Terminology and Data Formats
9 Nov 2021

11.2. Definitions

See Appendix A for Acronyms and Abbreviations, and Appendix B for Definitions used in this document.

12. Physical Interface

12.1. Weight

N/A as project is entirely software.

12.2. Dimensions

N/A as project is entirely software.

12.3. Mounting Locations

N/A as project is entirely software.

13. Thermal Interface

N/A as project is entirely software.

14. Electrical Interface

There will only be one aspect of Electrical Interface, which is a user interface. This is described below in the corresponding section.

14.1. Primary Input Power

N/A as project is entirely software.

14.2. Polarity Reversal

N/A as project is entirely software.

14.3. Signal Interfaces

N/A as project is entirely software.

14.4. Video Interfaces

N/A as project is entirely software.

14.5. User Control Interface

The user control interface, also referred to as the User Interface in other sections of this document, will allow the user to upload one or more images and will return a prediction for each image from the deep-learning model. The interface will then give the user some sort of feedback on both the processing of the images, as well as the outputs of the images once the model is finished processing them. This data will then be stored onto the database per the user's request, and analyzed using a variety of data analysis techniques via the backend API.

The front end of the user control interface will communicate with the backend, which contains the API calls necessary to interact with the deep learning model, and to return the model's prediction back to the frontend of the user in a digestible format. It contains four main categories – water uptake, solution data, image data, and dry weight. The data from the backend is displayed on data tables, and the analysis can also be queried for per the user's request.

The back end of the user control interface will send the data the user sends to it and store it in a database. The database data is also accessible via API calls to the backend, and analysis on the data can be done via API calls as well. All of this is accessible in the various tabs on the UI.

15. Communications / Device Interface Protocols

15.1. Wireless Communications (WiFi)

N/A as project is entirely software.

15.2. Host Device

The UI will require a device to run the website. The UI will have autoscaling to allow for a variety of host devices that can properly run the website.

15.3. Video Interface

N/A as project is entirely software.

15.4. Device Peripheral Interface

N/A as project is entirely software.

Deep Learning for Hydroponic Soybean Growth

Samuel He & Mary Hughes

SUBSYSTEM REPORTS

REVISION – 1
27 April 2023

Change Record

Rev.	Date	Originator	Approvals	Description
-	12/04/2022	Mary Hughes		Draft Release
1	04/27/2023	Mary Hughes		Revision

16. Execution Plan

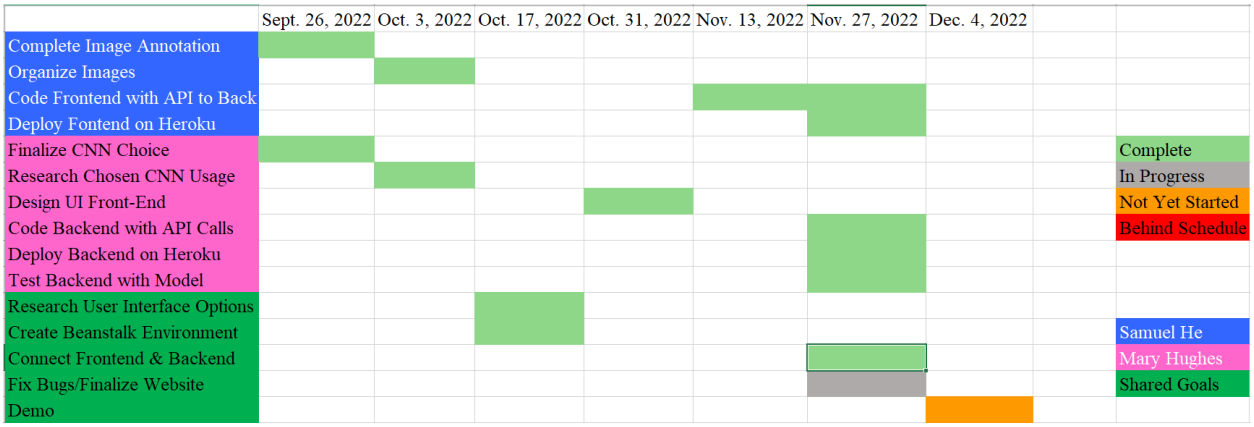

Figure 6: Execution Plan for Semester 1

Everything included in the First Semester's Execution Plan has been completed as of December 4, 2022.

