Water-Intensive Crop Disease Detection and Classification Using Machine Learning

A Project Report
Presented to
The Faculty of the Department of Applied Data Science
San Jose State University
In Partial Fulfillment
Of the Requirements for the Degree
Master of Science in Data Analytics

By

Nassim Ali-Chaouche - [redacted]

Jason Widjaja Djuandy - [redacted]

Nail Enikeev - [redacted]

Vrushali Pravin Menthe - [redacted]

May 13, 2024

APPROVED FOR DEPARTMENT OF APPLIED DATA SCIENCE

Dr. Jerry Gao, Project Advisor

Dr. Lee C. Chang, Department Chair

ABSTRACT

In the face of escalating challenges in agriculture such as disease management and yield optimization, advanced technological solutions have become crucial. This project leverages image data combined with remote sensing and meteorological data to monitor the health of rice as a water-intensive crop. Utilizing a complex data integration framework, imagery and numerical data are combined based on geolocation and timestamps, creating a rich dataset for each observation point. A novel aspect of this research involves the development of a hybrid deep learning model that synergizes the strengths of convolutional neural networks (CNNs) and dense artificial neural networks. Specifically, the visual data processed by a CNN (using transfer learning) is integrated with tabular data comprising weather and remote sensing information that is processed through a sequence of dense layers. This model is capable of performing detailed disease classification in rice crops, distinguishing between healthy plants and those afflicted by disease. Extensive model testing highlighted the superiority of hybrid models compared to their visual-only counterparts, and more specifically the advantage of a hybrid model employing a DenseNet121 architecture in the visual processing branch, due to its performance (98.4% accuracy) and computational efficiency. This model was consequently chosen for deployment. The culmination of this project is a web portal that presents the predicted disease status of the crops and their spatial distribution in a given field. This portal serves as an interactive tool for farmers, enabling them to visualize crop conditions effectively. By facilitating informed decision-making regarding crop management and disease treatment, the portal enhances the capability of farmers to implement more sustainable agricultural practices and improve yield outcomes by mitigating the impacts of the crop disease at hand.

ACKNOWLEDGMENTS

We extend our deepest gratitude to Professor Jerry Gao for his invaluable contributions to this research. The expertise and insightful critiques provided have greatly enhanced the quality of this work. We are especially thankful for the generous allocation of resources and the continuous support throughout the development of this study.

We would also like to acknowledge our family members for their continued support and encouragement.

TABLE OF CONTENTS

1. Introduction

1.1 Project Background and Executive Summary

In the realm of agriculture, the health of crops plays a pivotal role in ensuring food security and economic stability worldwide. Diseases in crops can lead to severe reductions in both the quality and quantity of agricultural produce, thereby threatening food supply and farmer incomes. Two main issues that hinder crop production are pests and crop diseases. According to the FAO (Food and Agriculture Organization of the United Nations), "FAO estimates that annually between 20 to 40 percent of global crop production is lost to pests. Each year, plant diseases cost the global economy around $220 billion, and invasive insects around US$70 billion" (FAO, 2019). Changes in weather patterns globally as well as current world events has also been shown to affect crop yields. According to CNBC, "There's a strained supply of rice as a result of the ongoing war in Ukraine, as well as weather woes in rice-producing economies like China and Pakistan" (Shan, 2023). Another article mentioned, "Rice prices surged to their highest in almost 12 years after supplies in Asia's primary staple food were limited by India's rice export ban and adverse weather conditions" (Tan & Shan, 2023).

The importance of keeping crop yield high at all times have been underlined with the facts stated above. Factors such as weather change and global political events may be unpredictable however early detection of pest and crop diseases will provide the ability to mitigate further damage done, and preventative measures such as removal of disease-affected crops can be done to prevent further damage and spreading of the disease. Doing so will in turn ensure food security and continued food production for the people. This project will look further into rice as a model for crop disease detection of water-intensive crops; these are crops that

require significantly more water for their growth and are a common staple in Asia where the larger part of the world population presides.

Traditional methods of disease detection often involve manual inspection of crops, which is labor-intensive, time-consuming, and prone to human error. With the recent technological growth in the field of machine and deep learning, the problem of crop disease and detection can be tackled with modern image classification solutions. Machine learning models such as Convolutional Neural Networks can be trained for image-based tasks, such as crop disease detection, and be able to classify which diseases could be present based on the picture of the crop. The machine learning aspect allows the automation of crop disease classification given that the model is trained and the model is fed through pictures of crops on the farm. Other solutions look into the weather data patterns as well as remote sensing information from locally placed hyperspectral sensors to detect the spectral values at which these crop leaves reflect. The use of machine learning and artificial intelligence allows for timely interventions, optimizing water usage, and reducing the need for extensive pesticide applications, which are often the default response to late-stage disease detection. Image data gathering can be collected with the help of drones such that farmers would not need to manually do physical intensive work in the fields daily. The developed machine learning model can utilize multiple sources of data such as images as well as tabular data (weather and remote sensing data) to be able to perform classification tasks of diseases based on the provided information. With the classifications in hand, farmers will then be able to take preventative measures to mitigate the damage done by the crop disease such as the removal of affected crops.

To create a water-intensive crop disease detection system, the project will initially train deep learning models suitably used for image classification tasks. Additional models will be

trained to detect crop diseases from pictures of crops as well as combining image and tabular data to see and explore whether the model may produce better results. The dataset used to train the models must be labeled field leaf images, in contrast to singular leaf images with a white background. Labeled data allows the model to perform classifications and field images allow the model to simulate what the end use case scenario will be like as images taken by farmers will be in the crop fields. Automation of image gathering can be easily applied and realized with the help of drones. The gathered data will then be fed through the model with the results displayed through a web portal which farmers can use for crop disease detection and to support or monitor the crop disease distribution and progression in their multiple farmlands. Farmers will then be able to create data-driven decisions to tackle the problem of crop diseases.

The impact of the project is to provide a working model concept helpful for the agricultural industry. Tackling the issue of water-intensive crop disease classification with the help of drone automation will allow farmers to better handle issues in food production and reduce the chances of food shortage and food insecurity. Farmers will be able to detect and efficiently handle the problem when compared to traditionally where the farmer has to be the one checking the fields on food and looking for potential crop-disease distributions on the farmland. One of the challenges with growing water-intensive crops is the large energy and water consumption needed to grow these crops. In regions where water is scarce, water must be utilized carefully and not wastefully, given that the farmer notices an area infested with crop disease; handling the disease-affected crops earlier without expending more resources on the disease-affected crops will be helpful in water and energy consumption as well. The project will also push the development of models for smart agriculture applications forward allowing farmers to be able to utilize their time more efficiently and productively elsewhere on other tasks.

1.2 Project Requirements

The goal of this project is to explore and develop machine learning models which not only use image detection features but also incorporate other information known to be a factor in water-intensive crop disease such as weather and remote sensing data. The usage of more data could potentially provide better results in water-intensive crop disease detection. The developed model can be applied into an interactive web portal application for the end user, the farmer, to use in order to gauge an understanding of crop disease progression in the fields. The requirements of this project is to first develop and explore machine learning disease classification models which can have the performance measured by the accuracy at which the model is able to detect an existing crop disease. Machine learning model comparisons between current solutions which purely use image detection against the proposed 'hybrid model' which incorporates weather and remote sensing data on top of the existing image detection model will be explored to see if the latter can yield better results. The model must be able to detect and classify images taken in the field to simulate the data taken by the end-user or by the use of drones. This is important as the majority of datasets available online provide labeled single-leaf images on a white background, and if these single-leaf images were used for machine learning model training, the final developed model might not be able to perform well when fed pictures of crop diseases taken in the fields. As the application for this project may include the use of drones in the future, measures need to be taken to ensure that the model is able to take in images of crops from different angles as well as different lighting conditions which reflect the images taken by drones during different times of the day as well. Other functional requirements include allowing the end user to be able to determine the dimensions of the farmland they have, as

different farmland may have different dimensions. Additionally, farmers may have multiple fields with differing dimensions, as well as different crops.

1.3 Project Deliverables

The deliverables of this project will include the deep learning model selected for the image classification task of detecting water-intensive crop diseases which may incorporate supplemental information in the form of weather and remote sensing data. The model will be able to detect water-intensive crop diseases given a test image after being trained on images of crops with diseases, as well as tabular data in the form of weather and remote sensing data. Training, validation, and testing sets will be generated, and different evaluation metrics will be used to determine the performance of the model. An optimized model should be able to perform disease classification at a high level of accuracy, and being a lightweight model is a plus. An interactive web portal will be built which can take multiple batches of images taken from the field, and provide classifications of multiple diseases for each supplied image. The predictions will be visualized in a grid format where the end user can customize the grid size. Frequent time interval classification allows for the end user to be able to look at the disease progression in the fields and come up with preventive measures to mitigate further damage done on the crops by certain diseases. Finally, a detailed report documenting the detailed steps and information gathered throughout the project will be included as well.

1.4 Technology and Solution Survey

In the domain of precision agriculture, swiftly and precisely identifying and categorizing crop illnesses emerges as a crucial task, directly impacting the health of crops, maximizing yields, and promoting sustainable agricultural methods. Leveraging advanced machine learning

algorithms offers a promising avenue to address this issue, facilitating the early detection and accurate classification of diseases across extensive agricultural fields.

Machine learning, and in particular deep learning, plays a crucial role in converting captured images into actionable insights. Among the wide array of algorithms available, Convolutional Neural Networks (CNNs) have demonstrated exceptional performance in image classification tasks. These models excel at autonomously learning hierarchical features from images, rendering them exceptionally suitable for identifying and classifying crop diseases based on patterns and anomalies in aerial images. In addition to the image data, numerical data containing weather and satellite remote sensing data will also be considered when constructing the models. For this data, Artificial Neural Networks (ANNs) containing dense layers can be utilized to extract any patterns and create a more robust model as opposed to utilizing solely image data. When simplicity and interpretability are of paramount importance, traditional machine learning algorithms such as SVM and k-NN may also be suitable.

However, the application of CNNs and other machine learning models in agricultural scenarios often encounters the challenge of a limited number of high-quality, labeled data. To address this, transfer learning has emerged as an effective tactic, allowing models pre-trained on large, generic image datasets to be fine-tuned for the specific task of classifying crop diseases. This method capitalizes on the generic image features learned by the model during its initial training, tuning it to identify specific patterns linked to various crop diseases. Transfer learning not only alleviates the issue of data scarcity but also accelerates the training process and bolsters model performance, providing a practical solution for real-world agricultural applications.

In addition to ANNs, CNNs, and transfer learning, model ensembles add an extra layer of complexity and robustness to machine learning applications for crop disease classification.

Model ensembles combine predictions from various machine learning models to enhance the overall accuracy and reliability of the system. Techniques such as bagging, boosting, and stacking are commonly utilized ensemble methods.

By merging image, weather and satellite remote sensing data, advanced deep learning algorithms, transfer learning strategies, and model ensembles, this approach to precision agriculture offers a thorough and robust solution for the classification of crop diseases. This not only guarantees the accuracy and reliability of disease identification but also supports timely and informed decision-making, contributing to sustainable agricultural practices. As these technologies are refined, their integration into precision agriculture is set to revolutionize the agricultural sector, helping to ensure food stability and sustainable farming in the midst of global challenges and population shifts.

1.5 Literature Survey of Existing Research

Harika et al. (2023) developed a model to predict diseases in black gram crops using leaf images. The goal was to classify four main diseases (anthracnose, leaf crinkle, powdery mildew, and yellow mosaic) that have a detrimental impact on black gram production in India. The authors utilized the Black Gram Plant Leaf Disease (BPLD) dataset containing 1000 labeled images across 5 classes, with 4 representing various diseased states and 1 healthy state. Several machine learning models, including k-nearest neighbors, decision tree, and random forest were trained on the dataset features. Additionally, an artificial neural network and a deep convolutional neural network model were tested. Model performance was evaluated using the metrics of accuracy, precision, and recall. Of all the models, the convolutional neural network achieved the highest predictive accuracy at 89.5%, along with strong precision and recall. The authors suggest future work could focus on identifying multiple diseases in a single plant image.

Saeed et al. (2021) developed a model to identify and classify diseases in multiple crops, specifically rice and corn, through the use of convolutional neural networks (CNNs). The goal was to create a generalized approach that could exploit existing CNN frameworks. They employed public datasets containing images of diseased rice leaves including brown spots, hispa, and leaf blast, as well as infected vs healthy corn leaves. The nature of the rice dataset was fairly complex, so they divided it into major (clearly diseased) and minor (disease symptoms are subtle) subsets. They evaluated variants of two CNN models - ResNet152 and InceptionV3 - by adding an additional dense layer prior to the final output layer. For the rice dataset, their ResNet152 model achieved 99.1% accuracy on the major disease subset and 82.2% on the minor subset. For the corn data, their InceptionV3 model obtained the highest accuracy at 97.8%. Overall, the strategy of fine-tuning CNNs with added layers demonstrated high accuracy in diagnosing multiple crop diseases from leaf images, and the authors highlight that this could facilitate automated disease diagnosis to aid farmers.

Jha et al. (2023) conducted an extensive analysis of various machine-learning techniques for identifying crop diseases from images. The authors shed light on prevalent challenges such as inconsistent image conditions and the necessity for high-quality training data. They reviewed past studies that applied algorithms like SVM, decision trees, KNN, and CNN for disease classification in crops like potato, rice, and corn. Depending on factors like the algorithm, dataset size, and number of classes, the reporting accuracy rates varied between 88 and 99 percent. Despite these promising results, they pointed out limitations around using single models and potential data loss during feature extraction. They suggested that the adoption of enhancements like ensemble models and swarm intelligence could further improve accuracy levels. The authors

advocated for ongoing research to develop rapid and precise systems to assist farmers in the timely identification of crop diseases.

Gupta et al. (2022) developed convolutional neural network (CNN) models for crop disease and weed detection with the aim of assisting precision agriculture. They utilized public datasets comprising images of diseased crop leaves (potato, tomato, bell pepper), along with soybean fields with weeds. For crop disease detection, they compared MobileNetV2, InceptionV1, and NASNetMobile CNNs, and ultimately NASNetMobile achieved the highest accuracy of 82.57%. For weed detection, they compared ResNet50V2, MobileNetV2, InceptionV3, and NASNetMobile CNNs, with InceptionV3 achieving the highest bounding box accuracy of 77.23% on the soybean images. Based on the results, the authors built a web application capable of identifying crop diseases and locating weeds in farm images, providing valuable insights to farmers. They suggest that with larger datasets, the models could be expanded to encompass more crops and enhance their performance further. Overall, the study demonstrates that CNNs have the capacity to accurately classify crop diseases and weeds from images to support agricultural automation and more informed decision-making. The models encountered obstacles related to the size of the datasets but showed encouraging results on available public data.

Soni et al. (2021) developed an approach using image processing to automatically detect diseases in multiple crop leaves, including cucumber, guava, groundnut, pumpkin, and cowpea. The authors collected a dataset of 406 images of healthy and diseased leaves affected by rust, anthracnose, and powdery mildew. Their methodology encompasses several steps: preprocessing, thresholding, Sobel edge detection, morphological dilation, and ROI extraction to isolate regions affected by disease. Using this approach on the dataset, a 98.27% accuracy was achieved in

correctly classifying diseased vs healthy leaves, with only 7 out of 406 samples being misclassified. The model also achieved high precision, recall, and specificity. The authors conclude that digital image processing techniques hold substantial potential in identifying multiple crop diseases from leaf images to assist farmers. Future work that they plan on conducting includes augmenting the dataset and delving further into machine/deep learning techniques to drive further improvements.

Hu et al. (2021) introduced a method based on a conditional generative adversarial network (GL-CGAN) to augment limited crop disease image datasets, aiming to achieve improved classification. Insufficient training data and class imbalance are obstacles to developing crop disease recognition models. Their GL-CGAN method combines GL-GAN, which performs global and local adversarial learning, along with ACGAN which facilitates conditioned image generation. The method was evaluated on the PlantVillage dataset of crop leaf images across multiple diseases. Synthetic data was generated through this process to expand the original dataset. Subsequent testing on the PlantVillage dataset reveals classifiers such as ResNet and VGG exhibited improved performance on the augmented dataset compared to the original, with higher accuracy, weighted accuracy, and F1-score. The observed improvements were marginal, indicating the method needs further refinement. However, the results underscore that GANs can generate useful synthetic training data when confronted with limited disease image sets. The authors have outlined plans to optimize their GL-CGAN method to produce more realistic crop disease images to achieve higher classifier performance. Overall, the proposed GL-CGAN method is a promising approach for data augmentation to improve automated crop disease classification with deep learning when faced with datasets that are constrained.

Ren et al. (2023) developed a multi-scale parallel fusion convolution network (MSPFNet) to enhance the accuracy of crop disease identification from image data. Recognizing diseases is inherently challenging due to varying lesion sizes and complex backgrounds. The MSPFNet model attempts to address these issues by using multi-scale convolution which extracts features at different scales. Furthermore, it also integrates a Convolutional Block Attention Module to focus on relevant features and suppress background noise. A notable feature of the network is a feature fusion module, which is designed to prevent the loss of disease details during the downsampling process. When tested on an apple leaf dataset comprising 5 diseases, MSPFNet achieved 97.2% accuracy, outperforming the performances of AlexNet, VGG, ResNet, DenseNet, and MobileNet-V3. Ablation studies conducted by the authors highlight that the multi-scale convolution and attention modules yielded improved performance over the baseline model. Visualizations also illustrate that MSPFNet better locates diseased regions across a variety of sizes. In summary, the multi-scale, attention, and fusion mechanisms enhanced the network's ability to learn from crop disease features, leading to state-of-the-art accuracy. The authors advocate for MSPFNet as an effective architecture for the automated recognition of crop diseases from images. The following is a table comparing all of the surveyed papers.

Table 1

Comparison of relevant works

Survey Author	**Solution**	**Method**	**Models**	**Results**
Harika et al. (2023)	Black Gram Crop Disease Detection	4 diseases - anthracnose, leaf crinkle, powdery mildew, yellow mosaic. 1000 images dataset.	k-NN, Decision Tree, Random Forest, ANN, CNN	CNN highest accuracy 89.5%
Saeed et al.	Multi-Crop	Rice dataset	ResNet152,	Rice -

(2021)	Disease Detection - Rice and Corn	separated into major/minor subsets. Tested ResNet152 and InceptionV3 variants.	InceptionV3	ResNet152 99.1% major, 82.2% minor. Corn - InceptionV3 97.8% accuracy.
Jha et al. (2023)	Comparative Analysis of Crop Disease Detection Methods	Review of image processing and ML techniques for crop diseases.	SVM, CNN, Decision Trees, etc.	Varying accuracy from 88-99% reported based on dataset, classes, and model.
Gupta et al. (2022)	Crop Disease & Weed Detection	Crop - compared CNNs on PlantVillage data. Weed - soybean images, bounding boxes.	MobileNetV2, InceptionV1, NASNetMobile etc.	Crop - NASNetMobile 82.57% accuracy. Weed - InceptionV3 77.23% bounding box accuracy.
Soni et al. (2021)	Multiple Crop Disease Detection	Preprocessing, thresholding, Sobel edge detection, morphological dilation, ROI extraction. 406 images dataset.	Custom algorithm	98.27% accuracy in detecting diseased leaves.
Hu et al. (2021)	Crop Leaf Disease Data Augmentation	GL-CGAN to generate synthetic data for classifier training. Tested on PlantVillage dataset.	AlexNet, VGG16, ResNet18, etc	Improved accuracy, weighted accuracy, F1-score after data augmentation.
Ren et al. (2023)	Apple Leaf Disease Identification	Multi-scale convolution, attention & fusion model MSPFNet. Public dataset of 5 diseases.	MSPFNet, AlexNet, VGG16, ResNet18, etc.	MSPFNet achieved 97.2% accuracy.

2. Data and Project Management

2.1 Data Management Plan

In order to effectively build the data workflow across the project, there are four components that have been worked out. The components are as follows: data collection approaches, management methods, storage methods, and usage protocols specifically designed for handling data from different types of cameras in the classification of crop diseases. High-resolution Images are taken from different crop fields using cameras. The dataset provides comprehensive coverage and timely obtained data. Each image is labeled with the respective disease presence. Labeled data is used as input for training and testing a model. Image metadata, such as GPS coordinates (crop location on the field) is also collected to further enrich the data by obtaining weather and remote sensing data for the development of the end-user solution application.

The design of data workflow starts with uploading new data from the cameras, which takes images of the crops in the fields in different locations. The images may be collected in different environmental conditions and orientation. This can lead to poor image quality or unclear details, which makes the image useless for training or testing purposes. To overcome the quality factor, a solution may be to have a script to automatically detect images unsuitable for analysis. Also, to ensure data consistency and quality manual reviews might be scheduled. The labeling for new training data should be done in collaboration with plant pathologists or agronomics to ensure the precision of disease classification. Labeling software such as Google Vertex AI or similar might be used to sustain productivity in this step.

The data workflow leads to an increasing number of instances in the dataset. As an object storage platform, to handle growing data size and image metadata, Google Cloud Storage is selected. Model classification records are stored in the Google Cloud SQL database.

The usage mechanism for the model training and finetuning is as follows: Google Colaboratory and the Google Vertex AI environment connects to Google Cloud Storage and refers to the datasets to ingest the data into processing and modeling scripts, that are coded in the Python programming language and deployed in the Google Cloud Functions. The scripts ensure that the images are standardized in terms of size, resolution, and format and perform feature extraction to catch disease patterns. In addition, there are scripts that fetch weather and remote-sensing data based on the extracted metadata for the hybrid models' numerical branch. Thus, from the data the model classifies crop diseases and the post-processing script uploads the classification records to Google Cloud SQL. The end-user web application, deployed on Google App Engine, uses SQL and API to retrieve classification records and present them to a user using a convenient graphical user interface for further analysis and decision-making process.

2.2 Project Development Methodology

The CRISP-DM approach is utilized for detecting diseases in water-intensive crops. CRISP-DM is divided into six stages. It offers an organized method for solving the agricultural domain's particular issues. Through its iterative nature, it refines the overall project by continuous improvement.

2.2.1 Business Understanding

Business understanding is the first phase of the CRISP-DM methodology. This phase identifies and defines the objectives, goals and planned approaches. Integrating machine learning

into agriculture has the potential to effectively address challenges such as water-intensive rice crop disease classification on a wide scale. This project brings together technologies with a focus on integrating imagery, remote sensing and weather data to detect crop diseases and categorize them.

In the first phase, it is crucial to address the needs of key stakeholders, particularly farmers and agricultural enterprises. Their valuable insights play a vital role in determining the project's objectives, which revolve around real-world consequences of diseases, optimal intervention timing and desired outcomes. This phase also involves evaluating the economic feasibility, potential return on investment and overall expectations of the technological solution.

Understanding the business context, the generated data is transformed into valuable information. This not only helps in disease detection but also equips users with the knowledge to make well-informed farming decisions. Ultimately, this enhances productivity and profitability.

2.2.2 Data Understanding

The second phase of CRISP-DM is data understanding. A fundamental understanding of data is required before diving into data-driven decision-making, especially in the agriculture department. In this stage, the complexities of the dataset are examined to gain an understanding of its characteristics, features and potential obstacles.

For this project, the sources of data comprises camera images taken in crop fields, remote sensing data and weather data. These images are expected to be high-resolution, detailed and capture both the vastness of the farm and the intricate features of the crops. The first step involves conducting an exploratory data analysis (EDA). This can include steps such as looking into the distribution of the target variable, checking the dimensions and the file types of the images, and checking for missing values in the remote sensing and weather data.

Additional information such as timestamps, geographical coordinates, or details about the images collected can be used to provide insights. The remote sensing data can provide NDVI and EVI information about the crops, and the weather data like temperature and rainfall are helpful. Another important aspect is assessing the quality of the data like missing images, flaws and distortions in the images, lighting conditions, etc. which is crucial because the accuracy and comprehensiveness of the data directly impact the performance of machine learning models.

Moreover, it's essential to understand the limitations of the data. There might be diseases or symptoms that are not adequately represented in the dataset or perhaps most of the images come from a season or specific geographical area. By examining and comprehending the data at hand, solid foundations can not only be laid for subsequent project stages but also proactively address any potential challenges to ensure our solutions are robust and well-informed.

2.2.3 Data Preparation

Data preparation is the third step in CRISP-DM methodology. It involves transforming data into a format suitable for modeling. In the case of applications and using camera imagery, remote sensing data and weather data, the approach to data preparation for detecting water-intensive crop diseases has its own unique requirements and challenges. The following steps are performed:

Data cleaning - To address any inconsistencies, inaccuracies, or gaps in the dataset during this process. In the context of camera imagery, this might involve filtering images without corresponding metadata information and removing duplicate images. Missing and erroneous values in the numerical data should be checked and dealt with appropriately, for instance through imputation or removing the missing values.

Data augmentation - To enhance the robustness of a dataset since it is visual in nature, augmentation of the images is done using various techniques. These techniques can include rotations, flips, zooms and color adjustments. Increasing the size of our dataset through augmentation methods exposes the model to diverse representations of the data. This helps prevent overfitting and makes the model more adaptable.

Feature engineering- This can include creating new variables from the remote sensing sensing and weather data which might be indicative of disease.

Data splitting- Splitting the dataset into training, validation and test subsets is crucial. Data will be split into 70% for training, 15% for validation and 15% for testing. This approach allows the model to train on one set of data, to fine-tune it on another set and then to evaluate its performance on an unseen set. This comprehensive method provides a perspective on how the model performs.

Handling imbalanced data- Some classes may contain less data than others. If it is not addressed, the model could develop biases towards the majority classes. To ensure a fair representation, techniques such as resampling utilizing weighted loss functions or generating data are recommended.

The thoroughness applied to the data preparation phase directly impacts the efficiency and accuracy of the subsequent modeling stage.

2.2.4 Model Development

The next stage in the project is the development of machine learning models. In the project, machine learning models are initially created that are specifically designed to analyze only crop disease image data. To process the images efficiently, Convolutional Neural Networks (CNNs) are favored. These networks have specialized layers that can identify patterns, edges and

textures in the images making them ideal for analyzing camera pictures. Additionally, models which incorporate both image data and the tabular data consisting of remote sensing and weather data will be created. The numerical data can be processed through an Artificial Neural Network (ANN) made up of a sequence of fully connected layers. Such strategic model selections are consistent with the goals, which include focusing on precise disease diagnosis and providing actionable information for the agriculture sector.

2.2.5 Model Evaluation

The fifth stage in the project deals with the evaluation of the models. When it comes to creating a machine-learning solution, for detecting diseases in water-intensive crops it's crucial to thoroughly evaluate the models. Accuracy, precision and recall measures will be used for how well the models perform in identifying diseases. The F1 score will provide an assessment of how the models perform by balancing the precision and recall metrics. To further assess how the models distinguish between healthy and diseased crops, the AUC ROC score will be used, with a higher score indicating better ability to distinguish between classes. Additionally, further examination of these metrics will be done to see how well different models detect individual healthy and disease classes. This also provides an evaluation on the differences in performance when comparing visual only models to hybrid models. This comprehensive evaluation process ensures that the models are not theoretically sound but practical in making informed decisions in the agricultural sector.

2.2.6 Deployment

The next and final phase in the project is the deployment phase. Once the machine learning models have been developed and evaluated, it then involves integrating these into an environment so that farmers and agricultural businesses can effectively utilize them.

In the project context, the deployment process will include integrating the models into a web-based portal. The backend system will then analyze these images using our models to identify and classify diseases. The results, which include disease distribution maps that show locations and severity levels will be visually presented to the users.

Furthermore, implementation monitoring will be done to ensure that the deployed models maintain accuracy as new data becomes available. This requires retraining or fine-tuning of the models. Lastly, the projects aim to facilitate adoption and usability by providing documentation, user guides and support resources. The goal is to enable end users to benefit from disease detection systems and optimize water-intensive crop yield and health within the agriculture community.