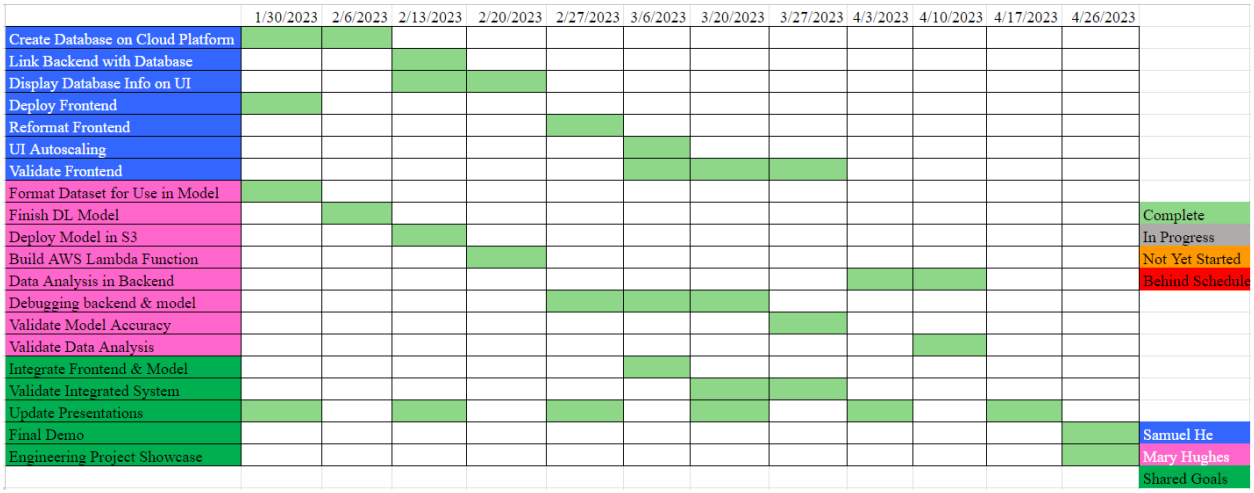

Figure 7: Execution Plan for Semester 2

Everything included in the Second Semester's Execution Plan has been completed as of April 26, 2023.

17. Validation Plan

Paragraph #	Test Name	Success Criteria	Status	Responsible Engineers
3.2.1.3	UI Front End Functionality	The User Interface works as expected, users can upload an image, webpage interacts as intended.	COMPLETE	Samuel He
3.2.1.3	Input Delivery to Back End	Input is successfully being delivered to the backend from the front end of the UI.	COMPLETE	Samuel He
3.2.1.3	UI Backend Communication with Model	The User Interface Back End API calls work as expected, and can return a prediction in a 3rd party testing platform.	COMPLETE	Mary Hughes
3.2.3.2.1	UI Output Delivery	An output is being delivered in the correct format to the UI.	COMPLETE	Mary Hughes
3.2.1.3	UI Readability	UI design is clean and understandable, easy to use.	COMPLETE	Shared
3.2.1.1	Day of Growth Identification (404)	The deep learning model is correctly identifying the day of growth of an input.	UNTESTED	Mary Hughes
3.2.1.2	Nutrient Solution Detection (404)	The deep learning model is correctly identifying the nutrient solution of an input.	UNTESTED	Mary Hughes
3.2.5.1.1	Application Failure Detection (404)	Internal testing properly identifies when the application fails to communicate with the deep learning model.	UNTESTED	Samuel He
3.2.5.1.1.1	Model Failure Detection (404)	Application correctly detects if the model has given a valid input to the UI.	UNTESTED	Samuel He
N/A	Full System Demo	The application and deep learning model process input as expected and deliver correct output to the UI.	UNTESTED	Shared

Figure 8: Validation Plan for Semester 1

All Validation expected to be done during the First Semester has been completed as of December 4, 2022.

Paragraph #	Test Name	Success Criteria	Methodology	Status	Responsible Engineers
8.2.1.2	UI Image Input	Users can upload up to 10 different images of any size with a 10 second maximum response time.	Upload 10 different image sets to the User interface, each with varying sizes and quantities ranging from 1-10 photos and 1-10 MB and time the response times.	PASSED	Samuel He
8.1.2	Webpage Autoscaling	Webpage autoscales properly to mobile and desktop screens	Test the mobile view of the website on at least 10 different mobile views, using React Native Layout Tester. Compare results.	PASSED	Samuel He
8.1.2	Webpage Interactivity	Webpage navigation interactions are functional	Test button pressing functionalities of each button on navigation.	PASSED	Samuel He
8.1.5	Database Outputs on Frontend	Webpage frontend has all database prediction information displayed properly	Upload 50 images and monitor predictions for them both individually and altogether. Input images into model directly. Compare results.	PASSED	Samuel He
8.2.1.2	Input Delivery to Back End	Image is successfully being delivered to the backend from the front end of the UI in <1sec	Monitor database to see if corresponding images and predictions are sent out and received. Send out time and retrieval time will be monitored by test cases in React.	PASSED	Samuel He
8.2.5.1.1	Application Failure Detection	Internal testing properly identifies when the application fails to communicate with the deep learning model.	Restrict access from the application to the model. Attempt to upload an image to the model.	PASSED	Samuel He
8.2.5.1.1	Application Failure Response	Webpage gives user a correct error message when incorrect image formats are uploaded	Upload a set of 15 different files that are not .jpg or .jpeg.	PASSED	Samuel He
8.2.5.1.2	Model Failure Detection	Application correctly detects if the model has given a valid input to the UI.	Create an invalid prediction response on the backend, and attempt to upload an image to the model.	PASSED	Samuel He
8.2.1.1	Day of Growth Identification	The deep learning model is correctly identifying the day of growth of an input.	Create 264 test cases with corresponding images that cover all of the different categories. Compare results with pre-determined day of growth inputs.	FAILED	Mary Hughes
8.2.1.2	UI Delivers Input to backend with API Calls	User Input images are successfully delivered to backend Flask app using the APIs.	Test the API using POSTMAN. Verify with VS Code terminal that the API was used.	PASSED	Mary Hughes
8.2.3.2.1	UI Output Delivery	An output is being delivered to the UI in the correct format, including the prediction and the accuracy of prediction.	Upload 20 different images to POSTMAN, one from each day of the growth cycle represented in the dataset, and verify the output shown in the POSTMAN console is the correct format for the UI to receive and interpret properly.	PASSED	Mary Hughes
8.2.1.5	Data Analysis	Data analysis is carried out and can be found on the tabs in the UI.	Analysis in Colab is completed without error, and final data files are saved for use on UI. Visual inspection is done to validate correctness of linear interpolation.	PASSED	
N/A	Full System Demo	The application and deep learning model process input as expected and deliver correct output to the UI.	Upload a set of 20 images, and compare their individual predictions with the model to the UI output.	PASSED	Shared
8.2.1.2 and 8.2.1.3	UI Backend Communication with Model	The User Interface Back End API calls work as expected, and can return a prediction in a 3rd party testing platform.	First, validate the Frontend communcation with the Backend. Secondly, send 30 images to the model using both POSTMAN and the UI. Compare response results.	PASSED	Shared
8.1.2	UI Readability	UI design is clean and understandable, easy to use on multiple brightness levels	Compare readability on at least 5 different monitor display/brightness settings.	PASSED	Shared

Figure 9: Validation Plan for Semester 2

All Validation expected to be done during the Second Semester has been completed as of April 26, 2023.

18. UI Front End Subsystem

18.1. Subsystem Introduction

The UI Front End Subsystem is designed for the user to easily input images via either file upload or taking a photo. The subsystem displays the name of the uploaded file, as well as the output of the prediction. The user interface was tested to confirm the response times of the API to the front-end website, as well as the response of the system to different file types.

18.2. Subsystem Details

The file upload portion of interface is shown below:

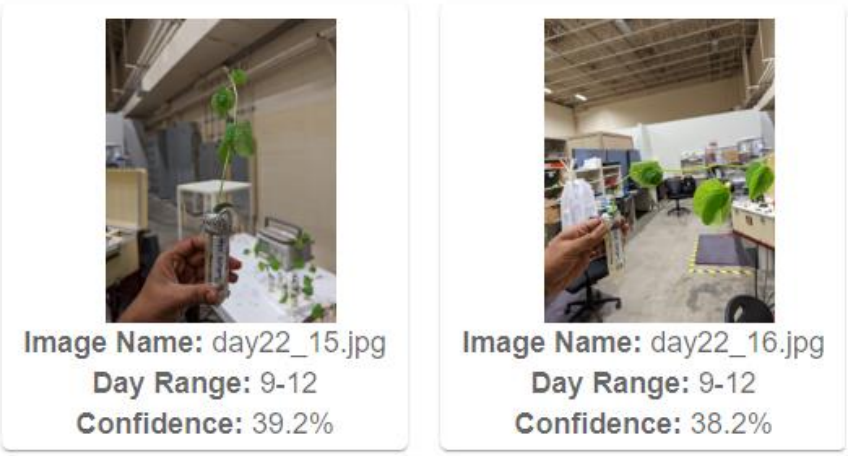

Figure 10: User Interface File Upload and Output

The primary challenge with the Front-end subsystem was the multiple file uploads and having to interact with multiple APIs in the backend. Because of the RAM restrictions on the backend prediction API, the API could only handle single file uploads. Thus, the front end had to simulate a multiple file upload by uploading the files individually in a queue-esque format. As each fetch request was unique, there was also an ID associated with each request, which was also associated with cards that were displayed on the frontend, as shown in *Figure 10*.

To ensure the id was retained, the id was stored on a data structure that was mapped to a generic card format, with each card data structure also holding an image object, and empty prediction and confidence fields. This field would be updated when the UI sends a prediction to the prediction API, which would then return a prediction and confidence for the prediction, as well as the associated id with the request. This id would then be used to display the data on the associated card on the user interface, which would have an identical id.

18.3. Subsystem Validation

The UI front end was tested for the speed of response of the front end based off the image size. This timing was done by taking the timing of the loading symbol on the front end and subtracting the time the back end spent on processing that image. We began with 128x128 pixel images, and then progressed to 1024x1024 images, increasing image size by adding 128 pixels to each dimension of the image. As shown in the figure below, the response time is relatively inconsistent, however there is a slight increase in time spent loading based on image size.

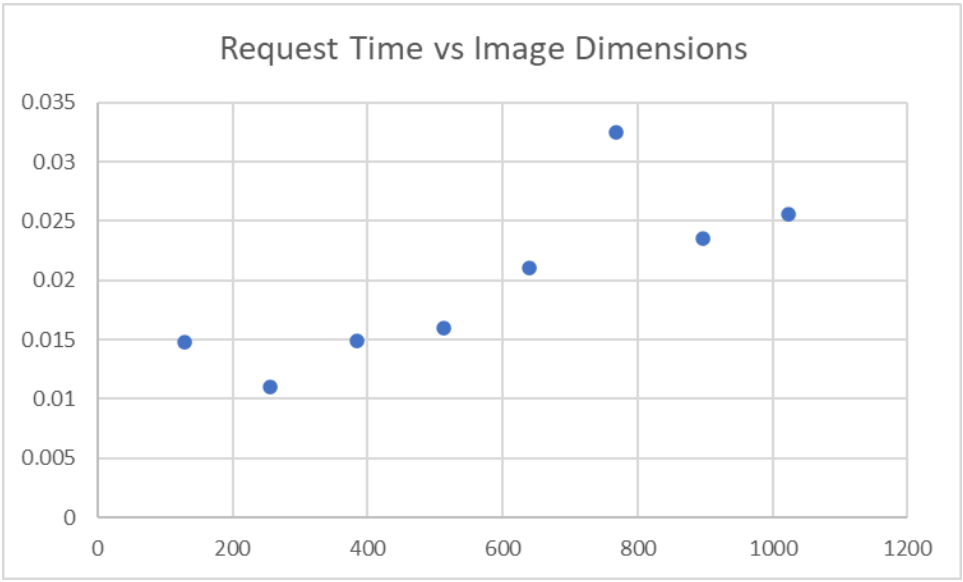

Figure 11: Front End Request and Load Time Based on Input File Size

19. UI Back End Subsystem

19.1. Subsystem Introduction

The back end of the UI will handle API calls, HTTP requests made on the web application, and will call the model via an H5 file. The main portion of the code written for the back end will employ Flask, a framework for Python. In addition to handling inputs and sending them to the SHDLM for predictions, the back end also receives the output from the model and generates a digestible format of this output to serve to the user on the front end. The back end also queries a PostgreSQL database that is linked to an Amazon S3 bucket that stores the image data.