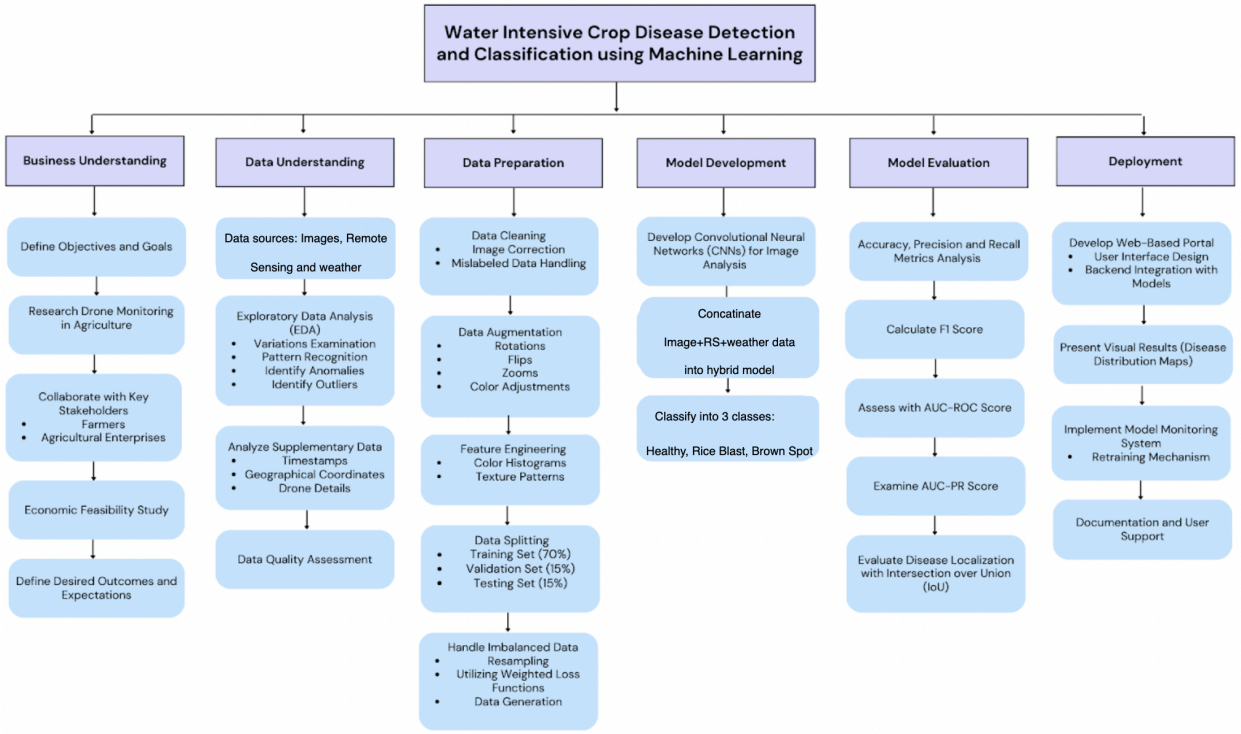
2.3 Project Organization Plan

The work breakdown structure (WBS) is used in this project to detect diseases in water-intensive crops. WBS helps organize tasks in a manner that clearly defines what needs to be delivered and then allocates resources effectively and manages budget. This creates a feasible schedule, and it also provides a representation of the project's scope making it easier to track progress and stay aligned with the project’s objectives. Additionally, the WBS allows to identify and address risks at every stage by breaking down tasks into units. It simplifies the overall complexity of the project, promotes better understanding within the team and ensures that no

critical aspect is overlooked. As a result, it can execute the project efficiently and maintain better organization throughout its implementation.

Figure 1

Work Breakdown Structure (WBS)



2.4 Project Resource Requirements and Plan

The entire solution requires suitable neural network model architecture, such as convolutional neural networks, aided by the use of transfer learning to increase the training speed and ameliorate the model's performance.

As a model development, training, and fine-tuning environment Google Colaboratory platform is selected. The environment provides Jupyter Notebook service. To maintain training at reasonable processing speed rates, a high-performance computing system is required. The

Google Colaboratory environment provides access to low, middle, and high-end Nvidia GPUs, such as T4, P100, V100, and A100 at an accessible resource for the project's budget cost.

Python programming language is selected as it has become the de facto standard for artificial neural network development and, more broadly, for deep learning and machine learning. Python is also used for the Flask web application development engine.

TensorFlow and PyTorch are selected as they are popular, flexible, rich in libraries, and vastly supported by community deep learning frameworks.

To train a core version of the model, Kaggle datasets are used. The datasets are publically available and free. Further, in a completed solution with the web application, to obtain new raw data for training and classification, a phone, stationary or a drone camera can be used. Such specifications are necessary for flying over the crop fields with multiple covering patterns, in different weather conditions and to take suitable quality images for neural network training.

When choosing a storage solution to manage high-resolution crop images for classification by a neural network model, it is crucial to weigh the capabilities needed for iterative fast reading and processing of the data. Also, to deal with high volumes of high-resolution images, storage scalability is a top priority. Considering the nature of the project velocity, which stands for how quickly data is being ingested into the model, can be the second priority. Thus, as an object storage platform, to handle existing datasets and images, Google Cloud Storage is selected as it provides: maintenance-free scalability at cheap rates, native integration with Google Vertex AI - the model development environment, and Google App Engine - web application deployment container service.

At the same time, the model classification records alone are stored in the Google Cloud SQL database, which is natively connected to the Google App Engine. The Cloud SQL relational

database is an optimal choice as it is perfectly suited to store model disease classification records.

The mentioned services are part of the Google Cloud Platform, which seamlessly integrates them providing an auto-scalable and maintenance-free solution. Such an approach allows the project team to focus on designing the data pipeline, developing a model, and building the solution. Furthermore, App Engine along with Cloud SQL, Vertex AI, and Cloud Storage, easily handles security setups, such as IAM roles and service accounts, to manage access for different APIs, users, and other components of the solution.

The team as a local desktop machine is using an Apple Macbook and HP laptops, which are operated by MacOS 14 Sonoma and Windows 11 operating systems. The local machines handle the teams' workflow and provide access to the internet, all the above-mentioned environments, and various services.

For effective communication, collaboration, and efficient project workflow, additional tools, such as Google Docs, Google Slides, WhatsApp, Gmail, Zoom, Google Meet, Miro, etc are used. The project resource requirements along with the plan are shown in Table 2.

Table 2

Project resource requirements and plan

#	Function	Resource Type	Resource	Time requirement	Cost
11	Deep Learning Framework	Software	Tensorflow 2.14.0 PyTorch 1.9	10/02/2023 - 5/10/2024	Free
12	Coding Language	Software	Python 3.11	10/02/2023 - 5/10/2024	Free
13	Text editor	Software	Google Docs	8/28/2023 -	Free

				5/31/2024	
14	Presentation composer	Software	Google Slides	8/28/2023 - 5/31/2024	Free
15	Gantt Chart, PERT Chart, Workflow charts and diagrams	Software	MIRO	8/28/2023 - 5/31/2024	Free
16	Team collaboration	Software	Gmail, Zoom, Google Meet, WhatsApp	8/28/2023 - 5/31/2024	Free

2.5 Project Schedule

From the beginning of the project, our aim has been to delve deep into water-intensive crop disease detection and classification, exploring potential technologies and tools for the final solution prototype. Research, including model selection and architecture design, has spanned the entire project to ensure optimal outcomes. The integration of innovative ideas and solutions at various stages has further solidified the project's foundation. The Gantt chart outlining the project timeline can be found below. This chart serves as a guide to ensure each task aligns with the overall project delivery schedule. It offers a detailed view of individual tasks, their sequence, the status of deliverables, and the team member accountable for each task along with the start date, finish date and duration, with arrows connecting tasks and indicating their order. The Gantt and Pert Charts are displayed in Figure 2-9.

Figure 2

Gantt Chart - Phase 1, Project Planning

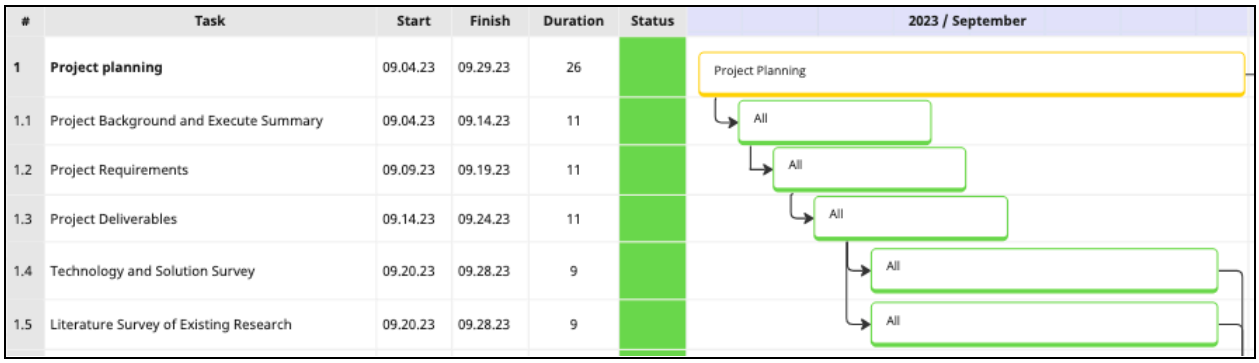


Figure 3

Gantt Chart - Phase 2, Data and Project Management Plan

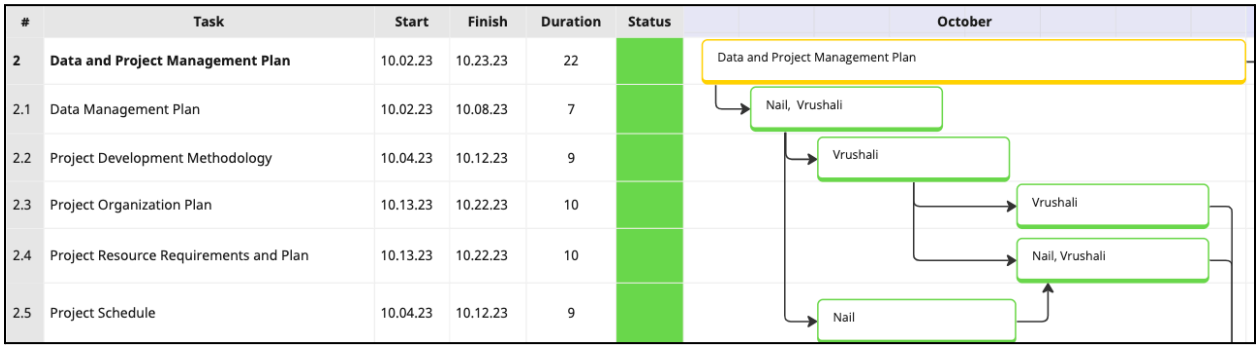


Figure 4

Gantt Chart - Phase 3, Data Engineering

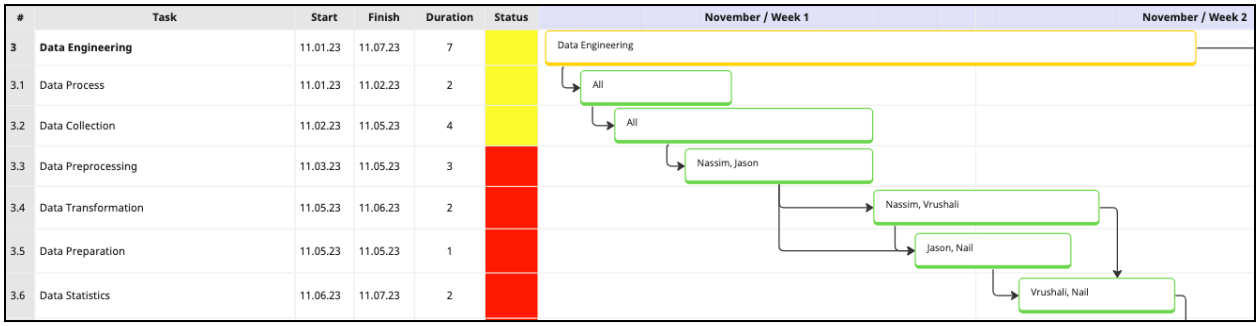


Figure 5

Gantt Chart - Phase 4, Model Development

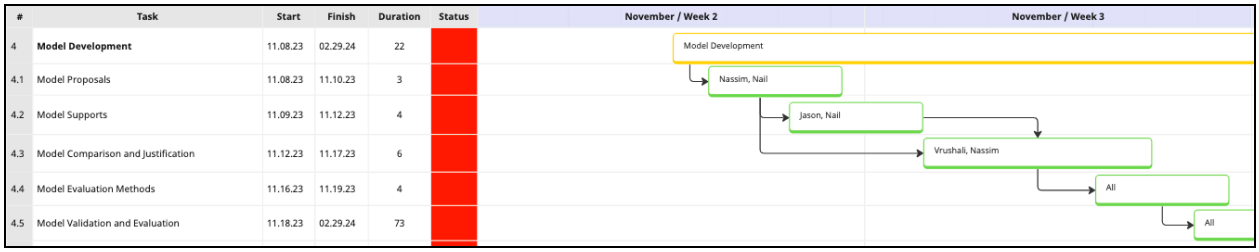
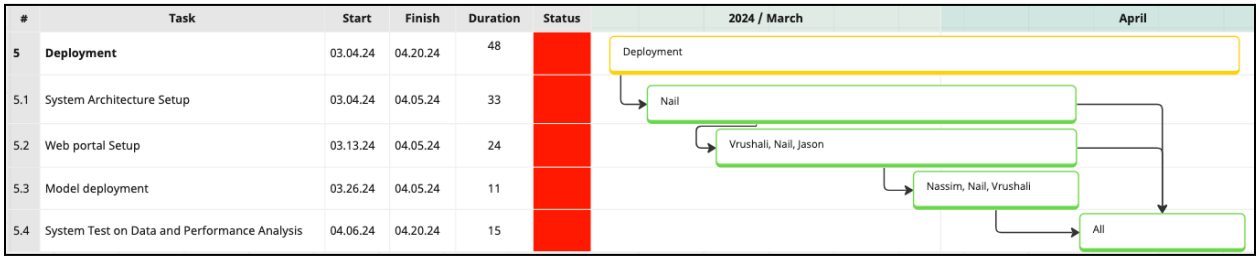


Figure 6

Gantt Chart - Phase 5, Deployment



The PERT chart helps to determine the shortest time needed to finish all tasks by identifying the critical route of the project. Red arrows indicate the critical path.

Figure 7

PERT Chart - Path between phases 1 and 2

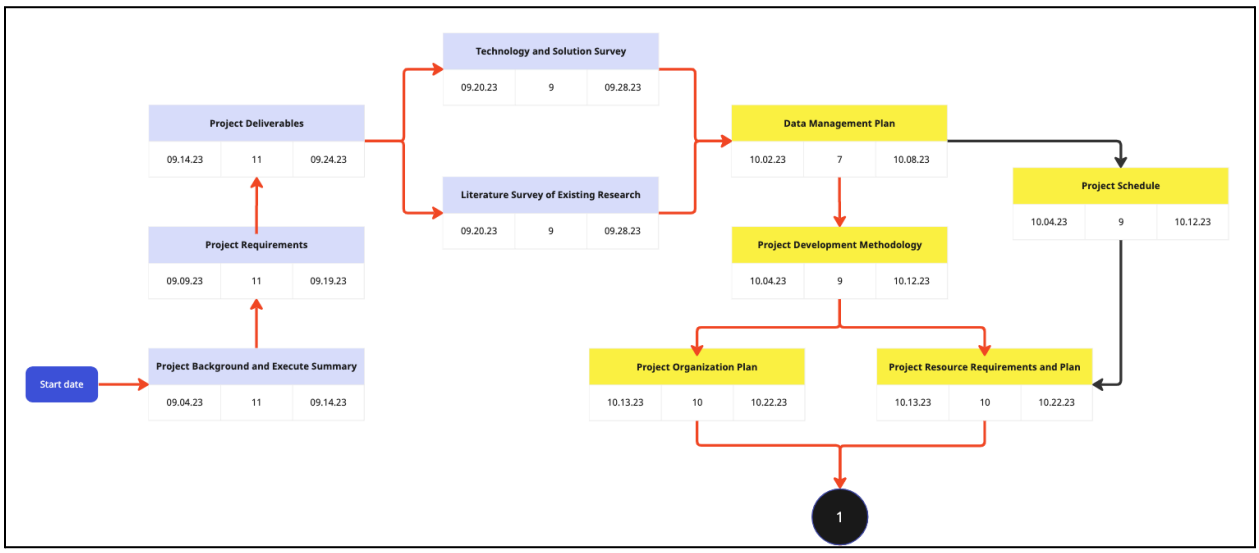


Figure 8

PERT Chart - Path between phases 3 and 4

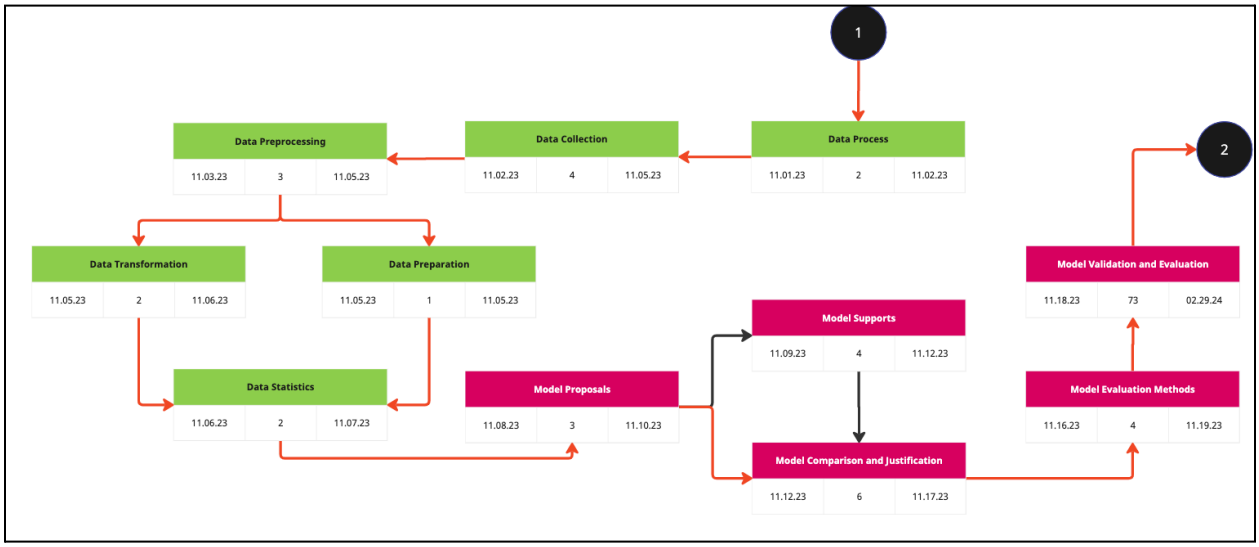
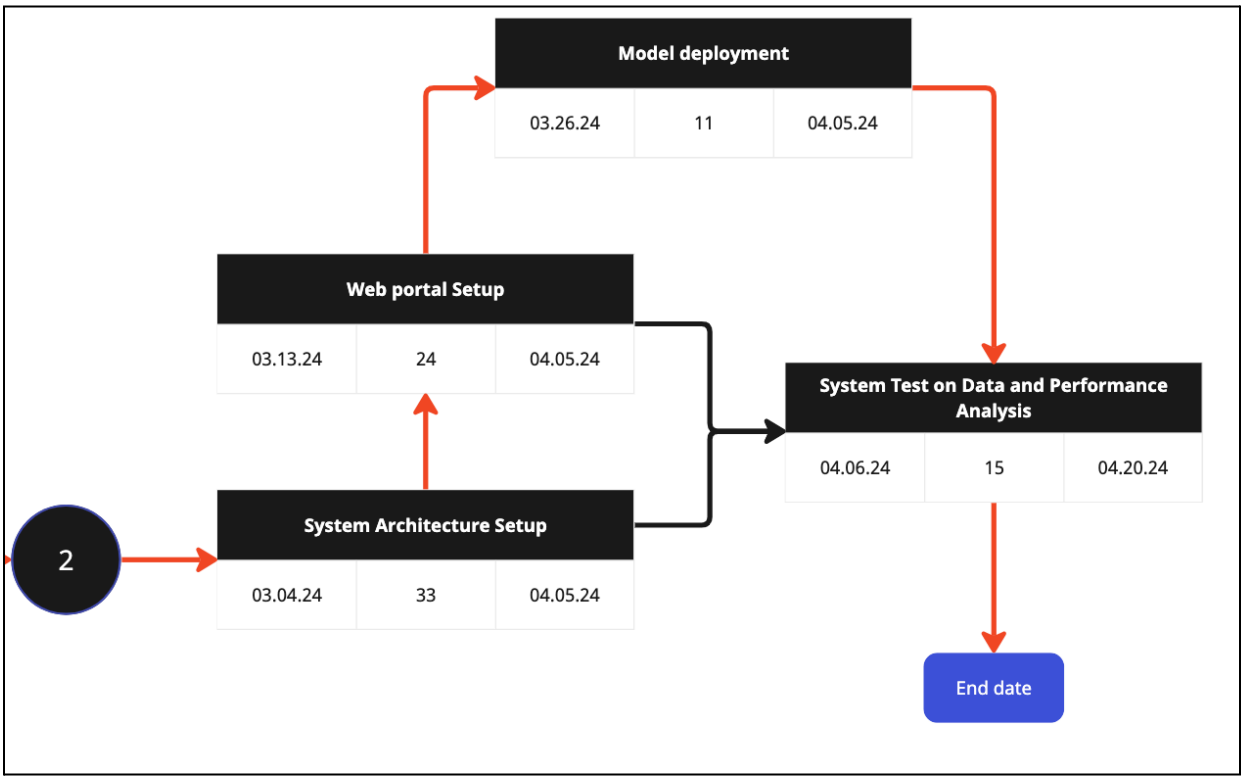


Figure 9

PERT Chart - Path between phases 4 and 5



3. Data Engineering

3.1 Data Process

In order to perform water-intensive crop disease classification, the collection of image data on the different crop plant diseases as well as the healthy crop plant must be done. This project will use rice as a water-intensive crop subject, consequently image data of diseases which affect rice plants are needed. The image datasets used in the project are taken from the 'Rice Leaf Diseases in Taiwan' Kaggle Dataset. The dataset contains labeled images of healthy, leaf blast and brown spot classes. The images in the dataset are geotagged meaning that for each labeled image contains metadata which tells the time and location in the form of latitude and longitude it was taken. API's can work with the metadata information and provide remote sensing data which is utilized to further enhance the machine learning process of the project allowing the model to incorporate both visual image data as well as remote sensing data to possibly make better classifications.

After collection, the initial step is to perform data pre-processing for data cleaning. This can include resizing the image files to ensure compatibility with the proposed models to be used later on as well as removing images which lack clarity and removing duplicates. Data transformation and preparation such as image augmentation and splitting into training, validation and test sets will be done to the processed dataset to prepare the dataset to be fed through visual models such as Convolutional Neural Networks (CNN) or even Pre-Trained CNNs. The Pre Trained CNNs such as VGG19 may require the image data to be fed through in a specific format such as an input dimension of 224x224 pixels and three color channels (RGB). For application, the trained model will then be able to classify crop images taken from the drone as it is flying through the field. This project will also look into incorporating other information such as weather

data and remote sensing information on top of the image data collected into other machine learning models. Comparisons will be made between visual only machine learning models and the hybrid models which incorporate both visual and tabular information to see if the hybrid models could potentially perform better than its counterpart. The images are classified by the model, based on the disease classification and relative to where the image was taken with respect to the entire field can be produced showing the different diseases present on each grid in the field. Farmers will then be able to utilize the data provided to be able to detect and handle crops with diseases in their fields.

3.2 Data Collection

The 'Rice Leaf Diseases in Taiwan' Kaggle Dataset is selected as the image data source for this project. It contains a combined total of 927 labeled images of Rice Diseases. The classes in this dataset are Healthy, Brown Spot and Leaf Blast which are among the most common diseases in rice crops. As mentioned earlier, this dataset in particular contains metadata information for most of the images which gives information about the time and location for each labeled image. The API's can fetch weather data and remote sensing data based on the time and location of each image metadata collected which enhances and enriches the initial visual dataset into a dataset which contains both image and tabular data. The class distribution of the initial dataset can be seen in Table 3.

Table 3

Initial Class Distribution for Rice Leaf Disease in Taiwan Dataset

Rice Leaf Diseases in Taiwan Dataset Class Distribution		
Rice Blast: 519 images	Healthy: 316 images	Brown Spot: 92 images

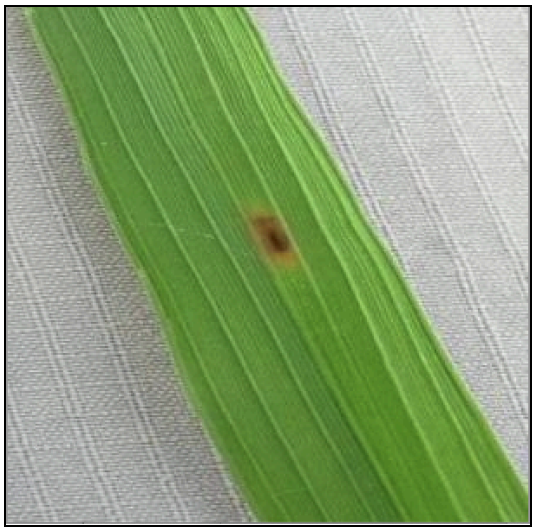
As stated earlier, both images of healthy rice plants and images of rice plants with different diseases will both be needed for the model to be able to learn the patterns and later distinguish and classify a rice disease if present. Common rice diseases such as brown spot, and rice blast are selected as the disease cases for the classification task. Figure 10 shows an example of the image for each class in the rice disease dataset. Another thing to note is that images in this dataset are full plant images when compared to other datasets which usually use a singular leaf image as shown in Figure 11. For the purposes of this project full plant images which are provided in the selected dataset is what is ideal for the model to be trained upon as application envisioned is that a drone would be flying on top of the rice field taking images of whole plants and not singular plant images with white background. Doing so, simulation of gathered data in an end-use case scenario is possible.

Figure 10

Sample Image for each Class in Dataset

Figure 11

Singular Leaf Image Example



Having collected the image dataset, a script was run to collect the metadata for each image in the dataset collected. Time and location information was collected and fed into API's for weather and remote sensing information. The weather data was fetched using the Visual Crossing API and remote sensing data was fetched using the Google Earth Engine API. Weather is well known to be a factor in the occurrence of certain diseases in crops this is why this information was fetched and fed through to the model later on. Remote sensing data refers to the hyperspectral reading based on the emitted or reflected energy. Satellites are able to detect changes in green vegetation using remote sensing data and are calculated in the form of Normalized Difference Vegetation Index (NDVI) as well as Landsat Enhanced Vegetation Index (EVI). To be able detect change in vegetation indices previous NDVI and EVI values will also be collected to observe for changes. Figure 12 below shows the combined tabular information for weather and remote sensing information for each image in the dataset. The tabular data contains information such as temperature, humidity, precipitation, NDVI, EVI along with NDVI and EVI values of previous cycles.

Figure 12

Tabular Data

	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS
0	P_20181227_153331_vHDR_Auto.jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:31	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
1	P_20181227_153343_vHDR_Auto (1).jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:43	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
2	P_20181227_153711_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:11	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
3	P_20181227_153709_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:09	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
4	P_20181227_154446_vHDR_Auto (1).jpg	24.074350	120.661598	2018-12-27	Brown Spot	2018:12:27 15:44:46	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328

3.3 Data Pre-processing

Data cleaning was done on the initial compiled image dataset to ensure that only raw images are kept in the initial data. Initial steps are to check for duplicates and remove them as it will improve the model efficacy and prevent bias. To do this, the image's hash is compared and images with identical hash are removed. As the remote sensing and weather data was needed, images with incomplete metadata (missing time or location or both) are removed and will not be used for the machine learning model.

Furthermore, the image dataset contains images of varying sizes. This will be an issue if the images were to be used to be trained on Pre-Trained Convolutional Neural Networks which have been trained on large datasets with specific input image dimensions. Commonly used Pre-Trained CNNs such as VGG19 and Residual Networks (ResNet) have an input dimension requirement of 224x224 pixels with three color channels (RGB). This also means that grayscale images are to be removed. To prepare the image dataset to be trained through the CNNs, each image in the dataset is resized to the specific input dimension requirement of the CNNs, for instance in the case of VGG19, 224x224 pixels. Some of the raw images were of fairly high resolution (such as 4000x3000) and far exceeded the input sizes required for the pre-trained models. Directly resizing these images to smaller dimensions can result in significant loss of detail and distortion of the aspect ratio. To mitigate this, we utilized 'tf.image.resize_with_pad',

a function that resizes the images while maintaining the original aspect ratio by putting padding to the areas that do not contribute to the distortion of the content's images. This step is of paramount importance for preserving the fidelity of the image's details. During the resizing we also employed antialiasing, a process which smoothens the image to help prevent the kind of jaggedness that occasionally occurs when higher-resolution images are scaled down. Similar to the padding, this step is important to preserve the quality of the image's features, which are vital when training the convolutional neural network models. Figure 13 shows an example of the resized image from the original data source to the target size of 224x224 pixels. Table 5 shows the class distribution of the combined dataset post pre-processing stage.