19.2. Subsystem Details

The back end of the UI has the infrastructure to handle GET and POST requests made within the web app. The backend is made up of two different APIs, with one in charge of database queries and data analysis, and one in charge of model predictions. The Database API is connected to a PostgreSQL database, as well as an Amazon S3 bucket file storage system. The database tables store multiple types of data, such as solution, water uptake, and dry weight. Each of these tables have unique columns, with the image data table also storing a URL to the S3 bucket image.

```
soy-api2::DATABASE=> select * from image_data;
                  id                  |  image_name   | day_prediction |                       image_url                        
      |                    segmented_image_url                     |      accuracy
--------------------------------------+---------------+----------------+--------------------------------------------------------+---------------------
-------------------------------------+---------------------
 ebccc613-2d79-453c-b753-c93fea5afdcb | day22_16.jpg  | 9-12           | https://soy-api-s3.s3.us-east-2.amazonaws.com/day22_16.jpg  | https://soy-api-s3.s3
.us-east-2.amazonaws.com/day22_16.jpg  | 0.38348686695098877
 25190126-8643-42a5-80f0-3cb10f909d06 | Day_14-22.jpg | 9-12           | https://soy-api-s3.s3.us-east-2.amazonaws.com/Day_14-22.jpg | https://soy-api-s3.s3
.us-east-2.amazonaws.com/Day_14-22.jpg | 0.3966885507106781
 6f765988-a7f0-41f1-8b55-770498bfc51a | Day_11-25.jpg | 9-12           | https://soy-api-s3.s3.us-east-2.amazonaws.com/Day_11-25.jpg | https://soy-api-s3.s3
.us-east-2.amazonaws.com/Day_11-25.jpg | 0.39824050664901733
 0e8017da-97f1-4f20-b2f4-8a77cb861e3f | image.jpg     | 9-12           | https://soy-api-s3.s3.us-east-2.amazonaws.com/image.jpg     | https://soy-api-s3.s3
.us-east-2.amazonaws.com/image.jpg     | 0.38770443201065063
 35263d3e-60ba-405e-af4f-9c2f3bbe2728 | day20_15.jpg  | 9-12           | https://soy-api-s3.s3.us-east-2.amazonaws.com/day20_15.jpg  | https://soy-api-s3.s3
.us-east-2.amazonaws.com/day20_15.jpg  | 0.39105305075645447
 61483bea-421d-45ac-8b2d-049b0c77805c | day22_15.jpg  | 9-12           | https://soy-api-s3.s3.us-east-2.amazonaws.com/day22_15.jpg  | https://soy-api-s3.s3
.us-east-2.amazonaws.com/day22_15.jpg  | 0.3912876546382904
 e2ccf972-dca1-4163-8a3a-5f0d429c04de | day22_16.jpg  | 9-12           | https://soy-api-s3.s3.us-east-2.amazonaws.com/day22_16.jpg  | https://soy-api-s3.s3
.us-east-2.amazonaws.com/day22_16.jpg  | 0.3857319951057434
 dda0aa9e-cc8d-4775-8078-97c935a059f6 | day26_6.jpg   | 9-12           | https://soy-api-s3.s3.us-east-2.amazonaws.com/day26_6.jpg   | https://soy-api-s3.s3
.us-east-2.amazonaws.com/day26_6.jpg   | 0.3960172235965729
(8 rows)
```

Figure 12: Image Data Table

The primary challenge when creating the backend subsystem was storing the image data properly. The image URL had to be stored on the database, however sometimes naming conventions were different within the S3 bucket system when they created the URL for the image in the database. Additionally, the user-given solution data from the user interface had to be analyzed on the backend. The dry weight data was used to calculate biomass, the solution data was analyzed to find the solution types that were strongly associated with the growth of the soybean.

19.3. Subsystem Validation

The UI back end was tested for the speed of response of the front end based off the image size. back end spent on processing that image. We began with 128x128 pixel images, and then progressed to 1024x1024 images, increasing image size by adding 128 pixels to each dimension of the image. As shown in the figure below, the response time is relatively inconsistent, however there is a slight increase in time spent loading based on image size.

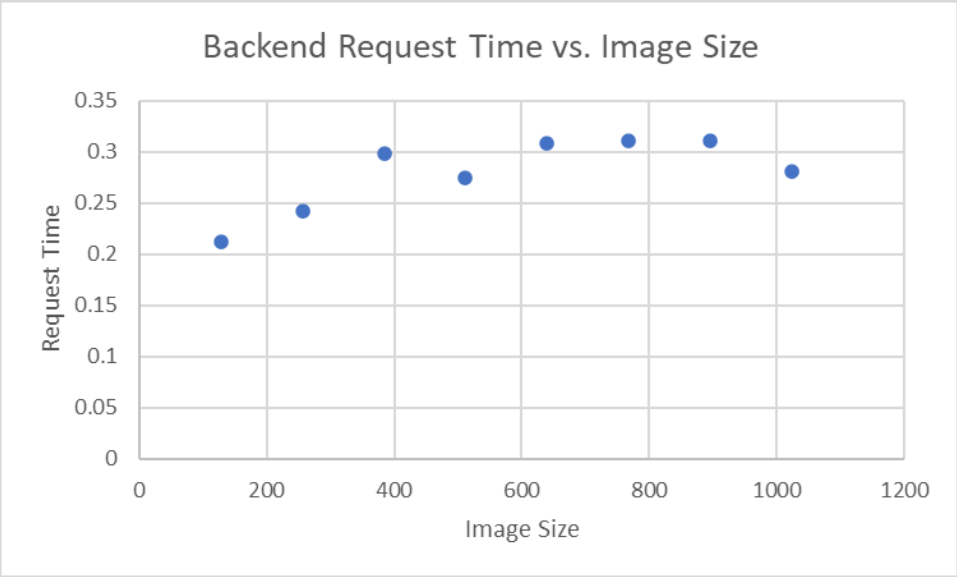

Figure 13: Back End Request and Load Time Based on Input File Size

Functionality of the back end was tested using POSTMAN as shown in *Figure 14* below. The HTTP for the back-end API was provided to POSTMAN, specified as a POST request method, accompanied by an image input for the test. For the test shown, the input image was of a soybean on Day 22 of the growth cycle. The model's output is presented here as POSTMAN's "pretty" version of the raw output from the SHDLM, giving a 39.6% accuracy of the prediction "Day 9 to Day 12". Although this image prediction is incorrect, this is the same output as the model was outputting when directly receiving data, and thus we know that the backend API is working as intended.