Figure 13

Original vs Resized 224 x 224 Image

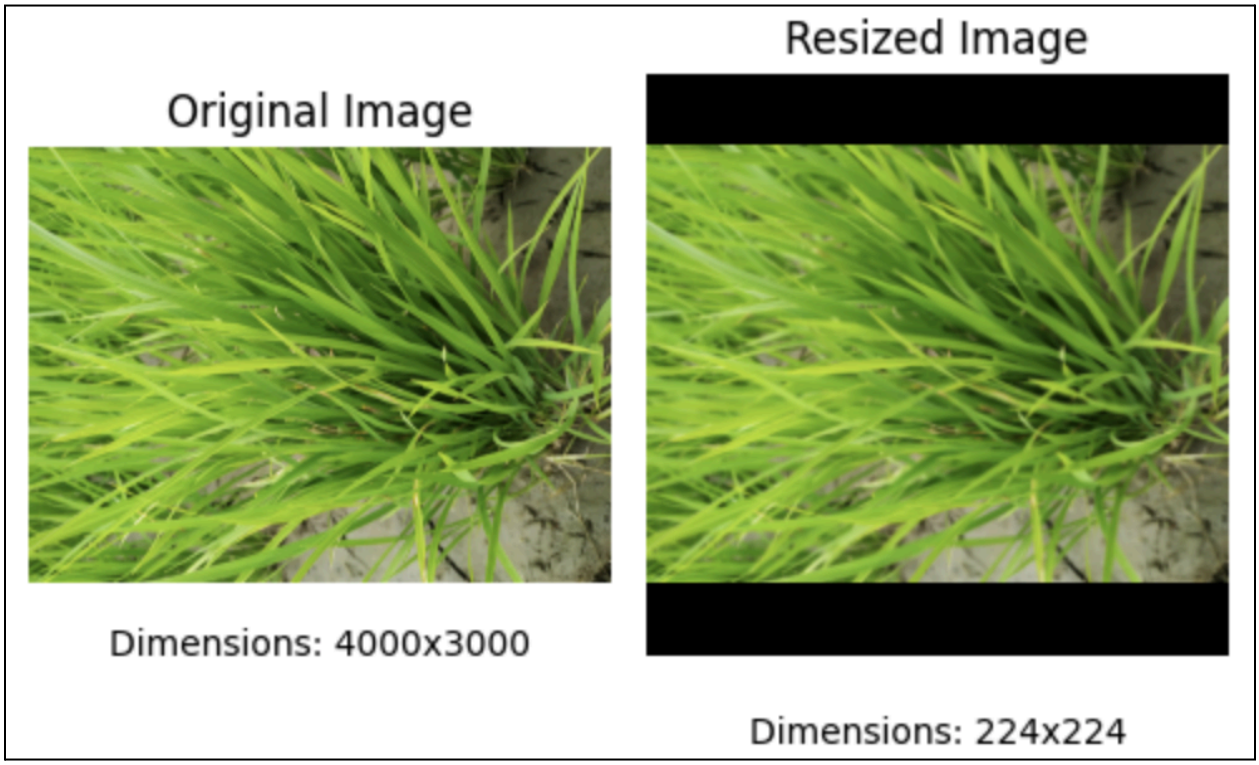


Table 4

Combined Dataset Post Preprocessing

Combined Dataset (Post Pre-Processing)	
Healthy Rice Plant: 306 images	Image Dimension: 224x224 pixels Color Channel: RGB
Rice Blast: 448 images	
Brown Spot: 75 images	

3.4 Data Transformation

Initially, we split the dataset into training, validation, and test splits. The distribution of the target variable was maintained throughout the splitting of the data. To counter the imbalance of the target class, and to increase the size of the data, we employed data augmentation on images in the train set only. Augmentation was not performed on the validation and test sets, since we want the data that the model is being evaluated on to be as close to real-world conditions as possible. Augmenting the validation and test sets may lead to an artificially inflated performance of the model. Data augmentation on the train set creates diverse sets of data, which may help the model to generalize better to a wider array of unseen images. The distribution of the classes in the train set before augmentation is shown in Figure 14. Figure 15 shows the distribution of the classes after the image augmentation.

Figure 14

Distribution of Classes in Train Dataset Before Augmentation

```
# Distribution of the train dataset before augmentation
train_df['Class'].value_counts()
```

```
Rice Blast    313
Healthy       214
Brown Spot     53
Name: Class, dtype: int64
```

Figure 15

Distribution of Classes in Train Dataset After Augmentation

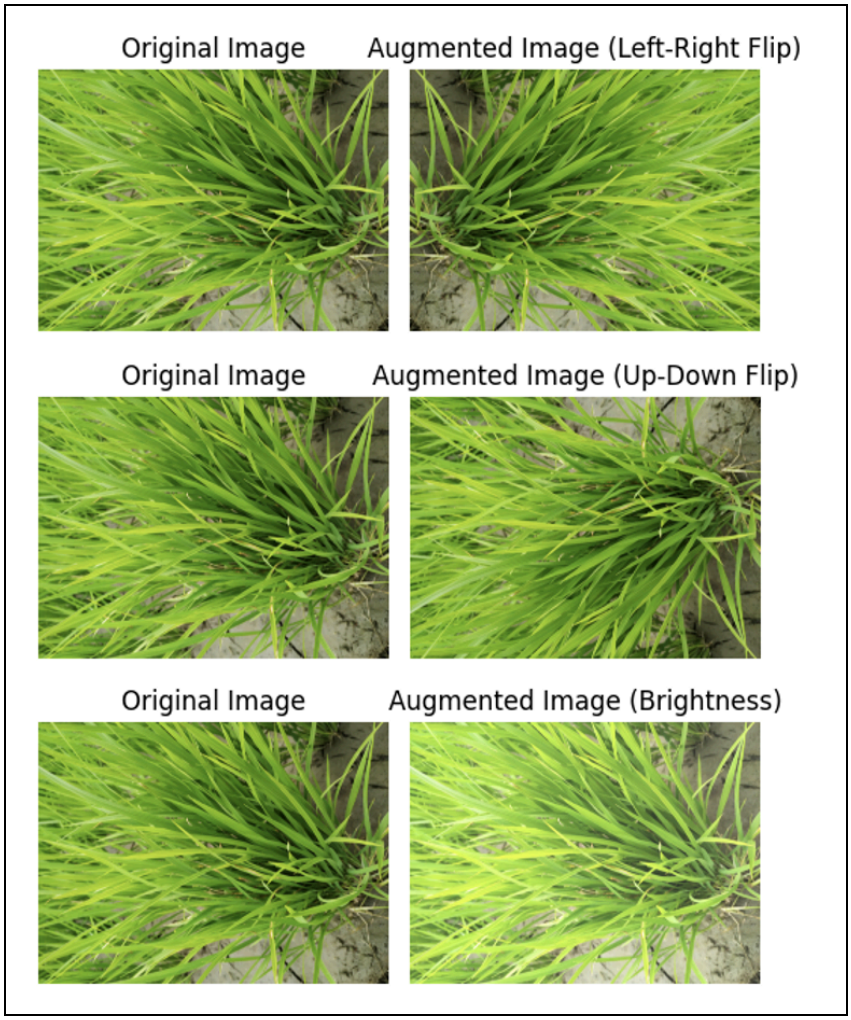
```
# Distribution of the train dataset after augmentation
train_df_with_augmentation_rice['Class'].value_counts()
```

```
Healthy        400
Rice Blast     400
Brown Spot     400
Name: Class, dtype: int64
```

The specific augmentations that were performed were horizontal and vertical rotations and brightness changes. The rotations were employed to make the model more robust to changes in rotation and to generalize better to a wider array of images. For the brightness changes, the images were randomly brightened or darkened with a maximum percentage change of 30. The purpose of this augmentation is to simulate various lighting conditions that may occur naturally due to environmental changes, for instance, due to the time of day. This augmentation technique helps the model generalize better to images with a variety of brightness levels. An example of each type of augmentation technique on a sample image is shown below:

Figure 16

Image Augmentation Examples



After transformation of the visual data is completed, further transformation for the tabular data still needs to be done before feeding both forms of data into the machine learning model. For the tabular data, the weather data was standardized using the StandardScaler function to transform all the numerical values to be in the same scale which will ensure no bias is introduced to the natural magnitude of the data. Binary indicators were added to the remote sensing data which shows whether there was a decrease from the previous 2 periods which might be indicative of disease activity in the crop. Figure 17 shows the sample of the standardized tabular data with binary indicators added to the NDVI and EVI fields.

Figure 17

Standardized Tabular Data with Binary Indicators

	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS	NDVI 1 Decrease	NDVI 2 Decrease	EVI 1 Decrease	EVI 2 Decrease
1	P_20181227_153343_vHDR_Auto (1).jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:43	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
2	P_20181227_153711_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:11	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
3	P_20181227_153709_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:09	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
5	P_20181227_155134_vHDR_Auto.jpg	24.074811	120.660849	2018-12-27	Brown Spot	2018:12:27 15:51:34	-0.723365	0.060275	-0.868913	1.263967	0.5403	0.3980	0.3624	0.4185	0.2271	0.2348	0	0	0	0
6	P_20181227_154452_vHDR_Auto_HP (1).jpg	24.074337	120.661612	2018-12-27	Brown Spot	2018:12:27 15:44:52	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0

TensorFlow's 'Dataset' API was used to transform the image and numerical datasets into a format suitable for training in TensorFlow, using the 'from_tensor_slices' function. During the conversion, a pipeline is utilized to convert the datasets into a format compatible with the pre-trained model in question. For instance, a 'preprocess_input' function of the VGG module in TensorFlow is used to standardize the pixel values to a format the pre-trained model expects. The 'Dataset.zip' function is used to combine the image and numerical datasets to ensure that the models receive both kinds of input. The training set was shuffled to make sure that the order of the images does not introduce bias or irrelevant pattern learning in relation to the actual content of the images. The training, validation, and test sets were batched and cached to increase computational efficiency and the convergence speed of the model.

3.5 Data Preparation

The data is split into training, validation and testing datasets in the split of 70-15-15 for the model training and evaluation processes. As mentioned previously, the image training dataset was the only set that was augmented to increase the diversity and to reduce the imbalance of the image training set, and to increase the generalizability of the model. The distribution of the image train sets before and after augmentation are shown in Figures 14 and 15, respectively. The image training, validation, and test datasets are converted to a format compatible with TensorFlow, and are further pre-processed to be compatible with the pre-trained models. As

such, the state of the images at this point is not meant to be presented in a visual format, and are instead only stored in a format that is only meant for the training process of the model.

For the numerical data, Figures 18 - 20 show the first few rows of the datasets.

Figure 18

Sample Numerical Train Dataset

```
train_df.head()
```

	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS	NDVI 1 Decrease	NDVI 2 Decrease	EVI 1 Decrease	EVI 2 Decrease
1	P_20181227_153343_vHDR_Auto (1).jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:43	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
2	P_20181227_153711_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:11	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
3	P_20181227_153709_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:09	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
5	P_20181227_155134_vHDR_Auto.jpg	24.074811	120.660849	2018-12-27	Brown Spot	2018:12:27 15:51:34	-0.723365	0.060275	-0.868913	1.263967	0.5403	0.3980	0.3624	0.4185	0.2271	0.2348	0	0	0	0
6	P_20181227_154452_vHDR_Auto_HP (1).jpg	24.074337	120.661612	2018-12-27	Brown Spot	2018:12:27 15:44:52	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0

Figure 19

Sample Numerical Validation Dataset

```
val_df.head()
```

	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS	NDVI 1 Decrease	NDVI 2 Decrease	EVI 1 Decrease	EVI 2 Decrease
0	P_20181227_153331_vHDR_Auto.jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:31	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
9	P_20181227_155141_vHDR_Auto_HP.jpg	24.074811	120.660849	2018-12-27	Brown Spot	2018:12:27 15:51:41	-0.723365	0.060275	-0.868913	1.263967	0.5403	0.3980	0.3624	0.4185	0.2271	0.2348	0	0	0	0
11	P_20181227_155149_vHDR_Auto.jpg	24.074811	120.660849	2018-12-27	Brown Spot	2018:12:27 15:51:50	-0.723365	0.060275	-0.868913	1.263967	0.5403	0.3980	0.3624	0.4185	0.2271	0.2348	0	0	0	0
19	P_20181227_155406_vHDR_Auto_HP.jpg	24.074484	120.660750	2018-12-27	Brown Spot	2018:12:27 15:54:06	-0.723365	0.060275	-0.868913	1.263967	0.5403	0.3980	0.3624	0.4185	0.2271	0.2348	0	0	0	0
34	P_20181227_160858_vHDR_Auto_HP.jpg	24.078341	120.650942	2018-12-27	Brown Spot	2018:12:27 16:08:58	-0.723365	0.051357	-0.868913	1.263967	0.3823	0.4466	0.3852	0.3083	0.3513	0.2857	1	1	1	0

Figure 20

Sample Numerical Test Dataset

```
test_df.head()
```

	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS	NDVI 1 Decrease	NDVI 2 Decrease	EVI 1 Decrease	EVI 2 Decrease
4	P_20181227_154446_vHDR_Auto (1).jpg	24.074350	120.661598	2018-12-27	Brown Spot	2018:12:27 15:44:46	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
7	P_20181227_154500_vHDR_Auto_HP.jpg	24.074337	120.661612	2018-12-27	Brown Spot	2018:12:27 15:45:00	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
8	P_20181227_154449_vHDR_Auto.jpg	24.074350	120.661598	2018-12-27	Brown Spot	2018:12:27 15:44:49	-0.723365	0.060275	-0.868913	1.263967	0.3160	0.3335	0.2184	0.2176	0.1857	0.1328	1	0	0	0
18	P_20181227_155333_vHDR_Auto_HP.jpg	24.074484	120.660750	2018-12-27	Brown Spot	2018:12:27 15:53:33	-0.723365	0.060275	-0.868913	1.263967	0.5403	0.3980	0.3624	0.4185	0.2271	0.2348	0	0	0	0
20	P_20181227_155352_vHDR_Auto.jpg	24.074484	120.660750	2018-12-27	Brown Spot	2018:12:27 15:53:53	-0.723365	0.060275	-0.868913	1.263967	0.5403	0.3980	0.3624	0.4185	0.2271	0.2348	0	0	0	0

The validation dataset is utilized during the training process to help prevent overfitting of the model. An early stopping mechanism is implemented which monitors the performance of the model on the validation set, and prevents further training of the model when the performance on the validation set begins to degrade.

3.6 Data Statistics

Below are the data statistics and visualizations made to show the distribution of the dataset. Figures 21-26 show the different distribution of the raw, preprocessed, and split datasets along with the distribution of the training set before and after image augmentation.

Figure 21

Raw Dataset Distribution

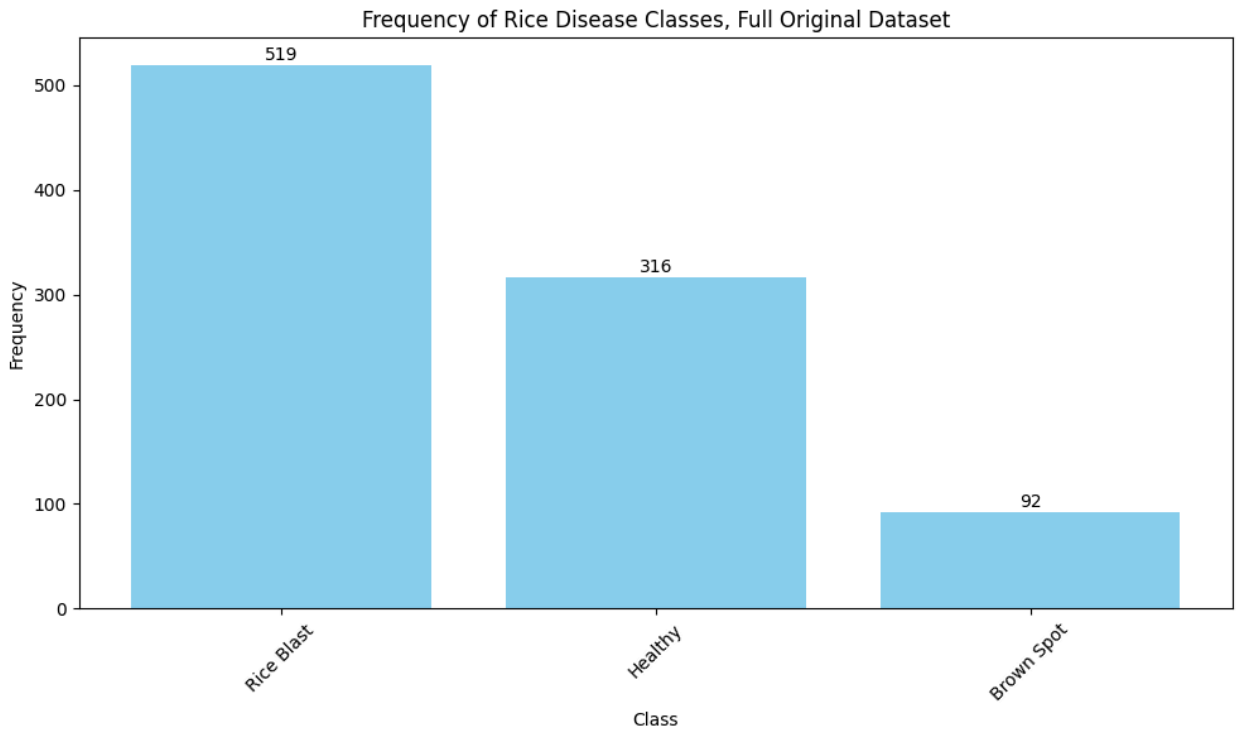


Figure 22

Distribution after removing duplicates

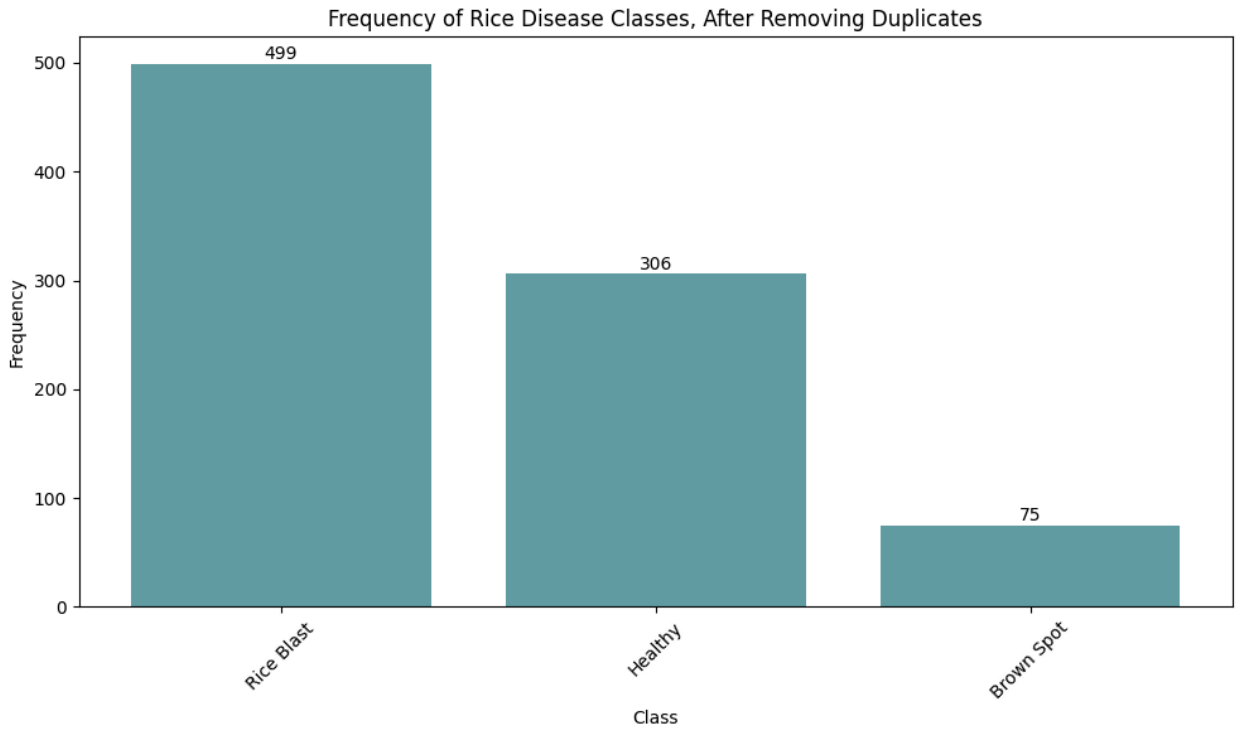


Figure 23

Distribution after filtering time and location data

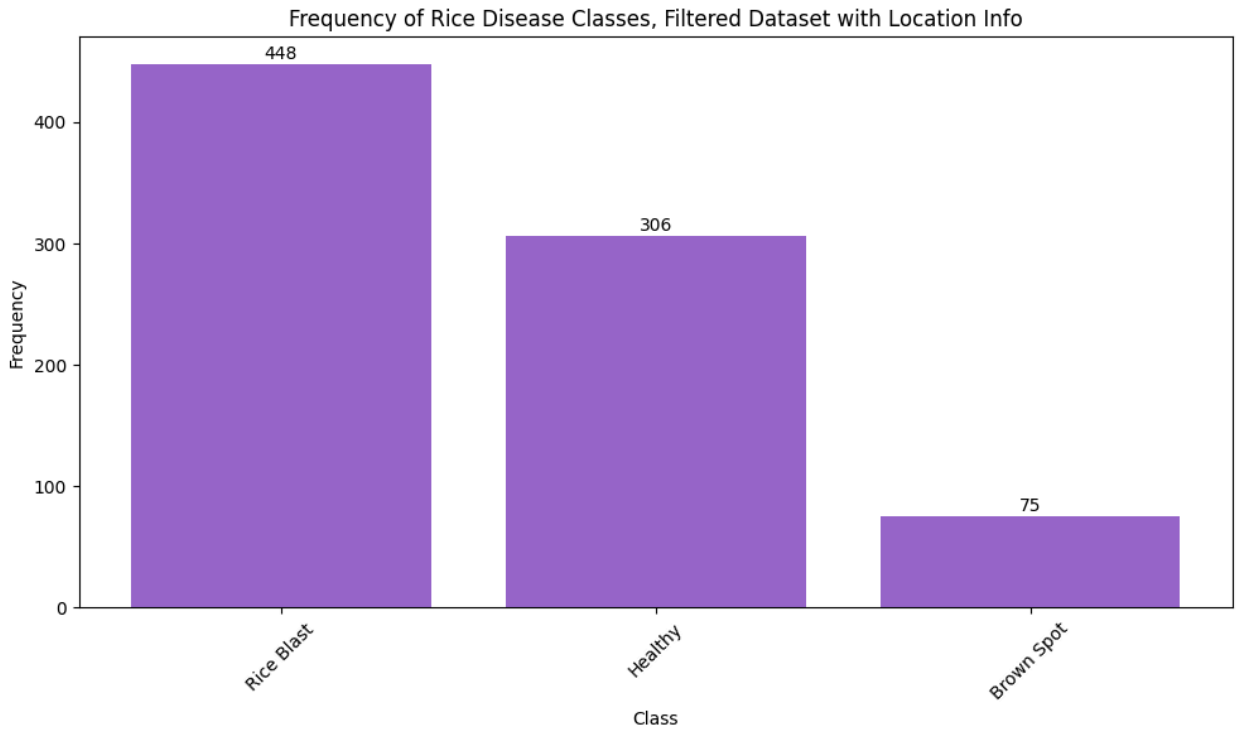


Figure 24

Train Distribution before Augmentation

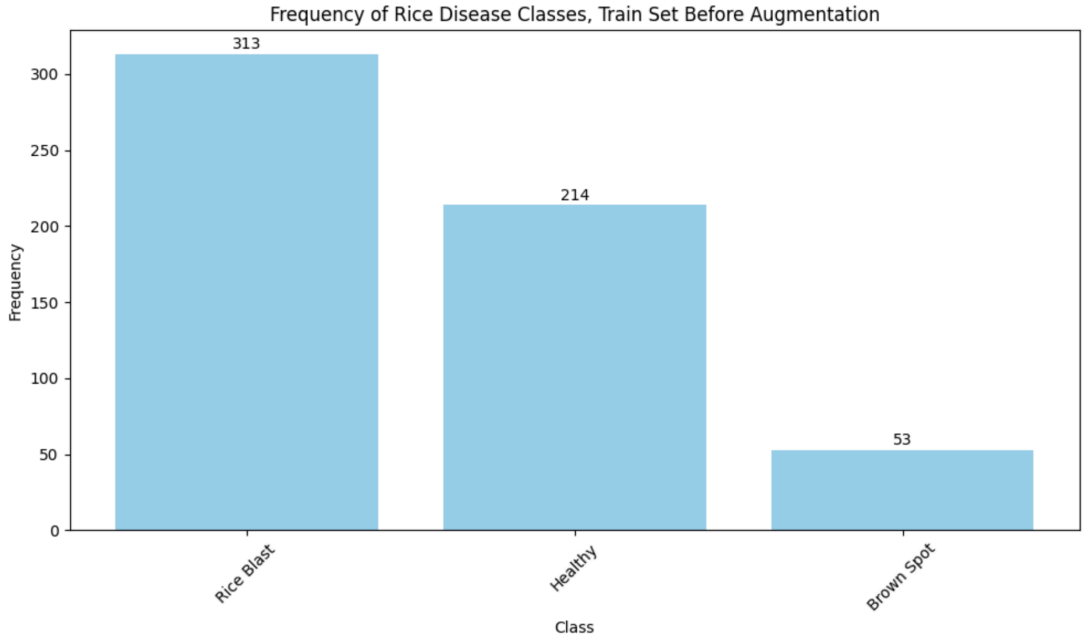


Figure 25

Train Distribution after Augmentation

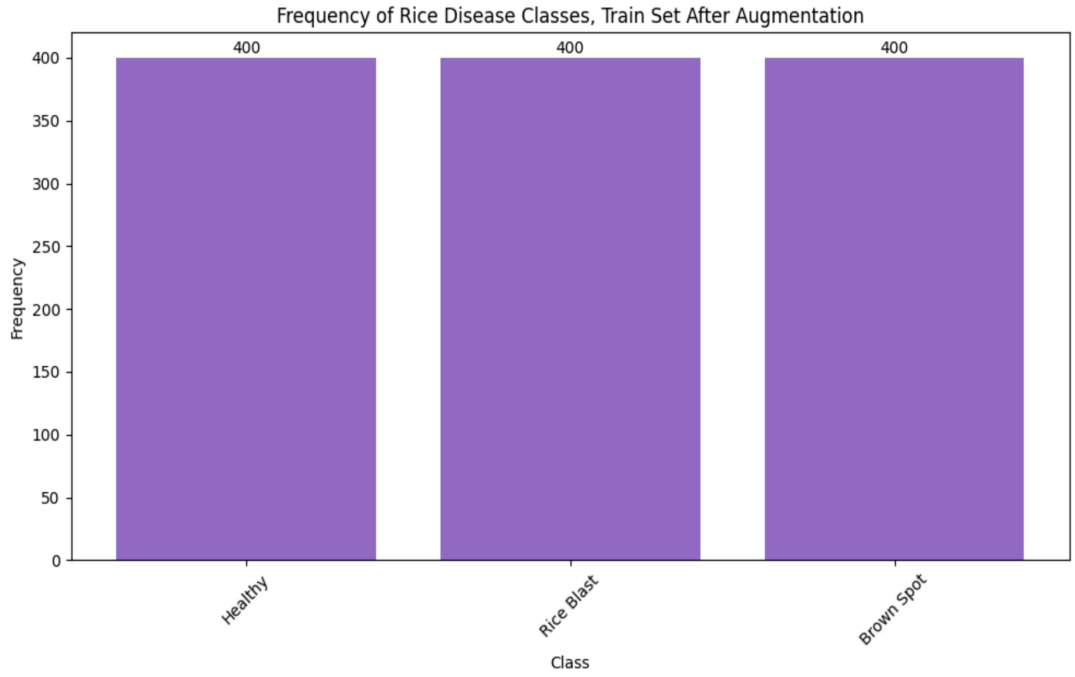


Figure 26

Distribution of Validation and Test Sets

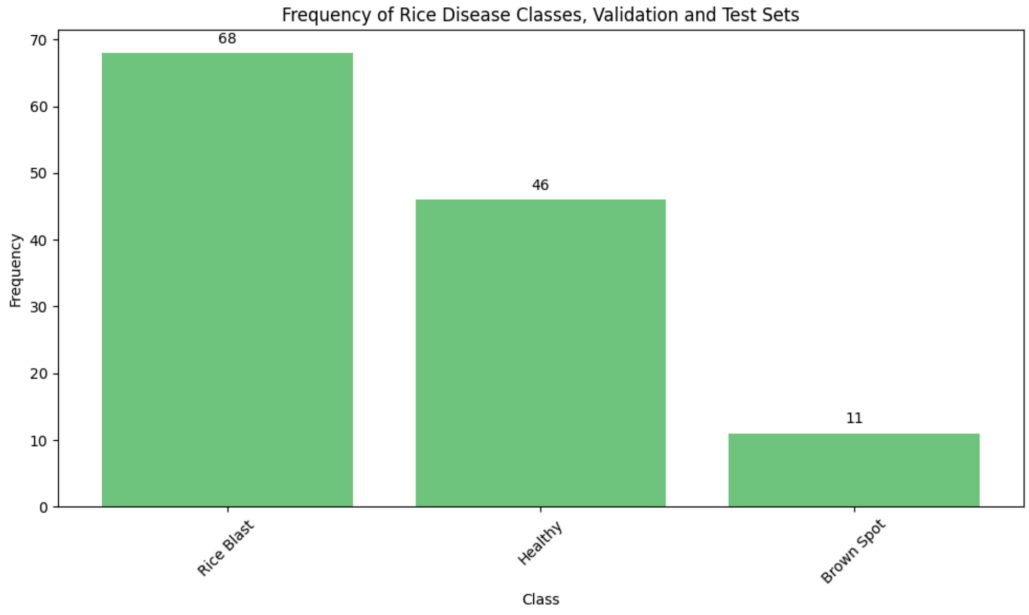


Figure 27

Statistics for tabular data

```
combined_data.head()
```

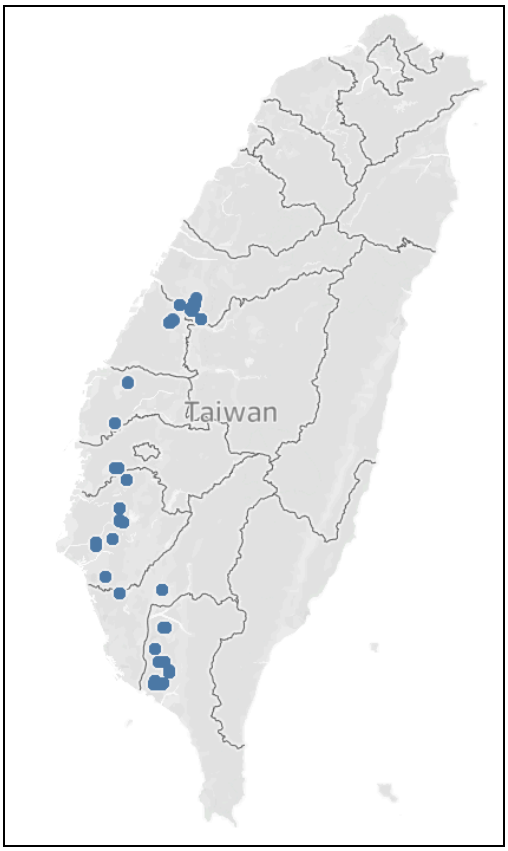
	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS
0	P_20181227_153331_vHDR_Auto.jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:31	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
1	P_20181227_153343_vHDR_Auto (1).jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:43	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
2	P_20181227_153711_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:11	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
3	P_20181227_153709_vHDR_Auto.jpg	24.073297	120.661364	2018-12-27	Brown Spot	2018:12:27 15:37:09	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328
4	P_20181227_154446_vHDR_Auto (1).jpg	24.074350	120.661598	2018-12-27	Brown Spot	2018:12:27 15:44:46	19.328571	76.664286	5.7	29.171429	0.316	0.3335	0.2184	0.2176	0.1857	0.1328

```
combined_data.describe()
```

	Latitude	Longitude	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS
count	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000	829.000000
mean	23.189876	120.502923	21.152550	76.597527	29.263478	20.242047	0.451565	0.405025	0.411362	0.284332	0.252975	0.251676
std	0.660420	0.128460	1.960557	2.665656	21.979152	4.591261	0.145848	0.123150	0.130431	0.108621	0.086300	0.092022
min	22.501092	120.219323	18.457143	72.257143	1.900000	13.742857	0.152000	0.170900	0.171600	0.095900	0.078400	0.097700
25%	22.515942	120.485031	19.307143	73.842857	15.231000	16.892857	0.323000	0.312000	0.275500	0.166900	0.189700	0.169400
50%	23.097482	120.487786	20.721429	76.078571	27.064000	17.728571	0.369300	0.370100	0.377900	0.268700	0.227800	0.236600
75%	24.031891	120.650942	22.764286	78.492857	28.733000	25.378571	0.591000	0.502100	0.543400	0.364500	0.301600	0.350300
max	24.119848	120.696254	27.450000	83.721429	273.800000	29.171429	0.711500	0.715400	0.603200	0.508100	0.499400	0.403400

Figure 28

Dataset Source Locations


Taiwan

4. Model Development

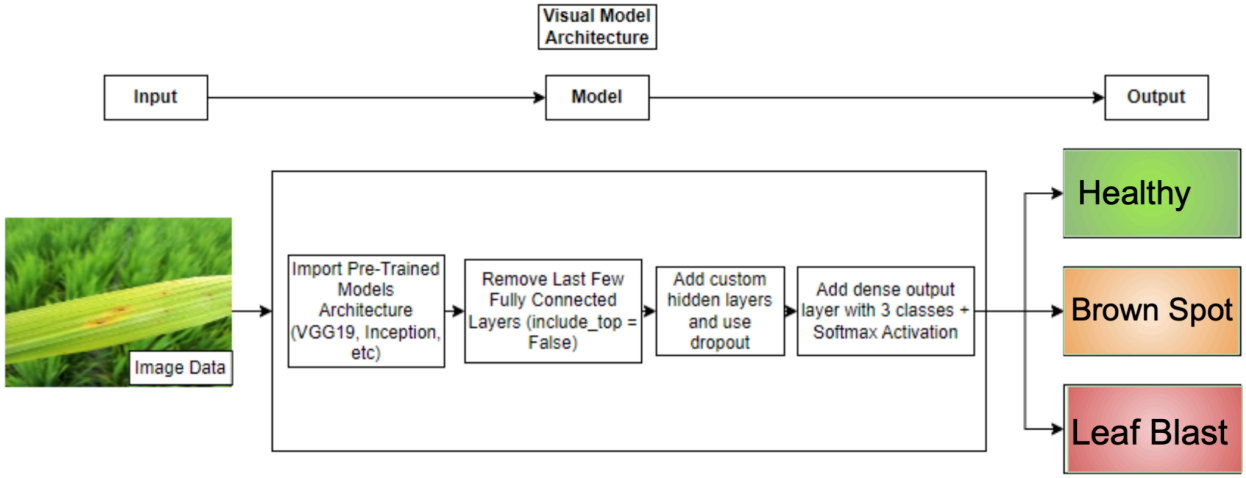
4.1 Model Proposals

This project will utilize a combination of multiple data sources to create a more robust model for the purpose of Rice Disease Classification. Visual data is gathered from a Kaggle Dataset, this dataset contains labeled images of rice leaf diseases in Taiwan. Images in the dataset are geotagged where location information in the form of latitude and longitude are stored as well. Combine location data with the date and time when the image was taken from the metadata of the image, weather information such as temperature, humidity and remote sensing data such as NDVI from satellites can be extracted from readily available APIs.

Transfer learning is to take a neural network that has been already trained on a quality large-size dataset, that already achieved a decent level of feature extraction with established weights, and fine-tune this model for the related problem using the crop disease image datasets. In this project four pre-trained models are used: VGG-19, ResNet-50, Inception V3, and DenseNet121. The fine-tuning process updates the weight of the pre-trained model to fit the crop disease image dataset. There are several benefits of such an approach. The first is efficiency, which means that neural networks don't need large amounts of labeled data for fine-tuning. Second, since the model has already learned features at depth before fine-tuning, this speeds up the training process on the crop disease image dataset. Third, such an approach typically leads to a much better performance of the model. Transfer learning is prevalent in areas such as computer vision, image recognition, and natural language processing. This will then be added with other features from temperature as well as remote sensing to create a hybrid model which takes into account other factors besides a purely visual model.

Figure 29

Visual Model Architecture



4.1.1 VGG-19

VGG-19 is a neural network architecture which is an improved deeper version of the base VGG-16 architecture. VGG-19 is showing robust performance in image classification tasks. According to a tomato leaf disease classification research paper by (Nguyen, 2022), the fine-tuned VGG-19 model achieved 99.72% accuracy in disease classification, which is the highest value among other fine-tuned experimented models, such as GoogleNet, ResNet50, and AlexNet. The model introduces 143 million parameters.

The neural network has three more convolutional layers and represents 19 layers in total in comparison to 16 total layers in VGG-16. Thus, In VGG-19 there are 16 convolutional (CNN) and 3 fully connected layers, and each of them has weights. The purpose of convolutional layers is to extract features and learn the relationships between them. In terms of CNN, more layers increase the ability of the model to extract valuable features and catch more complex relationships between them. Convolutional layers use small 3x3 filters with a stride equal to 1. This is the smallest size of the filter that is still able to capture details. In addition, between the

convolutional layers, there are max-pooling layers that have 2x2 size and stride equal to 2. The purpose of max-pooling layers is to reduce the spatial dimensions of the feature maps and prevent overfitting (Stephen, 2023). The input model takes a fixed-size 224x224 image and 3 channels (RGB). The set of CNN layers starts with 64 channels and doubles the number after each max-pooling layer. The architecture is followed by three fully connected and one softmax layer. The first two fully connected layers have 4096 channels, and the third one has 1000 channels. The softmax layer, as an output, represents the probabilities of the input for each class. The model architecture is shown in Figure 30.

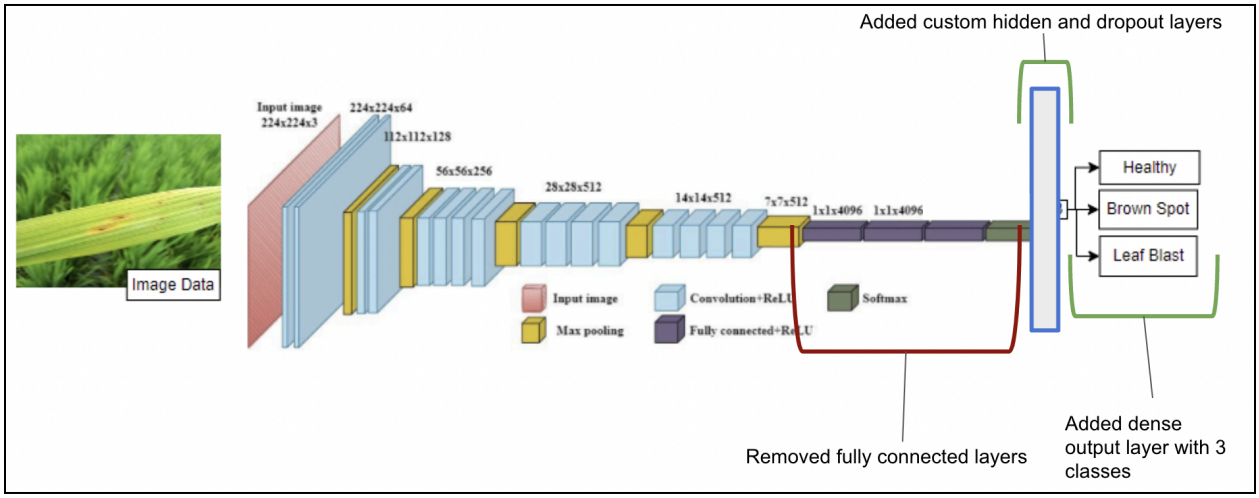
After each convolutional and fully connected layer, the Rectified Linear Unit (ReLU) activation function is used. The ReLU outputs positive input, or zero in case of negative input values. ReLU provides non-linearity to the model, which is important for catching complex relationships. Also, ReLU in comparison to other activation functions, like tanh and sigmoid, is computationally efficient due to its simplicity. ReLU decreases the likelihood of vanishing gradient, which is a common problem in deep learning networks. The disadvantage of ReLU is that if during the training neurons output values in the negative range, ReLU will assign zero, switching off the neurons. This problem is known as “dying ReLU”. To overcome this problem it is suggested to experiment with the Leaky ReLU activation function, which assigns a small non-zero value when a neuron outputs a negative value. However, experimenting with Leaky ReLU is not considered for the current project.

For handling multi-class classification problems, in the VGG-19 architecture the categorical cross-entropy loss function is used. Categorical cross-entropy loss decreases as the predicted probability tends to match the ground truth label. If the data and model are prepared correctly, during the training process categorical cross-entropy loss is expected to be minimized

over the epochs. The categorical cross-entropy loss is the perfect fit for VGG-19 since the softmax layer converts raw logits to probabilities, providing a native output format for the loss function.

Figure 30

VGG-19 architecture



Note: Adapted from "A VGG-19 Model with Transfer Learning and Image Segmentation for Classification of Tomato Leaf Disease," by T-H. Nguyen, T-N. Nguyen, & B-V. Ngo, 2022, *AgriEngineering*, 4(4), 871-887. (Modified) https://doi.org/10.3390/agriengineering4040056

As mentioned before, VGG-19 provides great performance for image classification. The depth of the neural network allows the model to learn complex patterns and extract more accurate feature maps.

Using 19 layers in comparison to 16 in VGG-16 is more computationally expensive. This is the cost of extraction of more complex features and thus, increased performance of the model. The model also consumes a significant amount of memory for the training process. Also, deeper learning leads to better generalization and increases the risk of overfitting. These are disadvantages that must be considered while implementing the project.

After the softmax layer converts logits into probabilities, the model provides clear and interpretable outputs to perform multi-classification over the four classes in the project dataset.

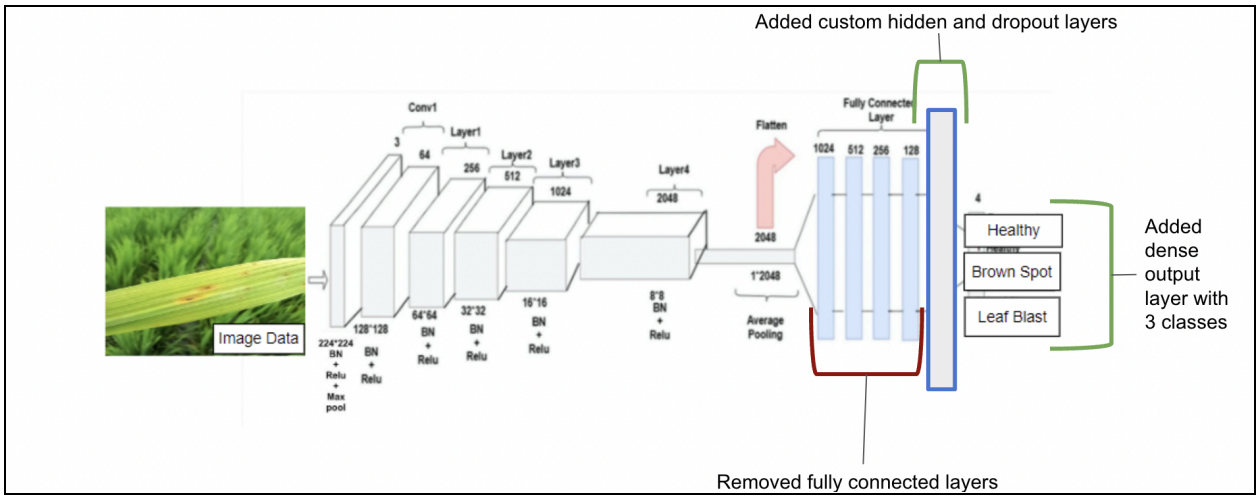
4.1.2 ResNet50

ResNet-50 is a version of the widely known and used ResNet architecture. It presented a new approach to building deeper neural networks. The model is a common choice for multi-class image classification problems. It has around 25 million parameters as a result of utilizing a unique architecture that includes residual blocks and skip connections.

The main concept of the model is deep residual learning, which is developed to address the vanishing gradient problem in deep learning. ResNet-50 has fifty layers, including convolutional, pooling, and fully connected layers. As in many cases of neural networks, the model starts with a single convolutional layer with max-pooling. Max-pooling reduces the spatial size of the input for further robust feature extraction. The input model takes a fixed-size 224x224 image and 3 channels (RGB). Four bottleneck layers are used to reduce complexity and training time. These layers consist of CNN layers that have a 3x3 size of the kernel. The amount of filters progressively increases holding 64, 256, 512,1024, and 2048 feature map dimensions. The more CNN layers there are, the higher the accuracy for the image classification tasks. There are 48 convolutional layers in total, and each of them is followed by batch normalization and ReLU activation function to provide non-linearity along with computational efficiency. Convolutional layers at the end of the network hold 2048 channels with average polling followed by the fully connected layer, which in turn ends with the softmax function. The softmax function culminates the network outputting probabilities over classes. The model architecture is shown in Figure 31.

Figure 31

ResNet-50 architecture



Note: Adapted from "Designing self attention-based ResNet architecture for rice leaf disease classification," by A. Stephen, A. Punitha, & A. Chandrasekar, 2023, *Neural Computer & Applications*, 35, 6737–6751. (Modified) https://doi.org/10.1007/s00521-022-07793-2

Residual blocks are a core unique element of the architecture. There are 16 residual blocks, each of which utilizes skip connections, which are taking an input, skipping one or more layers, and adding it to the output. Such an approach allows gradients to pass straight through skip connections during the backpropagation; thus performing the training through 48 convolutional layers and avoiding the vanishing gradient problem. The architecture makes it possible to capture and learn even more complex patterns at a fast convergence rate without facing issues that neural networks usually have.

For handling multi-class classification problems, in the ResNet-50 architecture the categorical cross-entropy loss function is used. Categorical cross-entropy loss decreases as the predicted probability tends to match the ground truth label. If the data and model are prepared correctly, during the training process categorical cross-entropy loss is expected to be minimized over the epochs. The categorical cross-entropy loss is the perfect fit for ResNet-50 since the

softmax layer converts raw logits to probabilities, providing a native output format for the loss function.

Even though the model is optimized in terms of memory usage for the training process, it still demands a significant amount of computational resources. In addition, the model has a relatively complex structure compared to simple model architectures. This is the cost of utilizing a deeper network with advanced feature extractions and outstanding classification performance.

4.1.3 DenseNet121

DenseNet121 falls within the family of Dense Convolutional Networks, characterized by its dense feed-forward connections. This means every layer is interconnected with every subsequent layer, creating a network comprising 121 layers in total. These layers include convolutional, fully connected, and max-pooling layers, altogether housing around 7 million parameters.

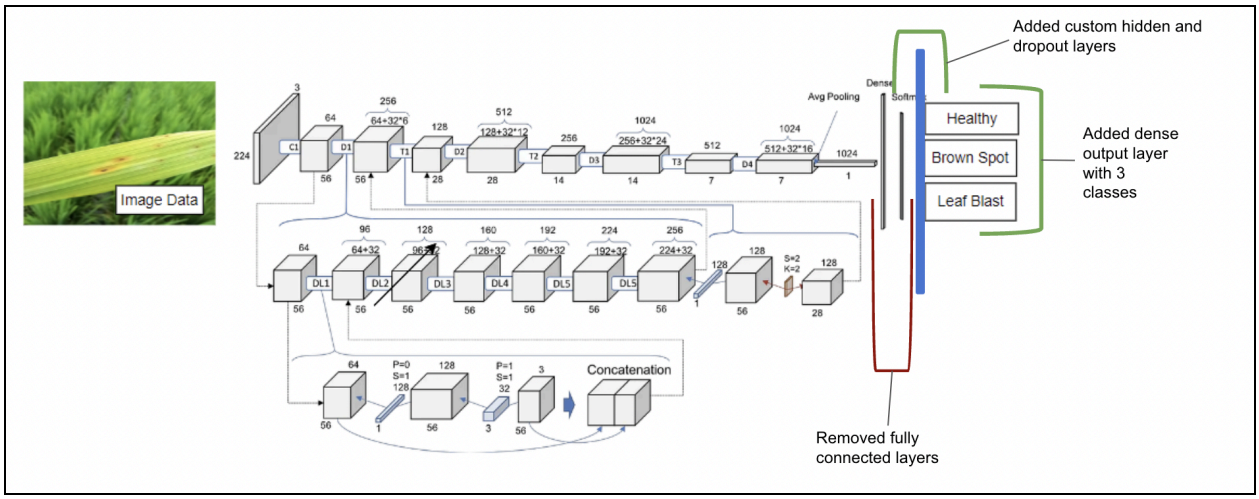
The architecture is built on dense blocks containing multiple layers. Each layer receives additional input from all preceding layers, integrating these inputs into the current layer's feature map, which then feeds into every layer that follows. The contribution of each layer to the overall network's learning state is determined by the Growth Rate - a hyperparameter denoted by “k” - which signifies the number of feature maps each layer adds to the network's total learning state. To manage the network's complexity and computational load, bottleneck layers are employed. These are convolutional layers with a 1x1 filter, succeeded by a 3x3 convolution, aimed at compressing the spatial dimensions of the feature map for subsequent inputs. Transition layers are situated between dense blocks, consisting of batch normalization, a 1x1 convolutional layer, and an average pooling layer to regulate the dimensions of the feature map.

Post each convolutional layer and batch normalization, the network uses the Rectified Linear Unit (ReLU) activation function. ReLU facilitates outputting positive values for positive inputs and zero for negative inputs, introducing non-linearity essential for modeling complex relationships. Its computational efficiency, primarily due to its simplicity, makes it preferable over other activation functions like Tanh and Sigmoid. ReLU also mitigates the common issue of gradient vanishing in deep neural networks.

Following the final dense block, average pooling is applied to condense each feature map to a 1x1 dimension, thereby reducing the parameter count for the ensuing fully connected layer. These layers then relay logits to a softmax activation function, which computes a probability distribution across the classes. The design effectively addresses the vanishing gradient problem by reutilizing features across the network, enhancing gradient flow and making the model both compact and efficient, especially when computational resources are limited. For optimization, DenseNet121 employs the Adaptive Moment Estimation (Adam) algorithm, which adapts individual learning rates for each parameter to facilitate more accurate weight adjustments. To tackle multi-class classification tasks, the model uses the sparse categorical cross-entropy loss function. This choice is optimal for image classification, identifying a single class among multiple classes without necessitating one-hot encoding preprocessing. This approach streamlines computation, an advantage when managing numerous classes.

Figure 32

DenseNet121 architecture



Note: Adapted from

https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a

4.1.4 Inception V3

Inception V3 is a predefined convolutional neural network and is mostly used for image recognition and classification. This model is suitable for water-intensive crop disease detection and classification. Typically, Inception V3 accepts an RGB image with dimensions of 299 × 299 x 3, which is marginally larger than other model's input formats but allows for more detailed input data.

Inception V3's architecture is complex, with numerous layers, but it is best known for its inception modules. These modules are effectively a network within a network, with numerous filter sizes (such as 1x1, 3x3, and 5x5) utilized in parallel within the same layer to gather data at different scales. This approach allows for quick processing and extraction of features from input photos, such as detecting leaf discoloration, pests, or other visual signs of agricultural disease. The inception module architecture is shown in Figure 33.

Inception V3 additionally uses factorized convolutions, which break down bigger convolutions into smaller processes, decreasing computational load without sacrificing the capacity to capture complicated information. It also employs asymmetric convolutions (such as 1x7 followed by 7x1) to boost efficiency.

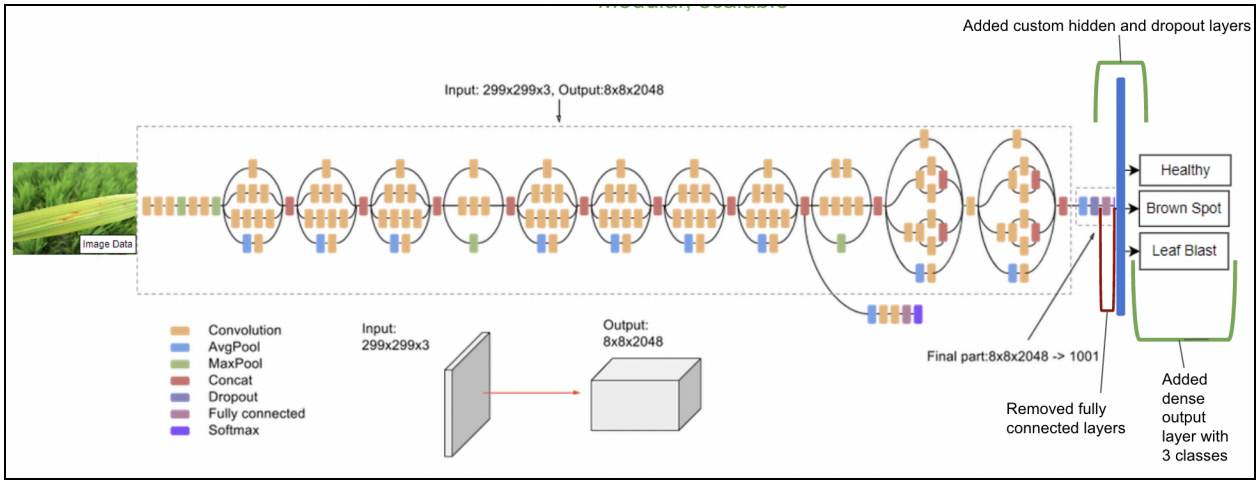
To reduce the dimensionality of the feature maps, this model makes use of both global average pooling and max pooling. This prevents overfitting and reduces the computational cost. Using this could also lead to the loss of some fine-grained image details.

Similarly to other models, the inception V3 model uses the Rectified Linear Unit (ReLU) activation function which adds non-linearity to the network. Relu also helps in speeding up the training period. To stabilize the training, batch normalization is applied to many layers which allows for a higher learning rate and faster convergence. The normalization process is done after every convolutional layer and before the activation function. Furthermore, it functions as a type of regularization, decreasing the requirement for additional regularization approaches such as dropout, especially in the inception modules.

Inception was originally designed for 1000 classes. However, this project concentrates only on four output labels. Those are brown spots, sheath blight, blast, and healthy leaves. For the calculation of the loss function, categorical cross entropy or optimizers like Adam/ stochastic gradient descent are used. The softmax function is applied in the output layer which provides a probability distribution of these four classes. The Inception V3 model has about 23.8 million parameters.

Figure 33

Inception V3 Inception Module



Note: Adapted from "A Review on Deep Learning Techniques Applied to Semantic Segmentation," by A. Garcia-Garcia, S. Orts, S. Oprea, V. Villena-Martinez, & J.G. Rodríguez, 2017, *ArXiv*. Retrieved from https://arxiv.org/abs/1704.06857

4.1.5 Hybrid Model

The hybrid model integrates image data processed by a Convolutional Neural Network and numerical data processed by a densely connected artificial neural network. These types of hybrid models in machine learning are particularly impactful when the predictions can benefit from the high-level features extracted from the images and the contextual information provided by the numerical data.