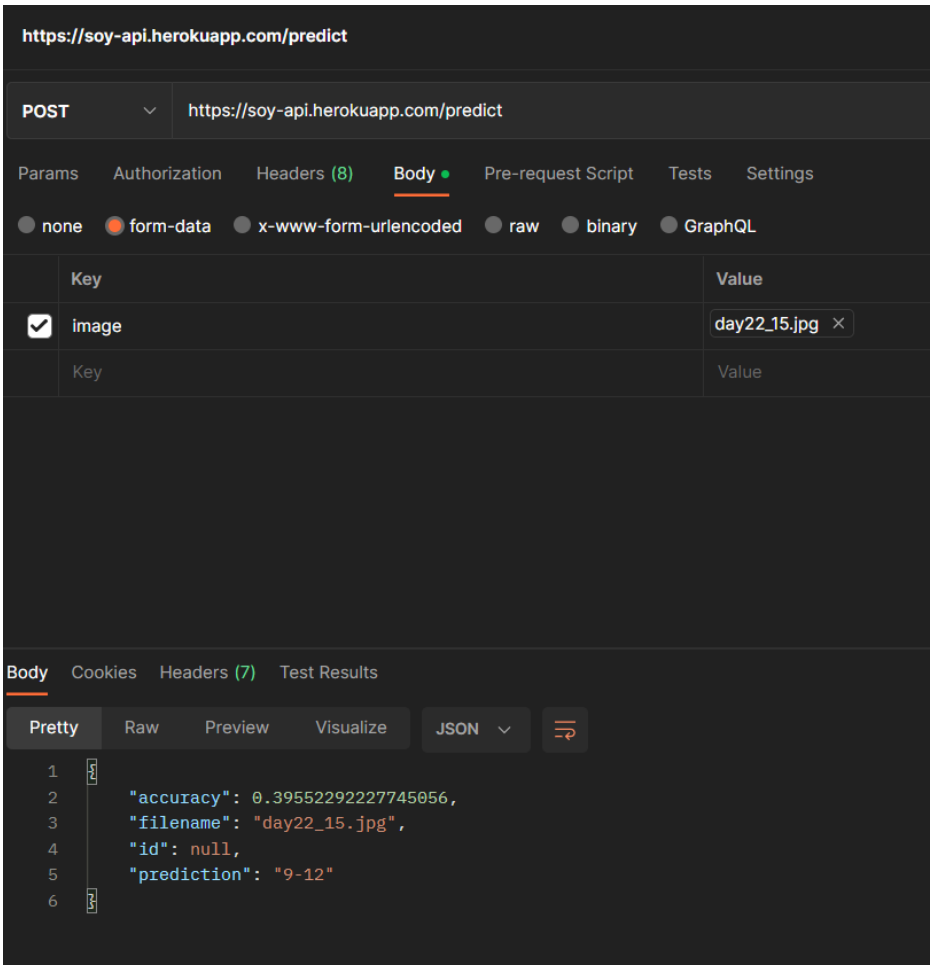

Figure 14: POSTMAN Back End Functionality Test/Simulation

20. Image Processing & Annotation Subsystem

20.1. Subsystem Introduction

The Image Processing and Annotation Subsystem was a method of creating annotated data by using the Computer Vision Annotation Tool, as well as a Python script to take the output data file and the original images to create a segmented image.

20.2. Subsystem Details

The image was first processed using the application known as CVAT (Computer Vision Annotation Tool). The data was then transformed into a JSON COCO form was then sent through a python script that parsed out the segmentation sections. Below are examples of before and after images of the annotation process.

Figure 15: Image Before the Annotation Process

Figure 16: Image After the Annotation Process

20.3. Subsystem Validation

There is no validation for the image annotation subsystem, as it is required for the training of the deep learning model.

21. Deep Learning Model

21.1. Subsystem Introduction

SHDLM, the deep learning model in this project, is a deep learning model created using Python's TensorFlow libraries, primarily Keras. The goal of this model is to employ feature extraction to classify images into day of growth. It is trained using annotated images produced by the Image Processing & Annotation subsystem.

21.2. Subsystem Details

SHDLM applies CNN technology through Tensorflow's Keras library to make predictions. A code snippet showing the use of Keras to create the CNN layers is given below.

```
#build linear model
model = keras.Sequential()
#CNN input takes tensors in shape (image_height, image_width, color_channels)
model.add(keras.Input(shape=(256,256,3)))
#add in data augmentation layer
model.add(data_rotation)
model.add(data_flipping)
model.add(data_contrast)
#add convolution and pooling layers to model -> this part is for the feature extraction of images
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=2))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=2))
model.add(layers.Conv2D(32, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=2))
model.add(layers.Conv2D(16, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=2))
model.add(layers.Conv2D(8, (3,3), activation='relu'))
model.add(layers.MaxPooling2D(pool_size=2))
#Flatten the last layer of pooling, so we can do a couple of fully connected layers
model.add(layers.Flatten())
model.add(layers.Dense(100, activation = 'relu'))
#Dense(x, activation = 'softmax') -> x = nodes on the last layer,
#and this is the number of classes you have for the dataset.
model.add(layers.Dense(4, activation = 'softmax'))
```

Figure 17: Use of Tensorflow's Keras Layers for SHDLM

Essentially, a Convolutional Neural Network is a series of layers that can extract features to classify inputs. *Figure 18* illustrates these layers, and includes elements seen in the code snippet, such as 'max pooling', 'conv', and 'softmax' layers. On the Conv2D layers, the first number specified is the size of the filter, called the depth in *Figure 18*, used to analyze the image for that layer. Please note that *Figure 18* was not created by this team, but accurately depicts the logic used for this project.

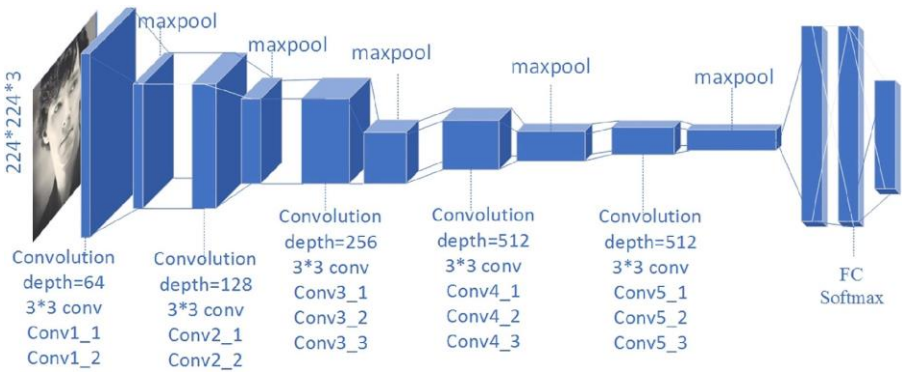

Figure 18: CNN Layers for Feature Extraction

Along with utilizing Keras' Layers, the deep learning model also employs metrics to identify the accuracy of a prediction made by the model. The accuracy data is part of the output the model sends, and it is given to the user to show the level of confidence the corresponding prediction was made with. Throughout the project, additional custom metrics were created to track precision, recall, and F1 Score, so that the team could get a better understanding of how the model was performing.