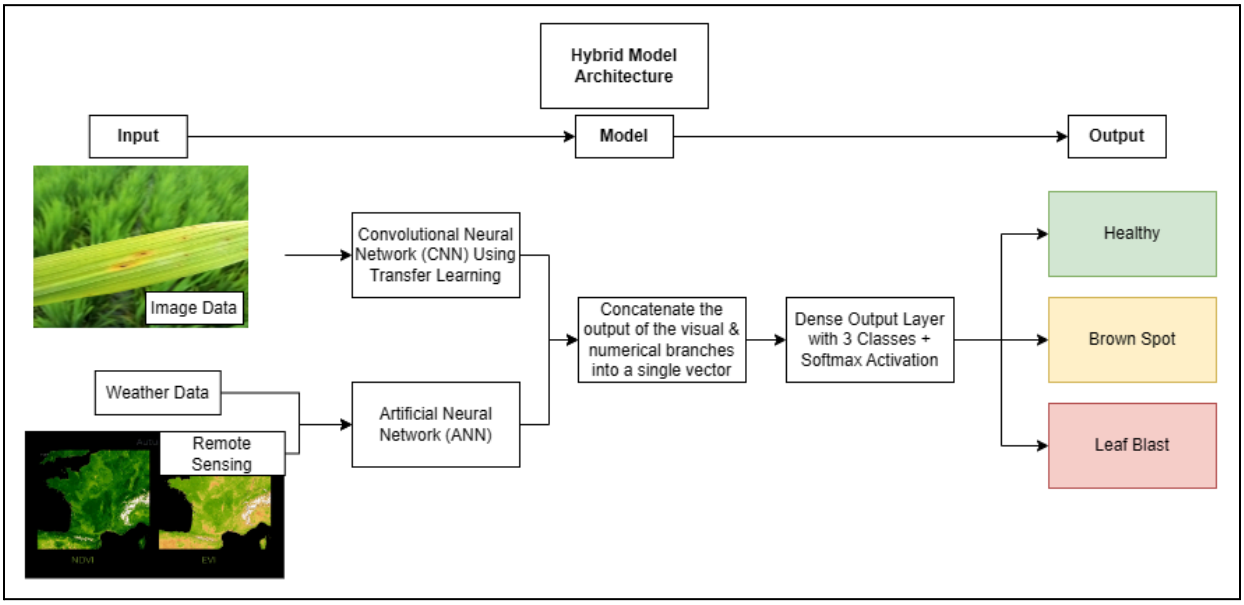
One component of the hybrid model is the visual/image input branch. This branch utilizes a pre-trained CNN, such as VGG19, as the base model to capture the generic features in the images. The last few fully connected layers are removed from the base model, and the output is then processed through additional layers, such as dense and dropout layers to capture the finer details in the images. This branch is adept at handling complex images, and to extract meaningful features that are crucial for tasks involving images. Another component of the hybrid model is the numerical data input branch (which in this case processes the weather and remote

sensing data). This branch processes structured numerical data through multiple densely connected layers. This branch captures the relationships/patterns in the numerical data.

The outputs from both the visual and numerical data branches are then concatenated into a single vector. This vector is then fed to an output layer to make the final predictions. The structure of the hybrid model is shown in the figure 34 below.

Figure 34

Hybrid Model Architecture



4.2 Model Supports

As the foundational platform for developing, training, and deploying the model, the Google Cloud Platform is selected. The platform is also used to store and process the data and deploy a web application for an end-user. Google Cloud Platform includes several tools, each of which is used for specific tasks. Different types of data from a variety of data sources, such as images from different types of cameras, including those collected using UAVs, are stored in Google Cloud Storage. Google Cloud Storage represents a bucket mechanism for storing a wide

variety of data types, such as raw structured, semi-structured, unstructured, image, and video data.

Python programming language is selected as it has become the de facto standard for artificial neural network development, as it provides both libraries for processing algorithms and functions to build models at any complexity level. TensorFlow and PyTorch are selected as the most popular, flexible, rich in libraries, and vastly supported by community deep learning frameworks. Also, model development, training, and fine-tuning can be performed using the same environment. Heavy training can be performed using high-end Nvidia GPUs, such as A100, within the budget of the project. GPUs are also available in the Google Vertex AI environment.

As the main unified environment for the model, the Google Vertex AI platform is selected. It is used to build, deploy, and scale models. The trained model files and running script are containerized using Docker. The Docker image is pushed to the Google Container Repository (GCR). The Google Vertex AI Environment supports the deployment of the model from the GCR container and provides an endpoint, that is used to invoke the model for predictions from the web application. Google Vertex AI is natively integrated with Google Cloud Storage and other Google Cloud Platform tools, such as Cloud Functions, which allows maintaining the project seamlessly across different environments. The data pipeline is built using Google Cloud Functions and Google Pub/Sub. The data preprocessing scripts, such as metadata extraction, weather and remote sensing data fetching, and dataset consolidation for the hybrid model, are deployed as Cloud Functions. The Cloud Functions are triggered by messages orchestrated by Google Pub/Sub service. The Pub/Sub publishes messages to the specific topics, and the topics invoke Cloud Functions or functions inside the web application container. The data pipeline not

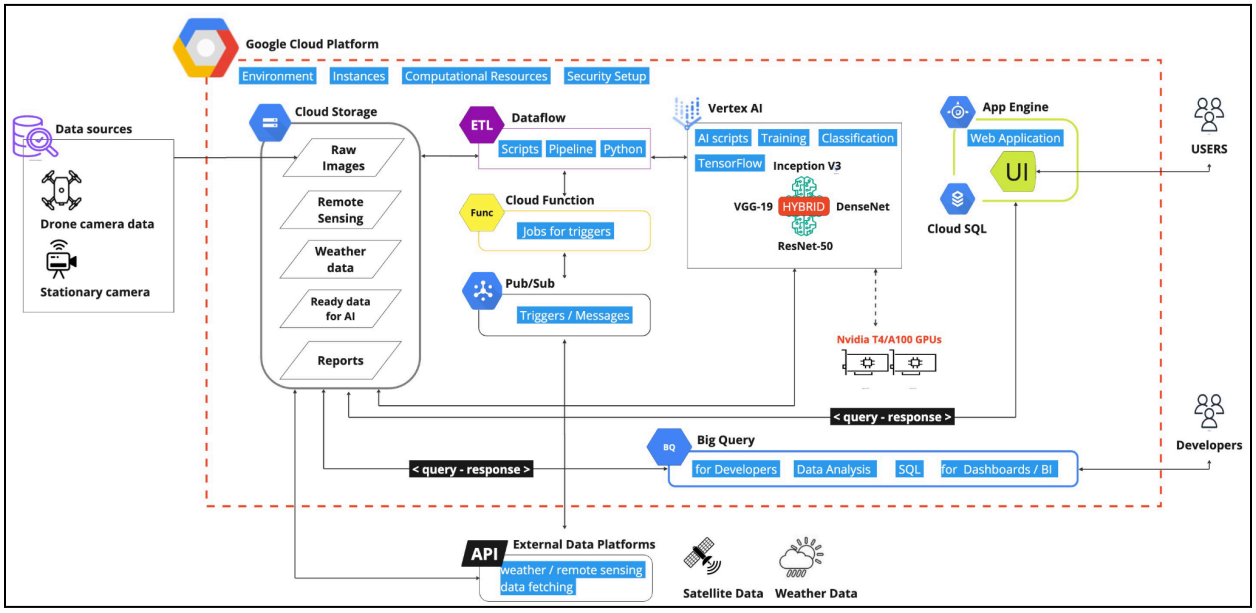
only uses already preprocessed data for the model, but also handles new data loading, preprocessing, and running on the dedicated model instance. Thus, the data pipeline within the Google Cloud platform connects the sources of data, preprocessing scripts, models, and the web application.

Model predictions (JSON and CSV reports) and project files, such as image metadata, weather and remote sensing data, consolidated datasets, are stored in the Google Cloud Storage and can be accessed as reports through the web application. The web application backend is coded using Flask web framework, Python, and AJAX. The frontend is coded using JavaScript language and Bootstrap UI framework. To deploy and host the web application to be accessible for the end-users, the Google App Engine service is used. It also allows the web application to scale on demand.

Thus, the Google Cloud Platform incorporates all key services of the project seamlessly, providing an auto-scalable and maintenance-free development environment.

Figure 35

Google Cloud Platform with Vertex AI showing the full solution cycle deployment



4.3 Model Comparison and Justification

Table 1 provides comparisons for the different base models used.

4.3.1 Model Comparison

Table 5

Model Comparison

Aspect	**DenseNet121**	**ResNet-50**	**Inception V3**	**VGG19**
Year Introduced	2017	2015	2015	2014
Architecture Depth	121 layers	50 layers	48 layers (modules)	19 layers
Key Features	Dense connectivity pattern, growth rate hyperparameter, bottleneck and transition layers	Deep residual networks, Skip connections, Batch Normalization	Inception modules, Factorized convolutions, Auxiliary classifiers, Batch Normalization	Deeper network with small (3x3) convolution filters, Max Pooling
Parameters	7 million	25.6 million	23.8 million	143.67 million
Input Size	224x224x3	224x224x3	299x299x3	224x224x3

Top-1 Accuracy (ImageNet)	~74.91%	~75.3%	~77.45%	~74.24%
Top-5 Accuracy (ImageNet)	~92.27%	~92.2%	~93.56%	~91.85%
Applications	Highly efficient image classification, object localization and segmentation	Advanced image classification, Object detection, Fine-grain classification	Advanced image classification, Object detection, Transfer learning	Image classification, Feature extraction in transfer learning
Pros	Efficient use of parameters, mitigates vanishing gradient problem, reduced number of parameters enhances training efficiency	Very deep networks with effective training, Addresses vanishing gradient problem	Efficient architecture, Handles multi-scale processing	Simplicity and uniformity in architecture

Cons	Increased memory and bandwidth requirements during training and inference due to dense connectivity	Computationally intensive, Complex architecture	Requires larger input size, More complex to implement than simpler models	Very deep networks, leading to high computational cost, Large number of parameters

4.3.2 Model Justification

DenseNet121

DenseNet121 distinguishes itself in the landscape of deep learning with its 121-layer architecture that prioritizes efficiency and performance. This model innovates through dense connections, ensuring each layer communicates directly with every subsequent layer, optimizing information flow and minimizing the model's complexity. This design significantly reduces the required number of parameters, making DenseNet121 not only efficient in computational terms but also highly accurate. It incorporates bottleneck layers to decrease computational demands and uses transition layers for efficient feature map management, resulting in a compact yet powerful network. Compared to more elaborate models such as Inception or ResNet, DenseNet121 stands out for its superior accuracy achieved with lower computational investment and parameter count, presenting an advantageous choice for settings with computational constraints. Moreover, its architecture naturally counters the vanishing gradient issue, supporting

deeper network training with less difficulty. Despite these advantages, the intricate connectivity of DenseNet121 may pose implementation challenges and necessitate greater memory allocation during the training phase, compared to more straightforward architectures.

ResNet-50

With the introduction of residual connections, ResNet-50 reshaped deep learning by addressing the vanishing gradient problem and enabling the training of much deeper networks than was previously possible. This depth is balanced with efficiency, as the skip connections make training easier and performance better. ResNet-50 has proven to be adaptable, with applications ranging from image classification to object detection and segmentation. However, due to its depth and architectural complexity, it is computationally demanding, making deployment in environments with limited computational resources difficult.

Inception V3

Inception V3 is well-known for its efficient and scalable architecture, which employs factorized convolutions and inception modules to reduce computational costs while maintaining performance. This architecture is especially adaptable for processing inputs at multiple scales, which is critical for complex image recognition tasks. Advanced training techniques, such as the use of auxiliary classifiers and batch normalization, help with effective training and faster convergence. Despite its efficiency, Inception V3's complex structure can be more difficult to implement and requires more computational resources than simpler models such as DenseNet121.

VGG19

VGG19 is distinguished by its depth combined with a consistent architectural layout that allows scaling and modification. This model excels at feature extraction, making it extremely

useful in a wide range of applications, particularly transfer learning, where its features are repurposed for different tasks. The design of VGG19 has had an impact on the evolution of CNN architectures. The depth and large number of parameters in VGG19, on the other hand, come at a high computational cost. It is computationally expensive and takes longer to train, which can be a significant disadvantage in large-scale implementations or when computational resources are limited.

4.4 Model Evaluation Methods

For a multi-class classification task, there are a number of different metrics that can be used. Some commonly used evaluation metrics for multi-class classification problems that will be discussed in this project are accuracy, macro precision/recall/F1-score, and AUC-ROC (adjusted for a multi-class setting). After the training phase, evaluating the performances of the models is necessary.

The proportion of correct predictions (both true positives and true negatives) out of the total number of cases examined is commonly referred to as accuracy. It provides a quick, intuitive measure of the overall performance of the model. The formula for accuracy is shown in (1).

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Instances} \quad (1)$$

The F1 Score is calculated by taking the harmonic mean of precision and recall. These two metrics are combined into a single number. F1 Score is particularly effective when a balance of precision and recall is required, and when the costs of false positives and false negatives are comparable. The formula for F1 score is given in (2).

$$F1 - Score = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (2)$$

Precision is an important statistic in situations where the cost or impact of false positives is considerable. It denotes positive prediction accuracy, reflecting the proportion of predicted positives that are correct. It is calculated by dividing the total number of true positives by the total number of true positives and false positives. While precision provides useful insights into the model's performance in properly detecting positive events, it ignores examples that the model incorrectly detects as negative, which is a significant limitation. The formula for precision is shown in (3).

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (3)$$

The model's ability to properly detect all actual positives is measured by recall, also known as sensitivity. The ratio of true positives to the sum of true positives and false negatives is specified. Recall is especially important in instances when failing to notice positives can be disastrous. A high recall demonstrates a model's ability to detect positive cases, making it an important statistic. The formula is given in (4).

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (4)$$

In order to obtain the macro precision, recall, and F1-scores for a multi-class setting, the respective metrics have to be obtained for each individual class, and then the average is taken. Obtaining the true positives, false positives, and false negatives for an individual class can be achieved by treating the task as binary (i.e. the class in question against every other class). It is possible to also obtain a weighted average for these three metrics, which assigns a higher weight to classes that are larger. The closer these metrics are to 1, the greater the ability of the model to correctly predict and identify each individual class.

The confusion matrix provides a thorough picture of a model's performance by tabulating the actual versus predicted classifications. The number of predicted and true labels for each class

is displayed. This granular breakdown is especially useful in multiclass classification situations since it allows you to see how well the model performs for each class and where it gets confused. The confusion matrix is useful for diagnosing a model's performance beyond mere accuracy, providing insights into the types of errors made by the model and directing changes in model training.

The Receiver Operating Characteristic (ROC) curve and the Area Under the Curve (AUC) curve are used to assess the effectiveness of a classification model at various thresholds. The ROC curve, which compares the True Positive Rate (recall) to the False Positive Rate (1-specificity), emphasizes the trade-off between correctly recognizing positives and minimizing false positives. The AUC, which ranges from 0 to 1, assesses the discriminative capacity of the model, with 1 representing perfect prediction and 0.5 suggesting random chance performance. In a multi-class task, the macro AUC-ROC score is obtained by calculating the AUC for each individual class through one-vs-one (OvO) or one-vs-rest (OvR) schemes. The average is then taken to obtain the AUC-ROC across all classes. AUC-ROC is useful for model comparison, especially in unequal datasets, although it can be excessively optimistic in severely unequal settings.

4.5 Model Validation and Evaluation Results

In total, eight separate models were trained. Four purely visual models were trained using CNNs and transfer learning, and four hybrid models were trained that incorporated image and numerical data, the architecture of which was discussed previously. In addition, ensemble models for each kind of model were evaluated, using both a majority voting and soft-vote scheme. The accuracy of all of the models on the test set are presented in the following table.

Table 6

Accuracies of Each Trained Model Type on the Test Set, along with Model Ensembles

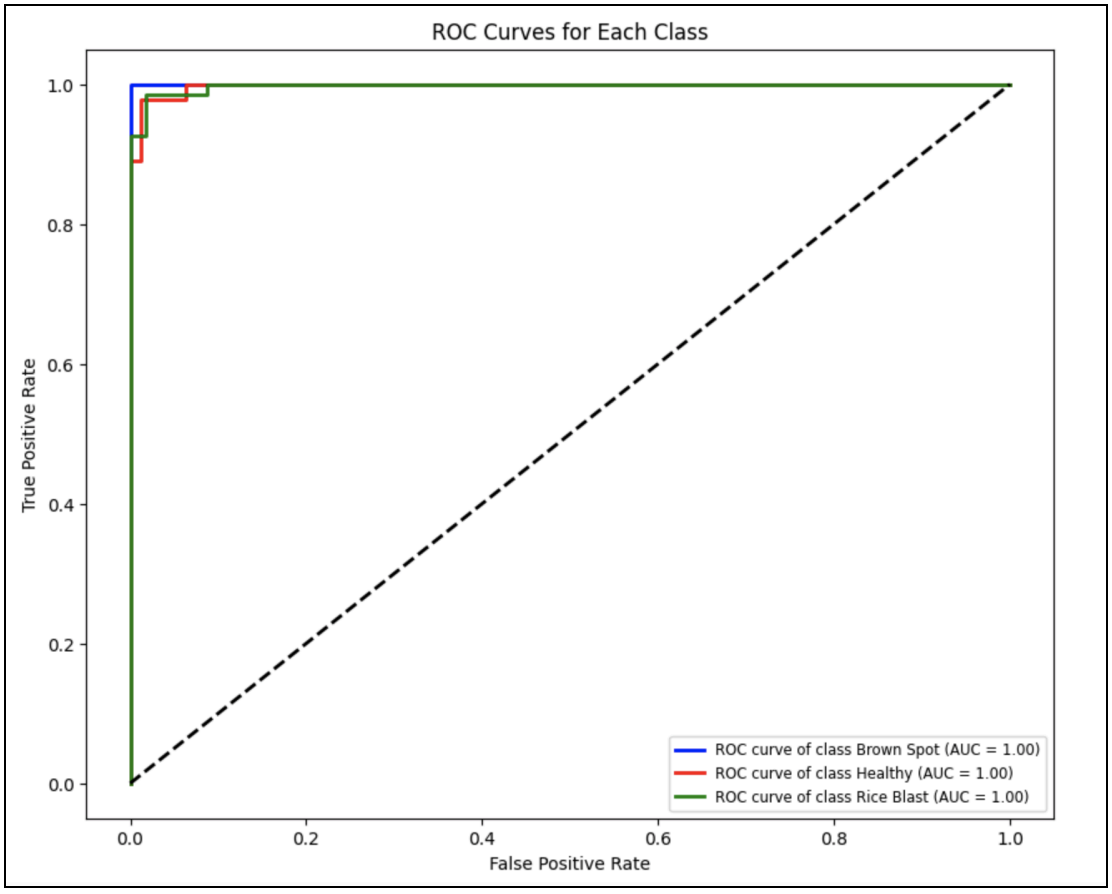
Base Model	**Purely Visual Model**	**Hybrid Model**
VGG-19	89.60%	91.20%
InceptionV3	92.00%	**96.00%**
ResNet50	93.60%	95.20%
DenseNet121	94.40%	**96.00%**
Ensemble (Soft Voting)	95.20%	95.20%
Ensemble (Majority Voting)	94.40%	94.40%

For each of the base models, the hybrid model outperformed their purely visual model counterpart. The hybrid InceptionV3 and DenseNet121 models have the highest accuracies out of all of the models, including the ensemble models (the ensemble models of the hybrid models likely did not perform as well since the errors of the individual models are most likely correlated). The hybrid DenseNet121 model is smaller than the InceptionV3 model, and thus the hybrid DenseNet121 model is chosen as the final model. Various visualizations and metrics of the results of this model will now be shown.

The ROC-AUC curves for each class (healthy and for each individual disease) using a one-vs-rest approach are shown in Figure 36.

Figure 36

ROC-AUC curves for Every Class, One-vs-Rest Approach



The ROC curves, along with corresponding AUC scores, show that the model performs exceptionally well in distinguishing between positive and non-positive cases for each class. The macro-average ROC-AUC score is 0.9984 which indicates that the model is exceptionally well at distinguishing between every class. The classification report, which includes metrics such as recall, precision, and F1-score, is shown in the next figure 37.

Figure 37

Classification Report for the Final Model

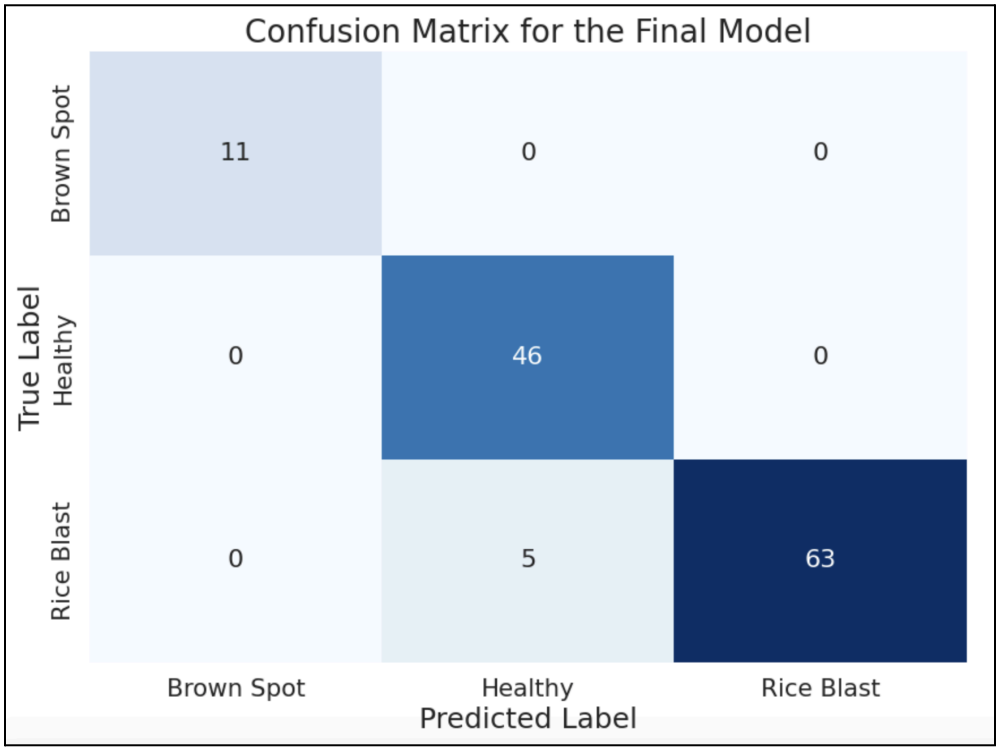
	precision	recall	f1-score	support
Brown Spot	1.00	1.00	1.00	11
Healthy	0.90	1.00	0.95	46
Rice Blast	1.00	0.93	0.96	68
accuracy			0.96	125
macro avg	0.97	0.98	0.97	125
weighted avg	0.96	0.96	0.96	125

The precision of the "Healthy" class is the lowest, which indicates that there are false positives for this class. Overall, the F1-score for each class, and the macro average F1-score is very high of 0.97, further underscores the predictive ability of the model.

The confusion matrix is shown in the next figure 38.

Figure 38

Confusion Matrix of the Final Model



Five cases of ‘Rice Blast’ were predicted to be healthy, which explains the recall and precision of the ‘Rice Blast’ and ‘Healthy’ classes, respectively. Every case of ‘Brown Spot’ and ‘Rice Blast’ was identified. Overall, considering every metric, the final hybrid DenseNet121 model performs very well at distinguishing between classes and has exceptional predictive ability by utilizing both image and numerical information.

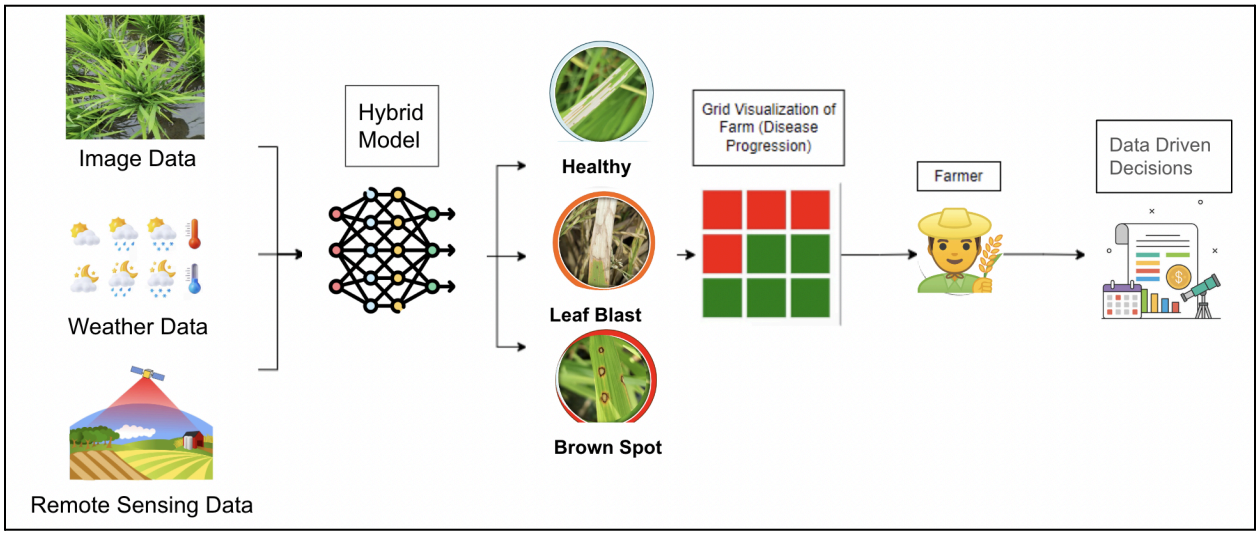
5. Data Analytics System

5.1 System Requirements Analysis

5.1.1 System Boundaries and Use Cases

Figure 39

System Use Case



This system is designed to detect the diseases in rice crops and classify them into brown spot, leaf blast and confirm healthy plant conditions. It is highly important in disease prevention and increasing food production. The system merges the data from various sources: high-resolution satellite imagery that gives insights into vegetation health through NDVI and EVI indexes, ground level detailed photographs of rice leaves showing disease symptoms, and crucial weather data including temperature, humidity and rainfall trends. The system's boundaries are broad, encompassing every step from gathering data to analyzing it and ultimately predicting and categorizing the crops status. It is structured to integrate with satellite data for NDVI and EVI, weather databases for conditions and cloud services, for effective data management and storage.

In this system, the actors who are involved are primarily the farmers, who benefit from the early detection of the rice crop diseases and the management; agronomists offering expertise in disease identification and mitigation strategies, data scientists who develop machine learning models; and system administrators, who ensure the system's operational integrity and data security.

The system's use cases are designed to meet the requirements of stakeholders involved in managing agriculture and controlling diseases. Firstly, the system offers real-time detection of diseases in rice crops, allowing farmers to take timely preventive measures. It identifies the diseases affecting the crops, giving insights to agronomists and farmers about the crops health. Additionally, it analyzes the datasets to improve the accuracy and reliability of disease detection and classification results. Another use case is prediction and forecasting the potential disease outbreak based on historical data trends and current weather conditions. The system also sends alerts and reports providing stakeholders with information on disease management strategies. Through these use cases, the system aims to play a role in sustainable agriculture by optimizing disease management practices to ensure food security and improve crop yield quality.

5.1.2 High-level data analytics and machine learning functions and capabilities.

This study focuses on detecting and classifying diseases in water-intensive crops such as rice. It also incorporates advanced data analysis and deep learning methods. It tackles the task of detecting diseases like brown spot, leaf blast, as well as healthy plants. The system uses four Convolutional Neural Networks (CNNs): VGG 19, ResNet 50, Inception V3 and DenseNet121. Each designed to process and analyze images captured from the farms. These CNNs extract features, such as variations in color, texture and irregularities, and shape indicating plant health.

To enhance the analysis this system also uses remote sensing information by incorporating Normalized Difference Vegetation Index (NDVI) and Enhanced Vegetation Index (EVI). These indices offer a view of the crop's health that extends beyond what is visible in the images alone. Additionally, it integrates weather data such as temperature, humidity and rainfall metrics for predicting disease outbreaks. By combining these data sources and applying deep learning algorithms, the system can accurately identify and categorize specific crop disease while also forecasting potential outbreaks through an analysis of environmental conditions and historical data patterns.

As mentioned previously, this project looks into multiple different data forms which means different models are utilized for the different data forms. Different machine learning models such as CNN for visual models, ANN for numerical models and even a combination of both in the form of a Hybrid Model which takes into account both visual and numerical data in its learning tasks.

5.2 System Design

5.2.1. System Architecture and Infrastructure

The system for water-intensive crop disease detection and classification consists of the frontend and backend components. The user interacts with the system using a cloud-based web application that has a convenient user interface (UI). The frontend component covers the UI. Using the UI, the user inputs new data and receives final classification reports. The backend component consists of several subsystems: data storage, data extraction, transformation and loading (ETL), CNN models execution, and reporting. The system architecture is designed to hold and perform a specific use-case scenario.

In the use case scenario, the user has raw images from the camera. A camera can be mounted on a variety of equipment, such as a drone, stationary or mobile camera. Images are taken from the specific crop field. The user gets access to the cloud-based web application through UI on any workstation, such as a laptop or cell phone, that has internet access and an internet browser. The user uploads crop images from the field into the system. The backend subsystem ETL is a data processing pipeline that orchestrates the data movement and transformation across the system, including results reporting to the user. At first, user images are processed to extract the metadata, which is date and location. Based on the date and location system fetches weather and remote sensing data from the external data platforms. The obtained data is then processed by cleaning and transformation steps. The output of these transformations is the prepared images for CNN models, and weather/remote sensing data for disease risk assessment script. CNN models perform crop disease detection and classification. Output is the classification report. The disease risk assessment script utilizes dependencies between weather and remote sensing data, and based on the common disease factors that occur in certain diseases, notifications can be made and be used to alert the farmer or application user. As a result of the hybrid model classification result and the risk assessment script, the final classification report is built. The final classification report is accessed by the web application and outputted into UI, thus, the user can conveniently read the results and consider them for further decision-making in agricultural methods of preventing disease spread and maximizing the crop field yield.

5.2.2 System Supporting Platforms and Cloud Environment

The system architecture is built on Google Cloud Platform products and services (GCP). The system environment diagram is shown in Figure 40. GCP provides cost-effective solutions that can be seamlessly integrated, which reduces setup and maintenance time.

As a data storage and management solution Google Cloud Storage (GCS) is used. GCS buckets can store raw and processed images, weather and remote sensing data fetched from external data platforms, and intermediate data generated in the project. Separate folders will be used to store data at different processing stages.

Further, the ETL pipeline is built on several services. Image metadata is extracted using the Google Cloud Functions, which script is based on the “Pillow” library and is triggered by Cloud Pub/Sub messages when new image data is uploaded to the Cloud Storage. The extracted metadata is stored in Cloud Storage and also accessed by Google BigQuery for analysis and future system development. Data cleaning and transformation steps are also deployed on the Google Cloud Function service. The Cloud Functions are triggered by Pub/Sub messages and run the ETL scripts that prepare the data for modeling. The output data is stored in Cloud Storage. The scripts are programmed using Python language and its libraries, such as Pandas, TensorFlow, PyTorch, and Scikit-learn.

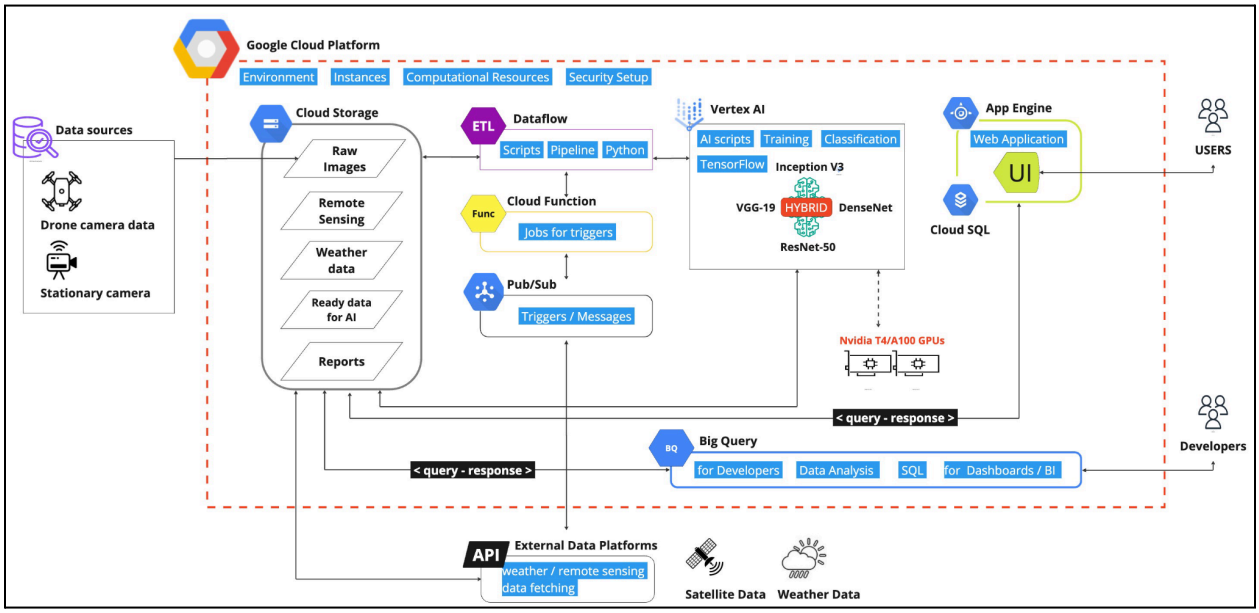
The Google Vertex AI is used for model training and deployment. The service supports direct access to the data in the Cloud Storage. The CNN model script is containerized using Docker and deployed on Vertex AI as the endpoint for predictions. Predictions are invoked by Pub/Sub messages that trigger Cloud Functions when the system user uploads new data.

The user-friendly web application with UI is built on the Google App Engine service. The App Engine provides cost-effective, fully managed, auto-scalable, and integrated with other system environment components. Image upload functionality along with observing final classification reports is implemented. The security is managed by Google Identity and Access Management (IAM), Service Roles, and Secrets Manager, which ensure that the system environment is secured and accessed only by authorized users.

Thus, architecture built on the Google Cloud Platform allows deployment of the project logic and functionality along with data distribution at high performance and flexibility. The platform provides a cloud-native environment for utilizing the effectiveness of image processing CNN modeling workflow.

Figure 40

The System Architecture Platforms and Cloud Environment



5.2.3 System Data Management Solution

The system's data flow is reflected in the diagram shown in Figure 41. From the data flow perspective, the system has entities and functions that exchange data in specific directions.

The system entities are the User, UI, Web application (App Engine), Storage (Cloud Storage), ETL pipeline (Cloud Functions), Triggers (Pub/Sub), External Data Platforms (API), AI platform (Vertex AI), and Analytics Platform (Big Query).

The system functions are Image Batch Upload, Report output, Folders that perform storage logic, ETL's extraction, fetching, cleaning, transformation and merging scripts, CNN

models and their ensembling script, and the hybrid model script. The system data is raw user images (jpg files), and table data (CSV files) for location, date, weather, remote sensing features, classification report files (JSON and CSV), and the final report.

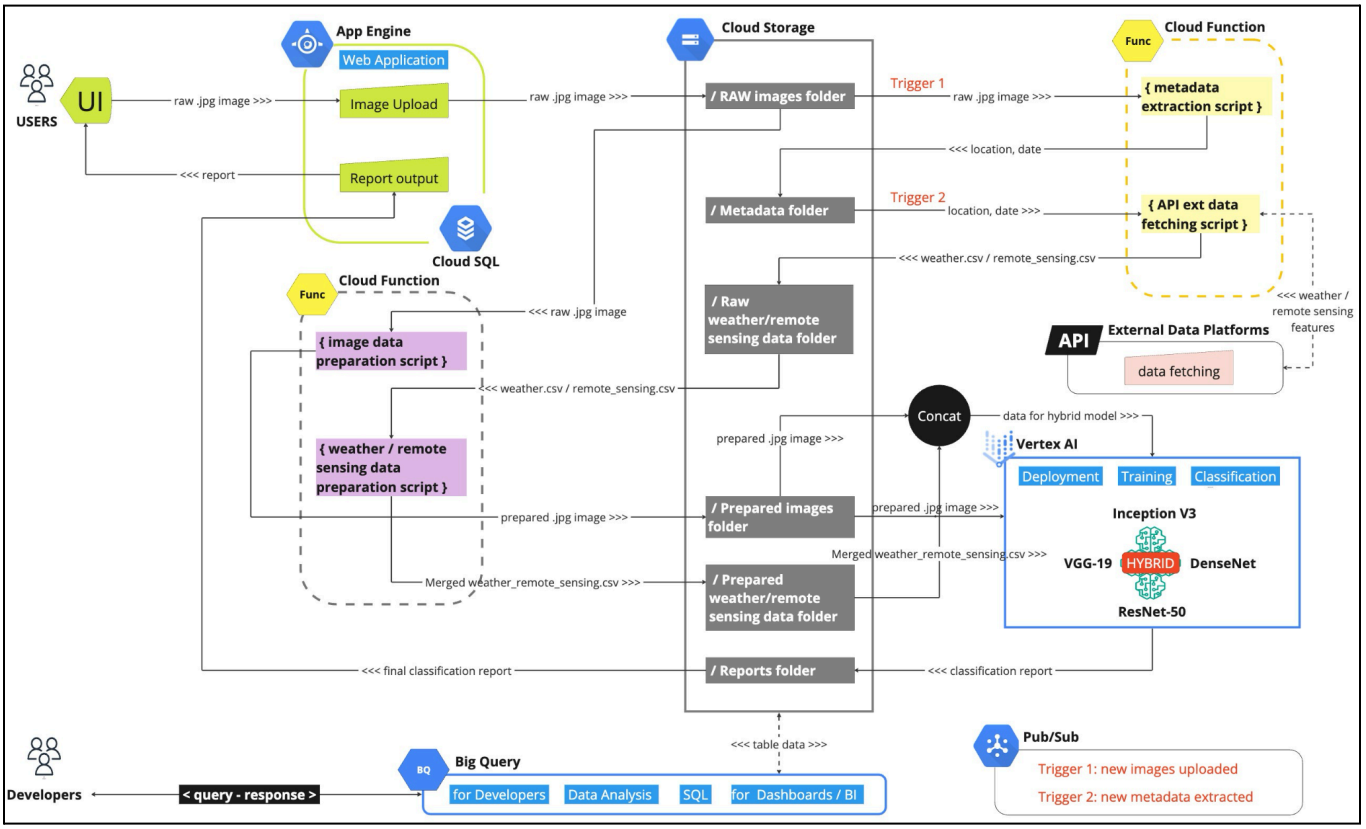
In the scenario, the User uploads data via UI using the “Batch Upload” function of the Web application. The data is written to the Storage using “Folder logic for RAW image files”. Then, after images are uploaded the Pub/Sub message triggers the Extraction script for image metadata. The extracted metadata is in table format and is written to the Storage as a CSV file using “Folder logic for metadata”. After new metadata is extracted the Pub/Sub message triggers the Fetching script, which uses the metadata CSV file from the Storage to fetch the weather and remote sensing data from External Data Platforms (API). Fetched data is in table format and written to the Storage as a CSV file using “Folder logic for Raw weather / remote sensing data”. Then, another Pub/Sub message triggers the Cloud Functions to run image and weather / remote sensing data cleaning and transformation scripts. The output is images and a CSV file that are written to Storage using “Folder logic for Prepared images” and “Folder logic for prepared weather / remote sensing data”.

The AI platform has the hybrid model script and model artifact files deployed. When newly prepared data is written to the Storage the Vertex AI hybrid model endpoint is triggered for predictions. The output is the classification report JSON and CSV files, which are written to the Storage. Reports are pulled from the storage to the UI using the Web application UI.

For data analysis and future system development, such as dashboard development, the Google Big Query is used to access table format data located in the Storage. The access is granted for developer roles.

Figure 41

The System Data Flow



5.2.3 System User Interface

The UI is the early prototype reflecting only the main necessary functions to execute the full cycle of the use case scenario of the Water-Intensive Crop Disease Detection and Classification solution. The goal of the current project is to create a fully functional Minimum Viable Product (MVP), which is a product with main features to satisfy early users and to receive feedback for future product development.

Figure 42 shows the Welcoming Screen after opening the web application. The next screen is User Authorization. Here User must enter his email address and password. The authorization screen is shown in Figure 43.

Figure 42

Welcome Screen of the Web Application

Figure 43

Authorization screen

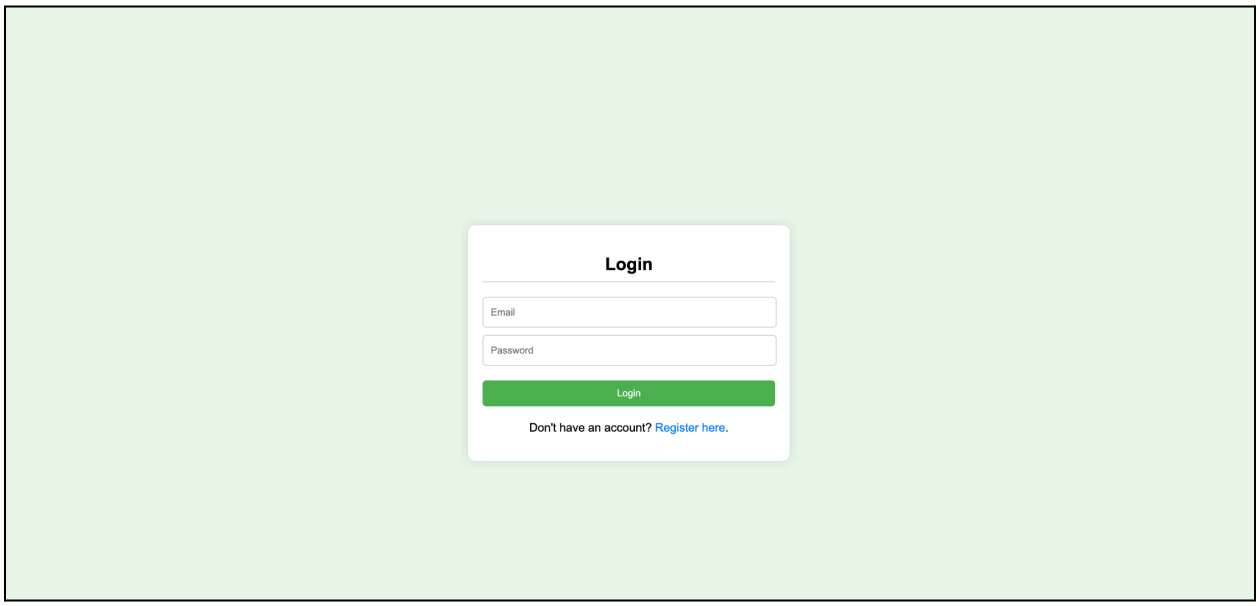


Figure 44 reflects the main screen. The main page consists of Fields, Batches, and Images tables. The user can conveniently observe items. The table has the following columns: Field ID, Batch ID, Field Name, User Name, Date Created, Date Uploaded, Label, and Confidence levels (Healthy, Brown Spot, Rice Blast). The Field name is the random name given by the user. The Field ID is the number assigned to the farm field for identification and distinguishing among other fields. A new Batch can be uploaded by clicking the button “Upload Batch”. The user must select the images from the local storage.

Figure 44

Main screen

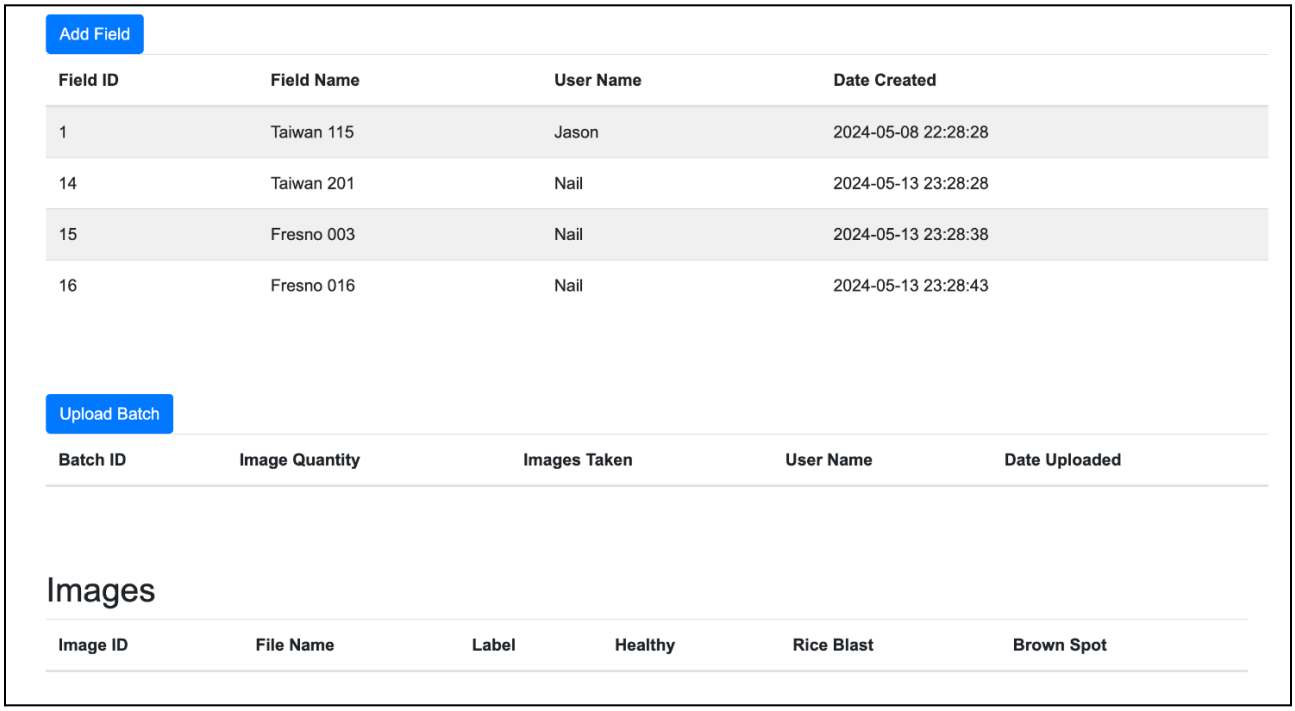


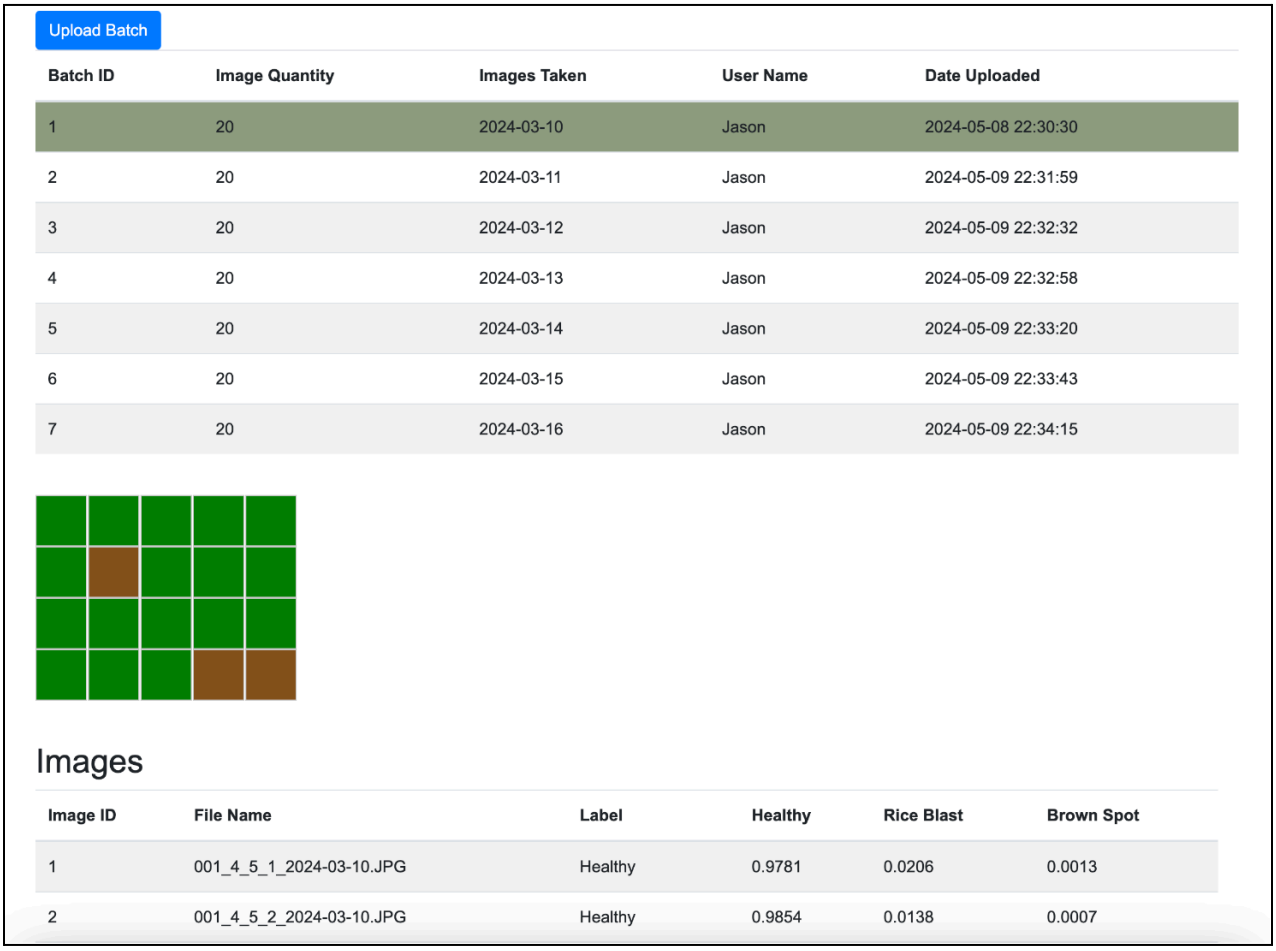
Figure 45 shows the Batches and Images tables. Batches are shown for the selected Field in the Fileds table. One Field can have multiple Batches. Each Field can have multiple Batches. The Date Uploaded column shows when the image batch was uploaded. The Label and Confidence levels (Healthy, Brown Spot, Rice Blast) indicate the results after AI models

performed classification on the uploaded data. The User column stores the record of the person who uploaded a batch.

The Grid is shown for the selected Batch. The Grid reflects the physical crop field with sectors. Each sector is the image taken by the camera. The Grid dimensions reflect the frequency of the images that were taken by the camera in one direction from side to side of the field. For instance, if the camera took 100 images in one row of the field (from side to side), then the Grid will have 100 sectors in one row. The color of the sector is the predicted Label. The green color is for “Healthy”. The brown color is for “Brown Spot”. The red color is for “Rice Blast”.

Figure 45

Batches of the selected Field



5.3 System Development

5.3.1 AI and Machine Learning Models Development

The project introduces AI and ML driven systems that have been developed to detect and classify rice crop diseases. By using Convolutional Neural Networks (CNN) this system makes use of a cloud-based web application that enables users to upload images of crop fields, for analysis. The backend system employs an ETL pipeline to process data, extracting metadata from the images to gather data, which is then paired with the images for CNN analysis. The project has a two-phase approach to disease detection. Comparisons in the results model will compare whether a purely visual model such as CNN is used or a Hybrid Model which takes into account both visual and numerical data (Rice Disease Images with the addition of weather and remote sensing information). This process generates a report on disease classification through the user interface offering valuable insights, for making decisions in agriculture.

The four CNN models that are used in this project are VGG 19, ResNet 50, Inception V3 and DenseNet121.

- DenseNet121 is distinct for its 121-layer dense architecture. The architecture includes ReLU (Rectified Linear Unit) activation functions, transition layers to efficiently manage the feature map, and bottleneck layers to decrease computational demands. DenseNet121 is able to achieve high accuracy with a low parameter count and is able to naturally counter the vanishing gradient problem, which aids in neural network training. Its architecture is skilled at identifying intricate patterns in images for initial crop disease detection.
- Inception V3 stands out for its efficient and scalable architecture which includes factorized convolutions and inception modules. It reduces computational costs along with

maintaining performance. This design excels in handling inputs of varying scales, making it particularly suitable for intricate image recognition challenges. For effective training and faster convergence, it uses advanced technologies such as auxiliary classifiers and batch normalization. Due to its complex architecture, this model is difficult to implement compared to other models.

- VGG 19 is notable for its deep and uniform architectural structure. It is capable of robust feature extraction across various applications, especially in transfer learning contexts. Due to its depth and large number of parameters involved in VGG 19, it has high computational cost.
- ResNet-50 resolved the vanishing gradient problem by introducing residual connections. It can train much deeper neural networks than previously possible because of it. These residual connections facilitate smoother training and enhance performance by introducing skip connections that balance depth with efficiency. ResNet-50's versatility is evident in its wide range of applications, from image classification to object detection and segmentation.

5.3.2 Implemented Designed System

The system design, for detecting and classifying water intensive diseases in rice crops follows a strategy that combines visuals, weather data and remote sensing data. The system involves using cameras attached to drones or other devices to capture images of crop fields. Users can easily upload these images through a user interface on their smartphones or laptops to a web application stored in the cloud. Once the images are uploaded in the system, the backend starts a data processing pipeline, also known as ETL (Extracts, Transform, Load). This pipeline then starts extracting metadata from the images such as latitude and longitude, and date. With the

use of metadata, the system collects the weather data such as temperature, humidity, rainfall, and remote sensing data such as NDVI and EVI for enhancing the analysis.

The images and external data undergoes cleaning and transformation and prepares them for processing by convolutional neural networks specifically designed for deployment, training and classification. A special 'confidence boosting script' within the system combines the model's results with data to create a classification report. The whole process showcases an automated approach to improving agricultural methods.

5.3.3 Input and Output Requirements, Supporting Systems and Solution APIs

- Input Requirements: User uploads raw JPEG images via a Web Application. Weather and remote sensing data are used to enhance the accuracy of image classification.
- Output Requirements: After classification, a detailed report is generated with analysis and insights. Images are prepared and combined with weather/ remote sensing data for future analysis or record keeping. Besides the model output, additional information can be displayed depending on the current weather and remote sensing information which can help farmers gauge disease risks based on known factors at which the disease thrives.
- Supporting Systems: Cloud Storage organizes images, metadata and processed data into folders. Cloud Functions activates scripts for metadata extraction and API data retrieval. Dataflow ETL services executes preparation scripts on images and weather data. Cloud Scheduler automates processes based on uploads or metadata extraction.
- Solution APIs: The system utilizes 2 APIs. Google Earth Engine API, which provides the location of the image taken and the date. Weather Crossing API, which fetches script, to incorporate weather data.

5.4 System Supporting Environment

System Platform

The system platform is built on Google Cloud Platform (GCP), utilizing Google Cloud MySQL to manage database management, Google Cloud Storage for image storage, and Google Vertex AI for machine learning models. Web application has been hosted on App Engine with data processing managed by Google Cloud Functions and Pub/Sub. The primary programming languages used are Python and JavaScript. The libraries used are JQuery for DOM manipulation and for handling AJAX calls, Werkzeug for developing authorization functionality in Python, and SQLAlchemy for interacting with databases. Logging is implemented for support logs. To ensure flexibility and smooth development, Flask is used for backend frameworks. Bootstrap has been used for creating frontend UI. Visual Studio Code is used for development and JetBrains DataGrip manages the database operations. This setup offers an efficient, scalable, and reliable foundation for a web application that classifies rice crop diseases accurately.

Programming languages and libraries

The project uses Python and JavaScript as its programming languages with each serving distinct roles in the application. Python is utilized for backend development, leveraging the Flask web framework for creating web applications. Python allows rapid development and easy maintenance of backend code. JavaScript is employed on the frontend to build interactive user interfaces enhancing the user experience. Various libraries are incorporated to improve the functionality and efficiency of the codebase. JQuery simplifies HTML DOM manipulation and event handling, making it easier to develop interactive web features. Werkzeug offers a set of libraries for building WSGI web applications in Python, enhancing the capabilities of the Flask framework. SQLAlchemy acts as an SQL toolkit and object-relational mapper which enables

smooth database interactions within the application. Moreover, the logging library supports logging activities assisting in debugging and monitoring the applications performance.

Frontend and UI Frameworks

The frontend of the system relies on a combination of HTML, CSS, and Javascript. This creates a visually appealing and responsive UI. HTML structures the content of the web pages, and CSS is responsible for styling, and designing attractive layouts across different devices and screens. JavaScript enhances the user experience by enabling elements on the UI. In UI design the system utilizes the Bootstrap framework which adapts well to devices. It simplifies the process of creating a contemporary and visually attractive UI without requiring extensive CSS modifications.

6. System Evaluation and Visualization

6.1 Analysis of Model Execution and Evaluation Results

In order to achieve better model accuracies for both the visual-only and hybrid models, further tuning was performed. For instance, a greater combination of dropout rates and the number of neurons in the hidden layers were experimented with. In addition, the total number of frozen layers was modified in order to further fine tune the base transfer learning models. Typically, transfer learning involves freezing all of the trainable layers and removing the last few fully connected layers, and then adding layers which are specific to the details of the labels in the original dataset (for instance having 5 neurons in the output layer if there are 5 classes originally). In our case, some layers at the end were unfrozen to allow for further training, which may help the model better adapt to the features of our specific dataset (in the hybrid model, the visual branch contains the pre-trained base model with the last few layers unfrozen). Deciding how many layers are frozen involves careful consideration since having a large amount of trainable layers increases computational demands and can heighten the risk of overfitting. This occurs due to the model being overly adapted to the training data which compromises its ability to generalize to new instances. After tuning each individual visual-only and hybrid model, ensembles were created similarly to the original models, using both soft-voting and majority voting. The results of the models after further tuning is shown in the table below:

Table 7

Accuracies of Each Trained Model Type on the Test Set, Before and After Fine-Tuning

Base Model	**Visual-Only Model (Old/New)**	**Hybrid Model (Old/New)**
VGG-19	89.60% / 91.20%	91.20% / 96.80%

InceptionV3	92.00% / 95.20%	96.00% / 96.00%
ResNet50	93.60% / 97.60%	95.20% / **98.40%**
DenseNet121	94.40% / 96.80%	96.00% / **98.40%**
Ensemble (Soft Voting)	95.20% / 97.60%	95.20% / 96.00%
Ensemble (Majority Voting)	94.40% / 97.60%	94.40% / 96.00%

After more extensive experimentation with the modeling parameters and fine-tuning, every model achieved a higher accuracy on the test set, except the hybrid InceptionV3 model which maintained the same accuracy (our task is a classification task, and thus the metrics are obtained by comparing the predicted labels and the true labels). Similar to the previous results, the hybrid models have higher accuracies compared to their visual-only counterparts. Out of all of the models, the hybrid ResNet50 and DenseNet121 models have the highest accuracies. The models are stored in a .h5 format. Since the hybrid DenseNet121 model is smaller in size than the ResNet50 model, the hybrid DenseNet121 model is chosen as the final model in order to have a more lightweight model for use in the application. The macro-average ROC-AUC score of this model using a one-vs-rest scheme is 0.9994, and the macro-average F1-score is 0.99, which further underscores the exceptional predictive ability of the model.