During the creation of the deep learning model, most parameters and hyperparameters were changed to find the best performance. These parameters and hyperparameters include number of layers, filter size, order of filter size, size of Dense layer, number of output classes, optimizer, loss function, and number of Epochs.

Additionally, to make the model as robust as possible, data augmentation was implemented. Data augmentation is defined by TensorFlow as, "a technique to increase the diversity of your training set by applying random (but realistic) transformations, such as image rotation." The augmentation techniques used by our team were rotation, flipping, and contrast, and they were realized as layers, which can be seen in *Figure 18*.

21.3. Subsystem Validation

The validation for the deep learning model subsystem primarily consisted of utilizing POSTMAN to upload individual images. Here, POSTMAN communicated directly to the backend Flask app which contained code that performed pre-processing and called the model for predictions. Output from POSTMAN is shown in *Figure 19.* As can be seen in *Figure 9*, the validation for the model failed, since the success criteria was making a correct prediction. Though the model was designed to be as robust as possible, it only correctly predicts Day 9-12 soybeans, as it is stuck within this range. This is because the data was biased towards the 9-12 range of the growth cycle, so the deep learning model is unable to differentiate plants outside of this range.

Furthermore, we validated that the backend Flask app was generating the output in the correct format, which was intended to be the Day of Growth range, followed by the accuracy of the prediction on the next line. This format is shown in *Figure 19* below.

```
"accuracy": 0.3988247513771057,
"message": "Model Prediction: Your plant is within Day 9 and Day 11 of
    the growth cycle."
```


Figure 19: POSTMAN Output of Model Prediction for Validation

21.4. Further Development & Usage

This deep learning model was created specifically for continued usage. If new data – in the form of images – becomes available, only a handful of tasks need to be completed to reuse the model.

21.4.1. Data Format & Data Collection

One must ensure that the new images are annotated in a similar fashion to the original annotated images used in this project; that is, the plant, or plants, should be isolated from the background of the image, and the background blacked out. If it is desired to not annotate images, as it was done manually, the new image data could be collected in a uniform fashion in the lab. For example, using a stand to hold test tubes and placing this apparatus in front of a plain background, such as a wall. Collecting data in this standardized way would eliminate the need for annotation. Supplementary data should be in the form of JPEG images. In accordance with the model training code, the images shall be labeled the same as the original dataset and placed in the Shared Google Drive, each image named in ‘Day x-y’ format, where x is the day of growth, and y is the number of the image for that day, numbered successively. The exact file path where these images shall be placed is given in the model code and does require access to the shared Google Drive provided by the project sponsor.

21.4.2. Model Code Changes

While the majority of the code can remain the same, some alterations should be made to properly include all new data available. The only mandatory change to the code is in the cell for importing the images – the second for loop shall be updated to cover the correct number of images to be imported. Originally, there were no more than 32 images per day, so it was made for a range of 35. It will need to be adjusted as necessary for additional data. Other items in the code that may be changed include the number of Epochs that the model is trained on. This shall be increased or decreased based on the performance of the model training: if the metrics are still increasing at the end of the training, one should increase the Epochs. Also, one could change any of the parameters and hyperparameters, which are optimizer, loss function, layers, activations and filter sizes on convolutional and dense layers, and metrics. Finally, the range of days could be altered to better match all ranges to each other, in order to have a balanced number of images in each.

22. Appendix A: Acronyms and Abbreviations

AWS	Amazon Web Services
AFD	Application Failure Detection
CVAT	Computer Vision Annotation Tool
CNN	Convolutional Neural Network
ICD	Interface Control Document
MFD	Model Failure Detection
N/A	Not Applicable
SHDLM	Soybean Hydroponic Deep Learning Model
UI	User interface

23. Appendix B: Definition of Terms

Deep Learning Model:
A computational model that processes data with multiple layers of pattern recognition in order to better identify data on a higher level.

Image annotation:
A method of assigning segments of an image a label to help train the model for future input data.

The following definitions differentiate between requirements and other statements.

Shall:	This is the only verb used for the binding requirements.
Should/May:	These verbs are used for stating non-mandatory goals.
Will:	This verb is used for stating facts or declaration of purpose.