6.2 Achievements and Constraints

In order to create a water-intensive crop disease detection and classification system using machine learning the following tasks need to be completed:

- Collect and prepare water-intensive crop diseases dataset. The dataset includes remote sensing data, weather data as well as image data.
- Develop machine learning models based on selective models and improved models to support water-intensive crop disease detection and classification
- Generate a disease distribution map with specific location and severity
- Develop a web-based portal that can:
    - Perform crop disease and classification based on a given dataset for a specific farmland
    - Support the monitor and analysis of crop disease distribution with severity in a selected farmland(s)

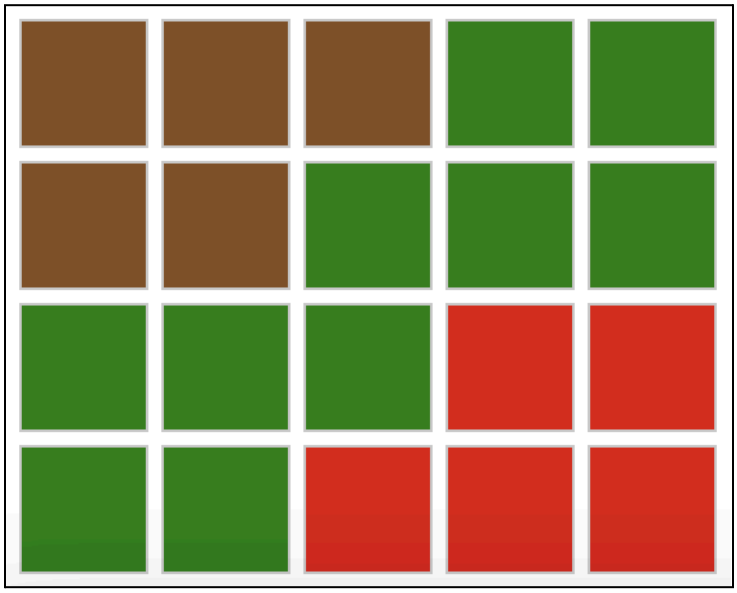
Rice will be used as the water-intensive crop model for this project due to its significance in global consumption and production especially in countries in Asia. The task of utilizing a combination of camera images, weather data and remote sensing data requires a dataset with a wide range of information. Weather data as well as remote sensing data can be extracted given time and location in the form of latitude and longitude. The image data used for the model also must be labeled images of the rice leaf plant taken in the field, not a singular leaf in a white background which most of the available datasets contain. Field images can simulate a drone moving through a rice field and taking images which is the application of this project. The one dataset that fulfills the following conditions is the Rice Leaf Disease in Taiwan which contains a total of 927 rice leaf images of classes brown spot, leaf blast and healthy. Images in the dataset are taken in the field and contain valuable metadata in the form of time and location. The image data was cleaned for duplicates and weather data with remote sensing data was extracted to create a dataset with a wide array of information containing both visual and tabular data.

The use of image data for classification tasks has been widely used, this project aims to explore the possibility of adding more data on top of image data to improve classification tasks of detecting diseases in water-intensive crops. A comparison of visual model results against a model that uses both visual and numerical data (hybrid model) will be required to see whether adding more features can improve the accuracy of disease classification tasks. The visual models used in this project are VGG19, ResNet50, InceptionV3, and DenseNet121. Ensemble models were used as well for the visual-only models. Then the hybrid model was tested which added on top of each existing visual model. Further fine-tuning where different activation functions, number of neurons, and freezing of layers were tested for better results. As shown in the results section, across the board the hybrid model performed better than the visual-only models. The final models are saved as an h5 file. This indicates an achievement in the machine learning model of this project which proves that additional information and data added on top of visual image data can in fact improve performance in crop disease classification. Using weather and remote sensing with image data can be very helpful and useful in smart agricultural fields where accurate detection and classification of diseases are imperative to the yield as early detection can lead to better handling of diseases and mitigate crop damage from diseases.

With the hybrid model able to perform disease classification tasks at a high level of accuracy, this information still needs to be presented to the user at the end. By performing multiple classification tasks at different areas in the rice field, a distribution map can be made which shows in a grid which represents the field, how many and which areas are affected by certain diseases. Performing this task daily can provide multiple distribution maps which inturn can show the progression of a certain disease in the field over time. An example of the distribution map is shown below.

Figure 46

Distribution map (brown for 'brown spot', red for 'leaf blast', green for 'healthy')



To optimize GUI an interactive web portal will be further developed allowing for users to easily perform crop disease classification and detection as well as monitor the crop field on a regular time interval to monitor if any disease progression is apparent in the field and needs to be taken care of immediately. This application will be very helpful, especially for farmers who tend to larger fields.

There were several constraints met due to the complexity of the project. Most of the issues met early on was the availability of a drone dataset, drone images found were not labeled and could not provide enough information visually whether the rice plant has a disease or not. It would be better to have even more images used for training the model however there were scarce datasets that fulfilled the requirements of the projects with most of them not containing any metadata at all. Using a dataset with geotagged smartphone images means that the assumption made here is that the drone will need to be flying at a low altitude to allow for better visual information to be captured. Other factors such as flight height and whether the propeller of the

drone can affect the stability of the leaves when images are taken could not be determined. Remote sensing data collected from NASA and it usually has a specific spatial resolution along with a value update of every certain number of days. For example, the MODIS Satellite has a low spatial resolution of 250 m. This will need to be taken into account when performing a drone flight pass through a field. The starting point and how the field will be presented in a grid must coincide with the information provided by remote sensing satellites. The dataset may have location and time data which allows for extraction of weather and remote sensing information; however, looking at the data distribution the data could not provide a day-by-day basis of information which could be really useful in learning how certain diseases progress. Finally, the model is based on rice datasets found in Taiwan meaning that this model may perform differently if given datasets from other regions with different climates.

6.3 System Quality Evaluation of Model Functions and Performance

The correctness of the model is measured and determined by the model's performance in correctly classifying the correct class (brown spot disease, leaf blast disease and healthy) when supplied with image, weather and remote sensing data. The model’s performance and correctness are shown in the table in the results sections underlining each model before and after further tuning to improve results. Run-time performance of the model is measured by utilizing the time function averaging it over 3 times. Since the application of the project is to run multiple images into the model to be able to provide a classification of disease presence in a field, the run-time performances for various tasks are shown in the Table below when dealing with 20 images as a benchmark.

Table 8

Run-time performance for a batch of 20 sample images with an average of 3 runs:

Feature	**Device Specs**	**Average Run Time (seconds)**
Extract Metadata from Images	MacBook Pro Processor: 2.3 GHz 8-Core Intel Core i9 Memory: 16GB 2667 MHz DDR4	0.1085 seconds
Obtaining Remote Sensing Data		34.4002 seconds
Obtaining Weather Data		1.4923 seconds
Combine Image Metadata, Remote Sensing and Weather Data		0.0832 seconds
Load hybrid DenseNet121 Model	Google Colab Pro V100 GPU	6.5606 seconds
Generate Predictions for all 20 instances		8.7212 seconds

6.4 System Visualization

The user-friendly interface starts with a login system allowing users to log in or new users to register. Upon successful login, a dashboard appears. Figure 47 displays the registration form for new users. Figure 48 displays a login page for existing users. After the registration of a new user, it gets stored in a database, which is shown in Figure 49.

Figure 47

Registration Form

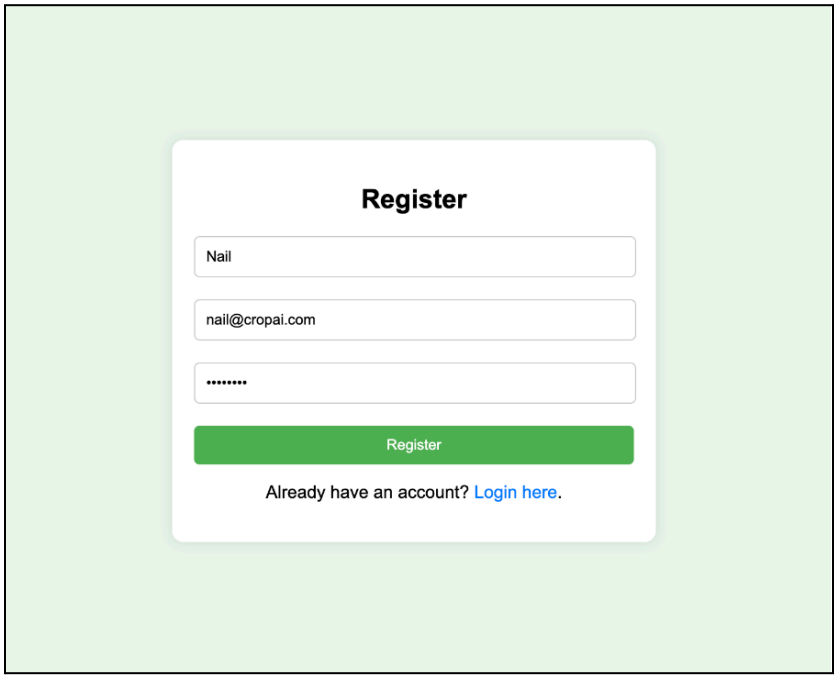


Figure 48

'User' table in the database

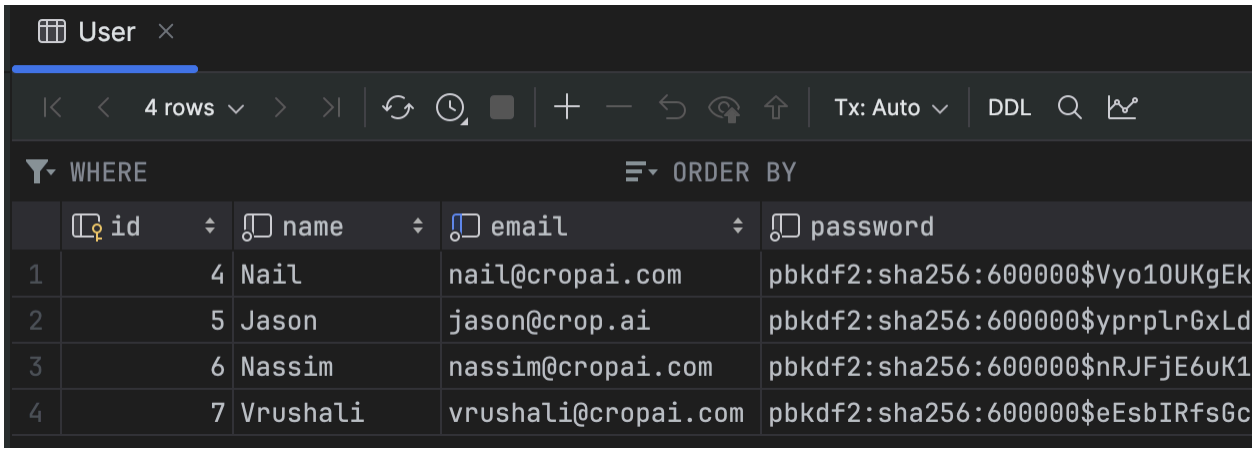
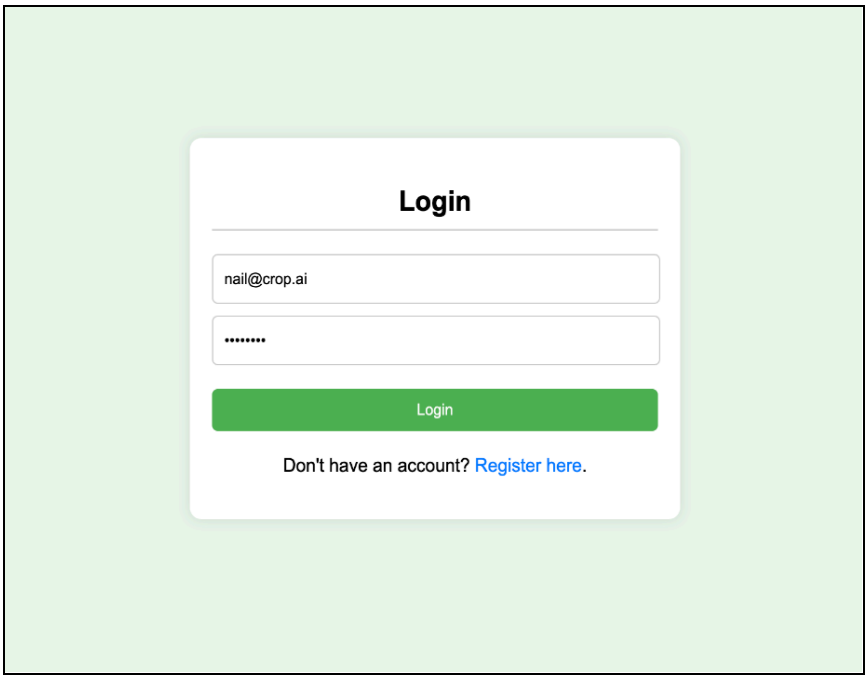


Figure 49

Login Form



After logging into the system, users are directed to the main page. There are 2 buttons: “Add Field”, “Upload Batch”. The first button allows the user to create a new crop field record. The second button allows the user to upload a new batch of images within the selected field. The main page is shown in Figure 50. The main page also displays the existing tables of fields and batches shown in Figure 51.

Figure 50

The web page header after successful Login

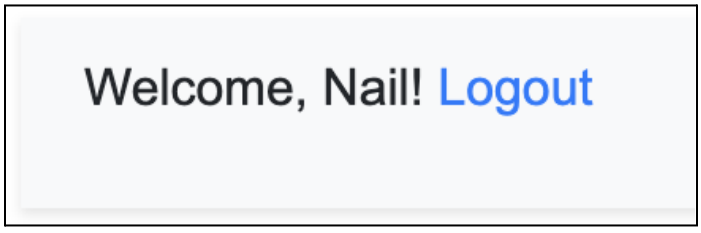
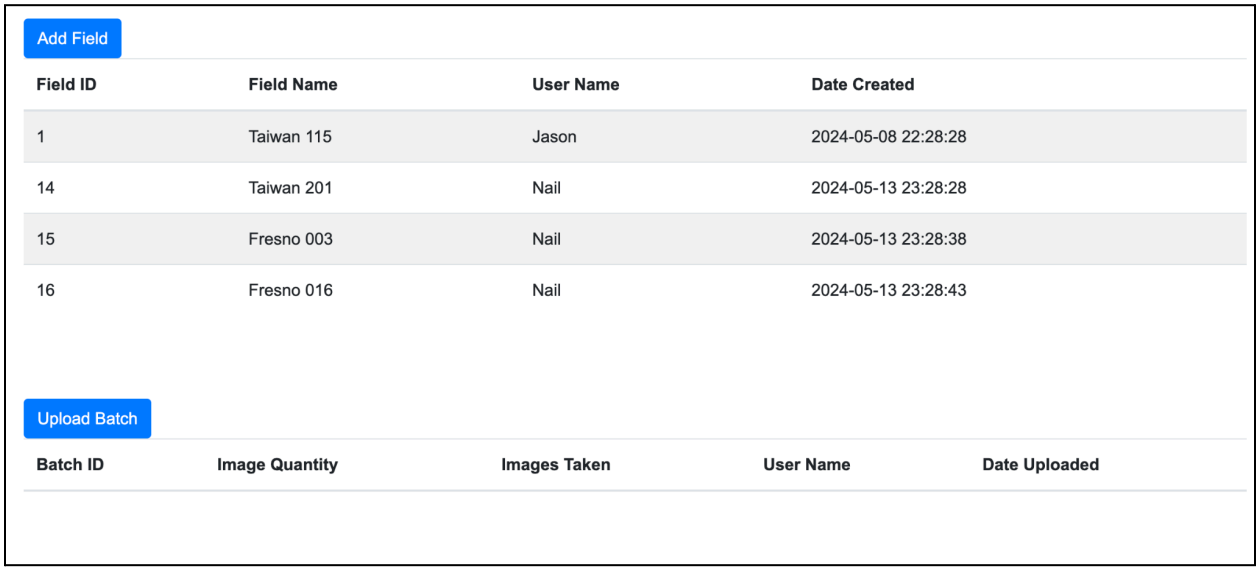


Figure 51

Existing tables of Fields and Batches



The user can add a new field in the "Add field" button shown in Figure 52. The database of the field table is shown in Figure 53.

Figure 52

Adding a new field

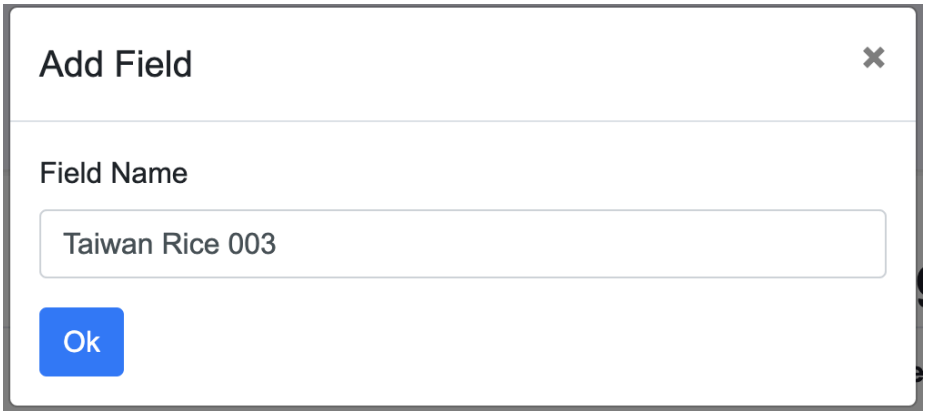


Figure 53

'Field' table in the database

Field

3 rows | Tx: Auto | DDL

WHERE | ORDER BY

	id	user_id	name	datetime
1	56	4	Taiwan Rice 001	2024-03-26 02:15:36.828077
2	57	4	Taiwan Rice 002	2024-03-26 02:15:50.251340
3	58	4	Taiwan Rice 003	2024-03-26 02:18:16.761158

To upload new images, the user clicks the "Upload Batch" button and selects images from the local user storage. Figure 54, Figure 55, and Figure 56 show the pop-up form, images selected, and database of the batch table respectively.

Figure 54

Adding a new Batch of images along with field area dimensions

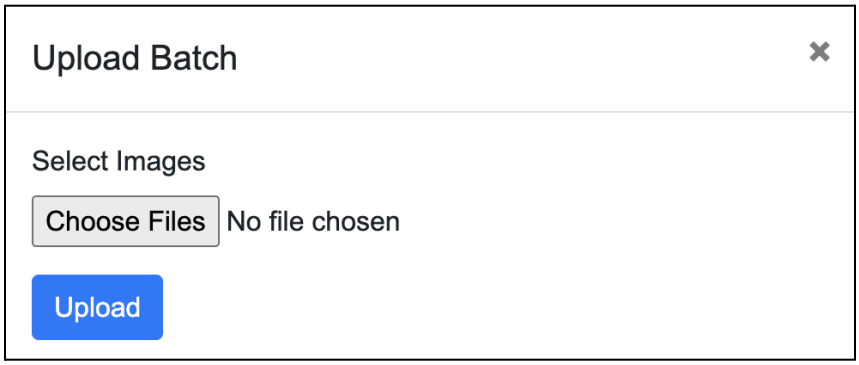
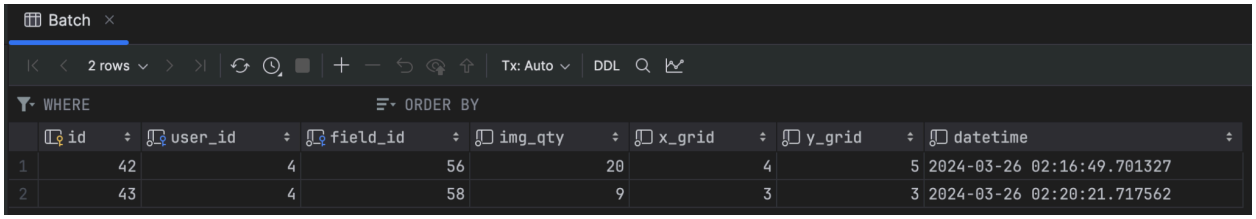


Figure 55

Selecting image files from the local storage

Figure 56

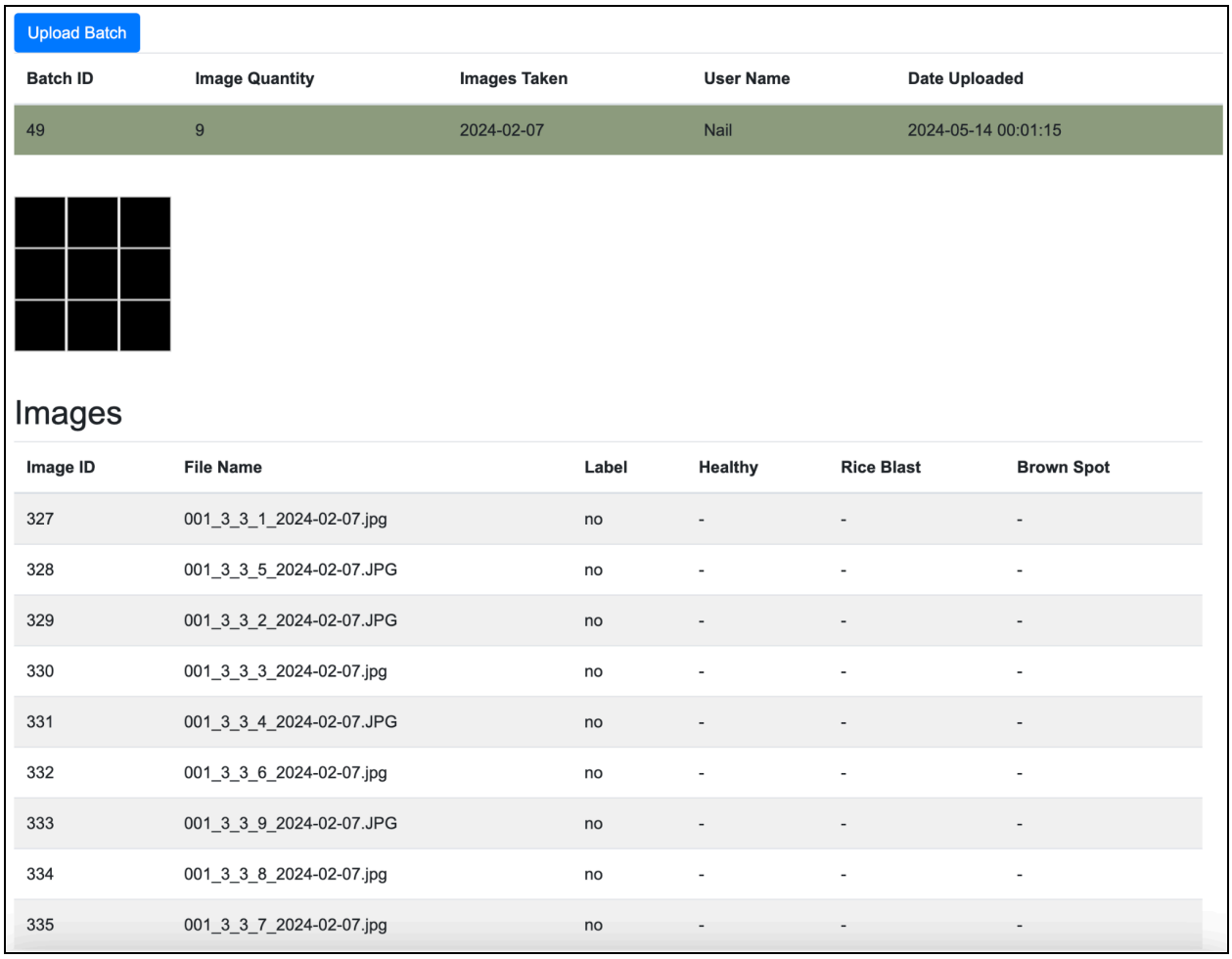
'Batch' table in the database



After uploading the images in the batches, the main page displays the details of the images. Figure 57 shows the main page and Figure 58 shows the image database.

Figure 57

The Batch for specific Field was uploaded (state before classification, labels are 'no')



Upload Batch

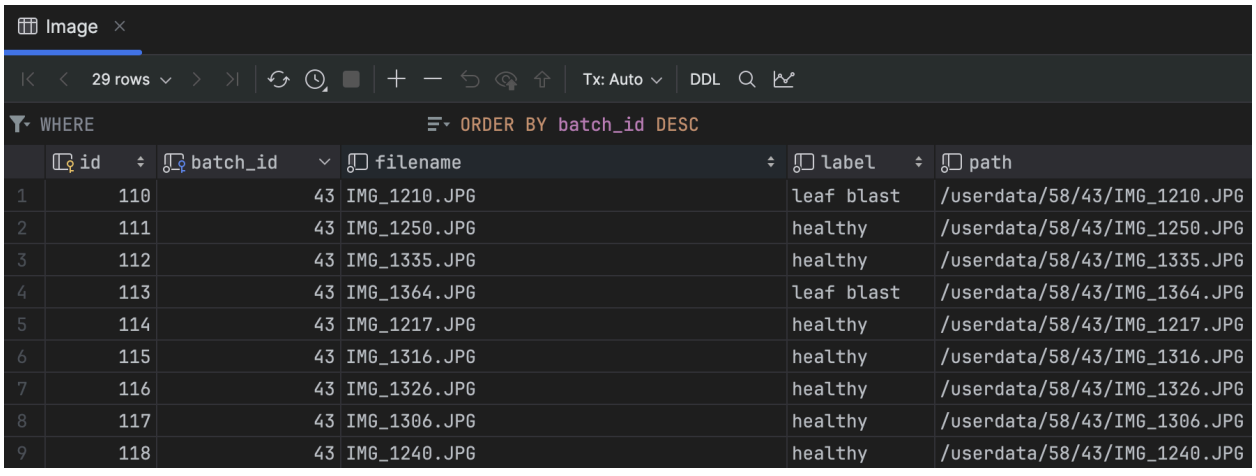
Batch ID	Image Quantity	Images Taken	User Name	Date Uploaded
49	9	2024-02-07	Nail	2024-05-14 00:01:15

Images

Image ID	File Name	Label	Healthy	Rice Blast	Brown Spot
327	001_3_3_1_2024-02-07.jpg	no	-	-	-
328	001_3_3_5_2024-02-07.JPG	no	-	-	-
329	001_3_3_2_2024-02-07.JPG	no	-	-	-
330	001_3_3_3_2024-02-07.jpg	no	-	-	-
331	001_3_3_4_2024-02-07.JPG	no	-	-	-
332	001_3_3_6_2024-02-07.jpg	no	-	-	-
333	001_3_3_9_2024-02-07.JPG	no	-	-	-
334	001_3_3_8_2024-02-07.jpg	no	-	-	-
335	001_3_3_7_2024-02-07.jpg	no	-	-	-

Figure 58

'Image' table in the database



Image

29 rows | Tx: Auto | DDL

WHERE ORDER BY batch_id DESC

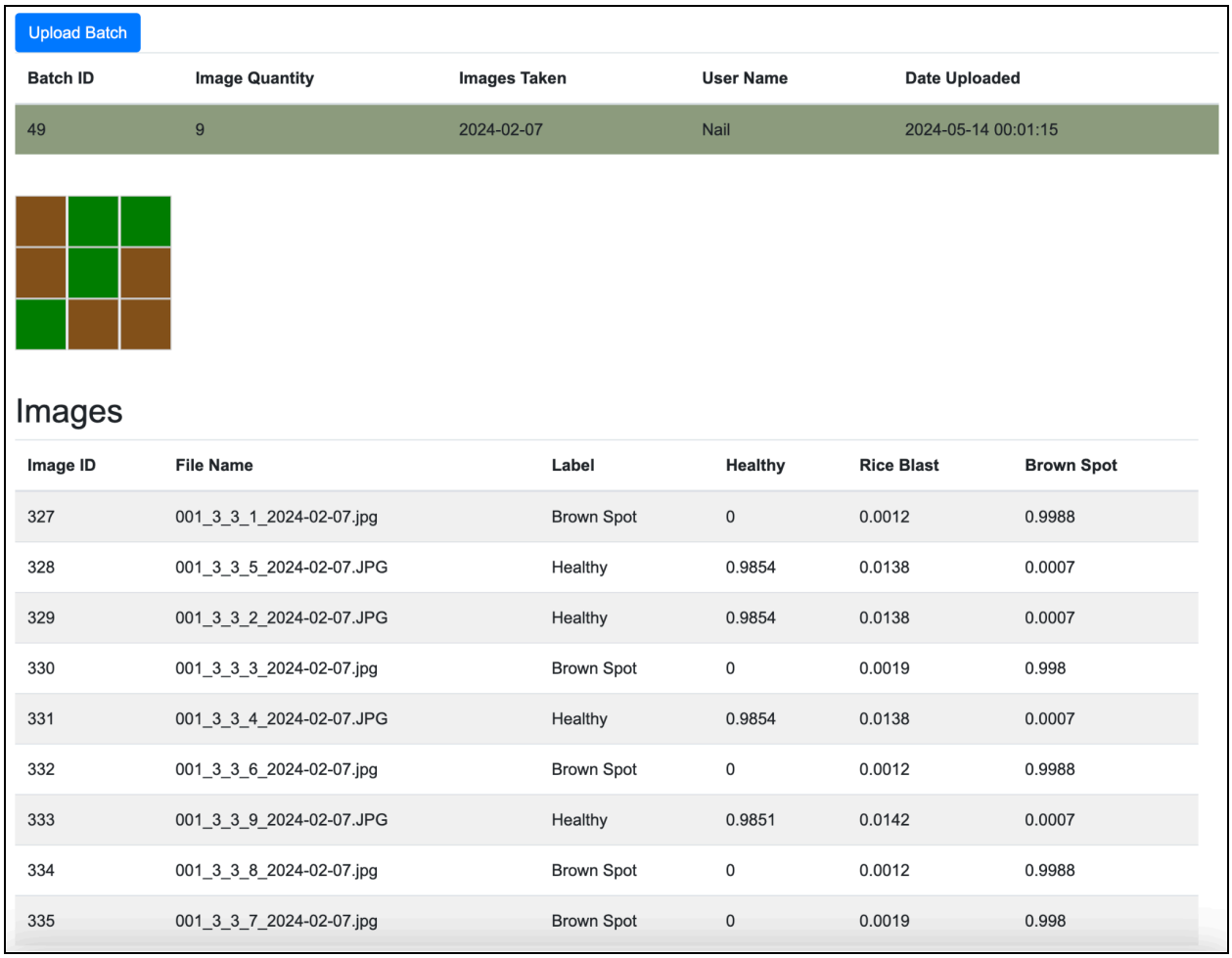
	id	batch_id	filename	label	path
1	110	43	IMG_1210.JPG	leaf blast	/userdata/58/43/IMG_1210.JPG
2	111	43	IMG_1250.JPG	healthy	/userdata/58/43/IMG_1250.JPG
3	112	43	IMG_1335.JPG	healthy	/userdata/58/43/IMG_1335.JPG
4	113	43	IMG_1364.JPG	leaf blast	/userdata/58/43/IMG_1364.JPG
5	114	43	IMG_1217.JPG	healthy	/userdata/58/43/IMG_1217.JPG
6	115	43	IMG_1316.JPG	healthy	/userdata/58/43/IMG_1316.JPG
7	116	43	IMG_1326.JPG	healthy	/userdata/58/43/IMG_1326.JPG
8	117	43	IMG_1306.JPG	healthy	/userdata/58/43/IMG_1306.JPG
9	118	43	IMG_1240.JPG	healthy	/userdata/58/43/IMG_1240.JPG

After a Batch has been uploaded, the Data Pipeline runs the Cloud Function to extract metadata from the batch images. Using the extracted metadata other Cloud Functions fetch weather and remote sensing data. Finally, the consolidated dataset is prepared and along with the images passed to the hybrid model for predictions.

The Hybrid Model classified images into the following categories: Healthy, Leaf Blast (disease), and Brown Spot (disease). The grid replicates the rice crop farm where the disease part of the farm is indicated with red color and the healthy part with green color. Figure 59 shows the result.

Figure 59

The Batch was processed by the Hybrid Model (state after classification, the labels have assigned classes)

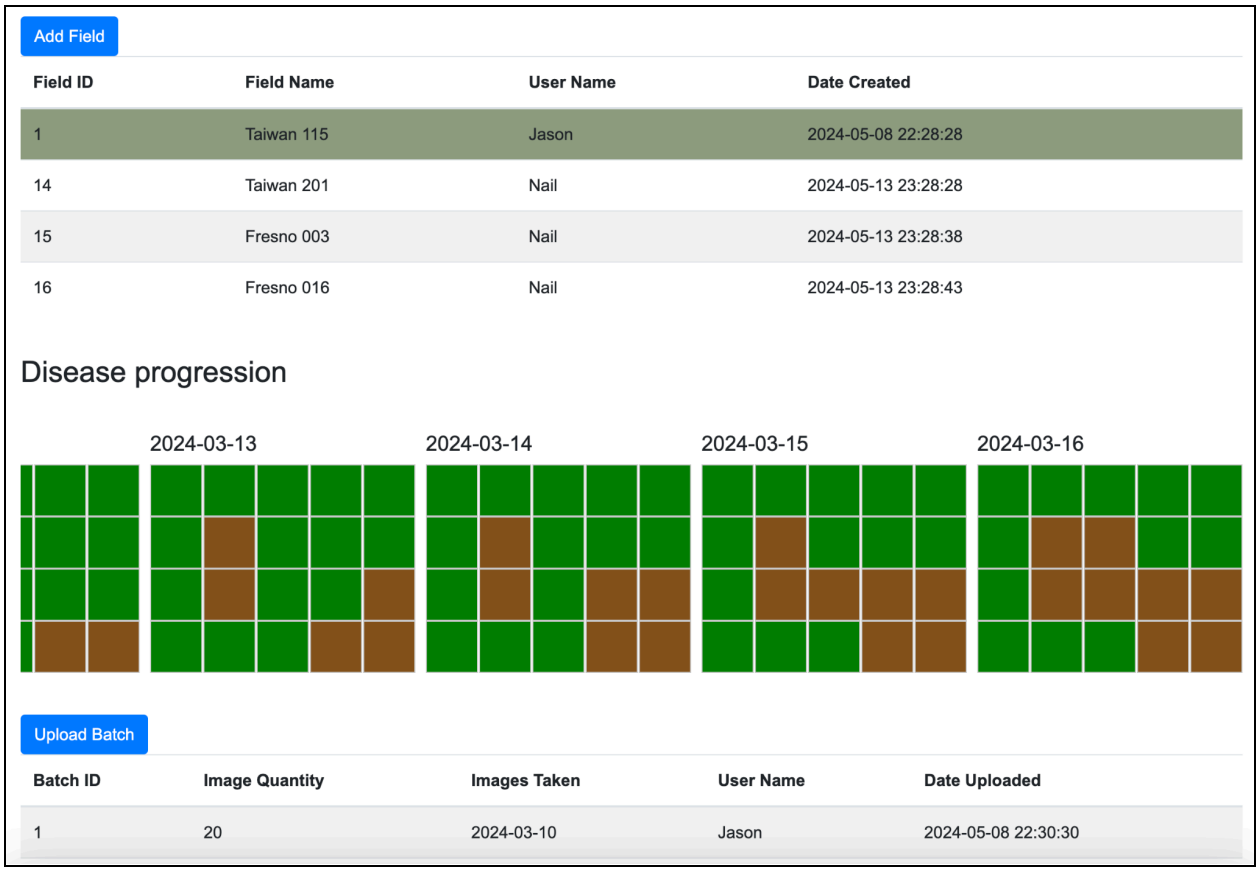


A new batch of images is uploaded to the database.

In the same way multiple batches for the field can be uploaded. Each Batch represents a specific date the images in the batch are taken. All batches go through the hybrid model and get classified into corresponding labels. The progression of disease can be seen in figure 60.

Figure 60

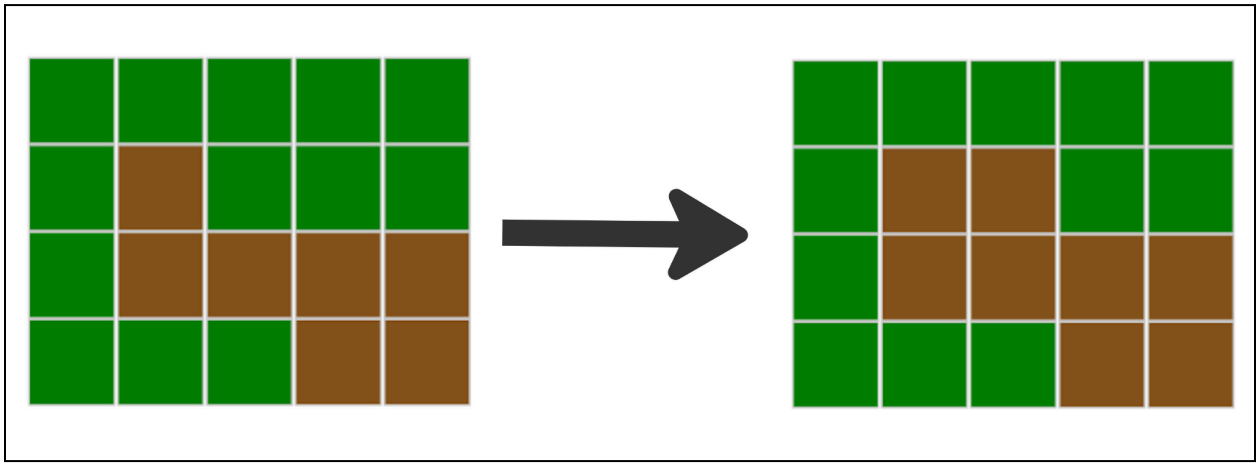
Disease progression for the multiple Batches for the same Field.



By analyzing image batches from one area of the field over time, a user can monitor the progression of the disease.

Figure 61

Disease progression



The grid above can show an example in disease progression in the field crop. By performing crop disease detection and classification from day 1 and day 2, by using the model and system, information regarding the crop location and disease can be localized and gathered. The grid above shows a sample change in crop health in the field, brown being disease (brown spot) and green being healthy. Performing detection on a time interval allows users or farmers to learn the disease progression in a very large field and allows farmers to create data-driven decisions on where and what they can do to prevent further spreading of the disease and mitigate the damage. The user interface will be improved in the future and the following screenshots from the system currently provide proof of functionality.

7. Conclusion

7.1 Summary

This research project involved the use of image data and tabular data, which include remote sensing and weather data, for agricultural health monitoring in water-intensive crops, in this case, rice. The project developed a range of machine learning models, including visual-only models and hybrid models that integrated both the image and numerical data. The use of ensemble methods through soft and majority voting, along with extensive fine-tuning, were applied to enhance model accuracy and robustness. Among these, the hybrid model utilizing a DenseNet121 base in the visual branch emerged as the most effective, due to its performance and model size compared to all other models at 98.4% accuracy. Due to these factors, this model was selected for deployment in the final system. In addition, an interactive web portal application was developed to visually represent the predicted disease status and its geospatial distribution in the field, offering an interactive tool for farmers to understand and manage crop health more effectively. This portal allows farmers to make informed decisions regarding crop management and disease treatment based on the comprehensive data provided by the applied hybrid model.

7.2 Benefits and Shortcomings

The project's primary benefit lies in its advanced approach to disease management in crops, significantly enhancing yield and reducing losses due to timely and accurate disease detection. The use of a combination of image and tabular data allows for a unique approach compared to that of purely image and purely tabular. The development of an interactive web portal enables convenient access to critical data, allowing farmers to interact with and understand the health of their crops in real-time, thereby facilitating informed crop management and disease

treatment decisions. However, there are several shortcomings. The system is currently limited to classifying only three classes (two disease and one healthy class) and relies on pre-existing imagery not captured by drones, which may not fully replicate the diversity of field conditions. Measures have been taken in the image augmentation steps to account for this; however, a drone dataset would be best for this scenario. Additionally, the focus on a single region (Taiwan) might restrict the model's scalability and adaptability to varied agricultural environments, potentially limiting its effectiveness in global applications as other parts of the world have varying climates.

7.3 Potential System and Model Applications

The system's flexible design is conducive to adaptation for a range of agricultural monitoring tasks. While initially validated on rice crops, the approach can be generalized to other crop types and a broader spectrum of diseases, enhancing its utility across the agricultural sector. Integrating actual drone-captured data could further revolutionize crop management, making the system more dynamic and responsive. Beyond agriculture, the model's ability to integrate and analyze different kinds of data presents opportunities for applications in domains such as environmental monitoring, where analysis derived from both image and satellite/weather data may be beneficial. The better performance of the hybrid model means that it may have potential to be applied to other fields which rely on image classification such as the medical field. Machine learning image classification is used in disease detection on a cellular level, integration of other tabular factors which are known to be a known factor of the disease can be applied. This may include patient information such as oxygen concentration in blood or white blood cell count. Hybrid models can then be produced using this information and hopefully be able to yield better results when compared to its image-only model counterpart.

7.4 Experience and Lessons Learned

Throughout the project, valuable experiences were gained in the realms of data integration and model development. Data exploration was a very important experience and lesson learned as there were difficulties in trying to collect a dataset which included labeled images of crop diseases as well as weather and remote sensing data. However, after further data exploration, some datasets contained images with metadata; this information contained the time and location of each image and using API's allowed the extraction of weather and remote sensing data. Key lessons include the critical importance of precise data alignment and the challenges associated with integrating mixed data types—image and numerical data—in a unified analytical framework. The project also underscored the need for models to be both accurate and robust, capable of handling the variability and unpredictability of real-world agricultural environments. The development of the web portal highlighted the importance of user-centric, user-friendly design and accessibility, ensuring that the data provided by the model is understandable and actionable for farmers.

7.5 Recommendations for Future Work

Future developments should aim to include actual drone-captured imagery to assess the conceptual model's efficacy in real field conditions actively. Expanding the model to detect a wider range of diseases across more crop types would broaden the system's applicability and utility. Incorporating data from additional regions around the world would not only help in refining the model's accuracy but also enhance its applicability and relevance on a global scale, making the system usable by a broader audience. Enhancing the web portal with additional interactive features could significantly improve its functionality, making it an even more

powerful tool for decision-making in agriculture. Collaborations with agricultural researchers and technologists could drive further innovations and refinements in the model.

7.6 Contributions and Impacts on Society

This project has the potential to deliver substantial contributions across multiple dimensions of society. Economically, it facilitates significant cost savings and yield improvements for farmers, enhancing the profitability and sustainability of agricultural practices. Educationally, it serves as a pioneering example of practical machine learning applications in agriculture, and especially the potential applications of using tabular information to supplement and boost image classification tasks. Socially, by improving the accuracy and timeliness of disease management, it directly supports food security and stability. The project also has cultural impacts, as it demonstrates the beneficial integration of advanced technologies in traditional farming practices, potentially leading to broader technological adoption across diverse agricultural communities globally and fostering a more sustainable and technologically advanced approach to smart agriculture.

References

A. Soni, J. K. Soni and S. Hota, "Detection of Multiple Crop Diseases using Image Processing Techniques," *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, Waltham, MA, USA, 2021, pp. 1-6, doi: 10.1109/HPEC49654.2021.9622812.

B. Gupta, S. Bomble, O. Gaikar, S. Chalekar, S. R. Vispute and K. Rajeswari, "Convolutional Neural Networks for Detection of Crop Diseases and Weed," *2022 6th International Conference On Computing, Communication, Control And Automation (ICCUBEA*, Pune, India, 2022, pp. 1-5, doi: 10.1109/ICCUBEA54992.2022.10010772.

C. Hu, Y. Liu, J. Xiang and Z. Chen, "Detection of Crop Leaf Diseases Using Gl-Cgan Based Data Augmentation," *2021 International Conference on Machine Learning and Cybernetics (ICMLC)*, Adelaide, Australia, 2021, pp. 1-5, doi: 10.1109/ICMLC54886.2021.9737251.

Duong-Trung, N., Quach, L.-D., & Nguyen, C.-N. (2019). Learning deep transferability for several agricultural classification problems. International Journal of Advanced Computer Science and Applications (IJACSA). https://doi.org/10.14569/IJACSA.2019.0100107

FAO. (2021, June 2). *Climate change fans spread of pests and threatens plants and crops, new FAO Study*. FAO. https://www.fao.org/news/story/en/item/1402920/icode/

Garcia-Garcia, A., Orts, S., Oprea, S., Villena-Martinez, V., & Rodríguez, J.G. (2017). A Review on Deep Learning Techniques Applied to Semantic Segmentation. *ArXiv,* https://arxiv.org/abs/1704.06857

H. Ren, W. Chen, P. Liu, R. Tang, J. Wang and C. Ge, "Multi-scale Parallel Fusion Convolution

Network for Crop Disease Identification," *2023 8th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA)*, Chengdu, China, 2023, pp. 481-485, doi: 10.1109/ICCCBDA56900.2023.10154853.

Jisan, M.(2023) Rice Leafs Disease Dataset. Kaggle. 2023 *https://www.kaggle.com/datasets/maimunulkjisan/rice-leaf-dataset-from-mendeley-data*

Nguyen T-H, Nguyen T-N, Ngo B-V. A VGG-19 Model with Transfer Learning and Image Segmentation for Classification of Tomato Leaf Disease. *AgriEngineering*. 2022; 4(4):871-887. https://doi.org/10.3390/agriengineering4040056

Ou. J and Chen. C. (2022). Rice leaf diseases in Taiwan [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DS/2012808

P. Jha, D. Dembla and W. Dubey, "Comparative Analysis of Crop Diseases Detection Using Machine Learning Algorithm," *2023 Third International Conference on Artificial Intelligence and Smart Energy (ICAIS)*, Coimbatore, India, 2023, pp. 569-574, doi: 10.1109/ICAIS56108.2023.10073831.

Rosebrock, A. (2021, July 9). *Keras: Multiple inputs and mixed data*. PyImageSearch. https://pyimagesearch.com/2019/02/04/keras-multiple-inputs-and-mixed-data/

S. Harika, G. Sandhyarani, D. Sagar and G. V. S. Reddy, "Image-based Black Gram Crop Disease Detection," *2023 International Conference on Inventive Computation Technologies (ICICT)*, Lalitpur, Nepal, 2023, pp. 529-533, doi: 10.1109/ICICT57646.2023.10134027.

Shan, L. Y. (2023, April 19). *Global rice shortage is set to be the biggest in 20 years*. CNBC. https://www.cnbc.com/2023/04/19/global-rice-shortage-is-set-to-be-the-largest-in-20-years-heres-why.html

Stephen, A., Punitha, A. & Chandrasekar, A. Designing self attention-based ResNet architecture for rice leaf disease classification. *Neural Comput & Applic* 35, 6737–6751 (2023). https://doi.org/10.1007/s00521-022-07793-2

Tan, C., & Shan, L. Y. (2023, August 22). *Rice prices soar, fanning fears of food inflation spike in Asia*. CNBC. https://www.cnbc.com/2023/08/22/asia-food-inflation-fears-rise-as-rice-prices-surge.html#:~:text=Rice%20prices%20soar%2C%20fanning%20fears%20of%20food%20inflation%20spike%20in%20Asia,-Published%20Mon%2C%20Aug&text=Rice%20prices%20surged%20to%20their,ban%20and%20adverse%20weather%20conditions

Z. Saeed, A. Raza, A. H. Qureshi and M. Haroon Yousaf, "A Multi-Crop Disease Detection and Classification Approach using CNN," *2021 International Conference on Robotics and Automation in Industry (ICRAI)*, Rawalpindi, Pakistan, 2021, pp. 1-6, doi: 10.1109/ICRAI54018.2021.9651409.

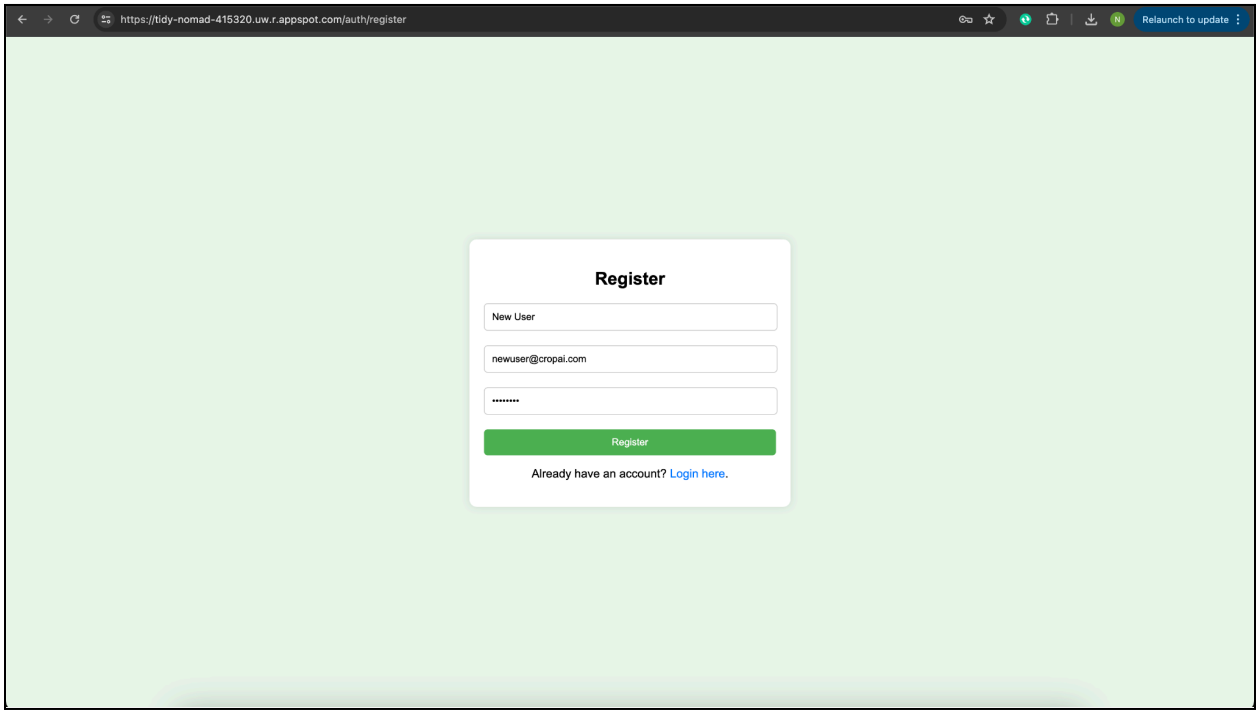
Appendices

Appendix A – System Testing

Use Case 1: Registration Page

Figure 62

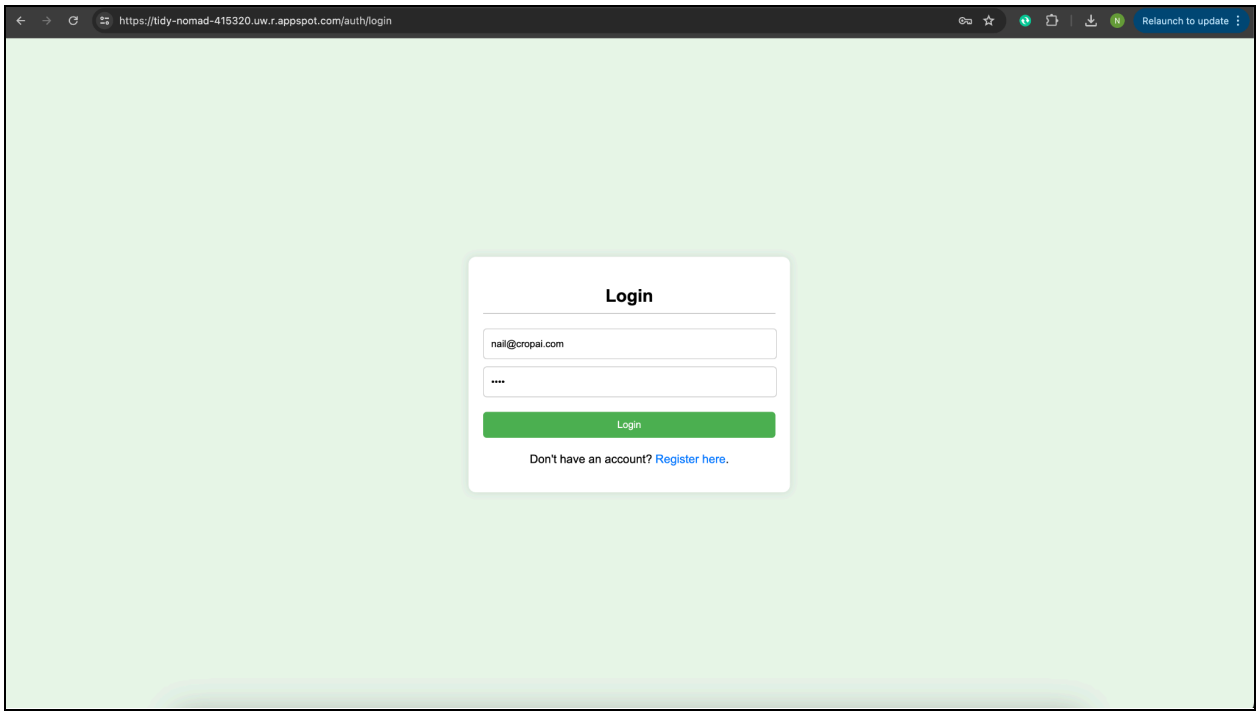
Register Form to add a new user to the web application



Use Case 2: Login Page

Figure 63

Login Form to get access to the web application functionality

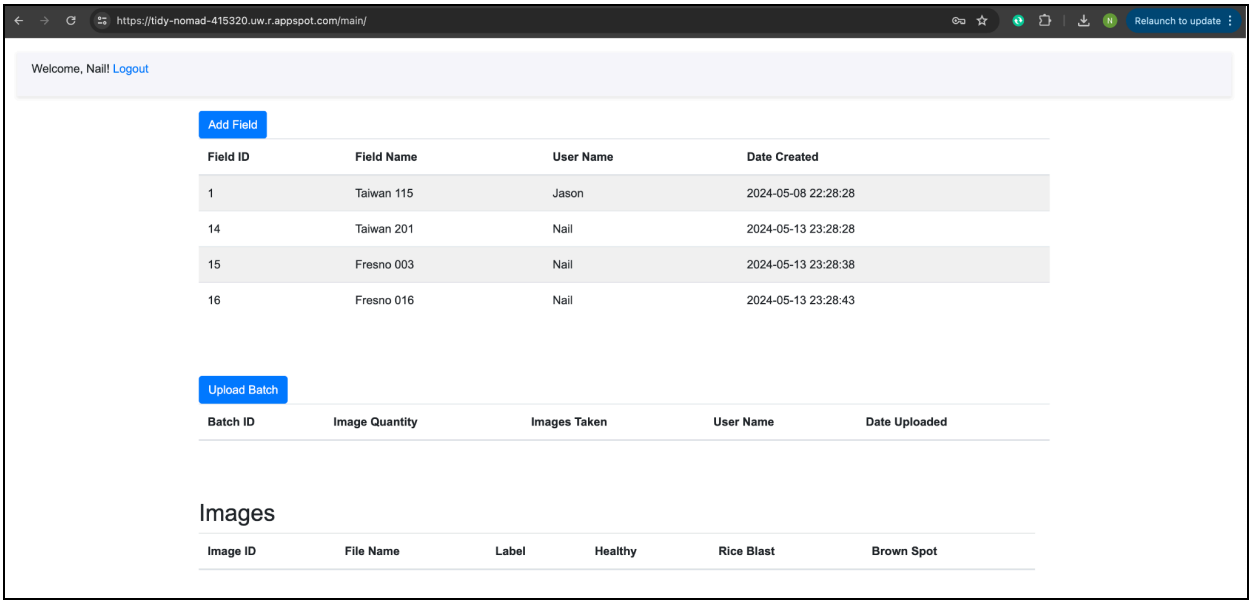


Use Case 3: Home Page

The home page consists of Fields, Batches and Images tables.

Figure 64

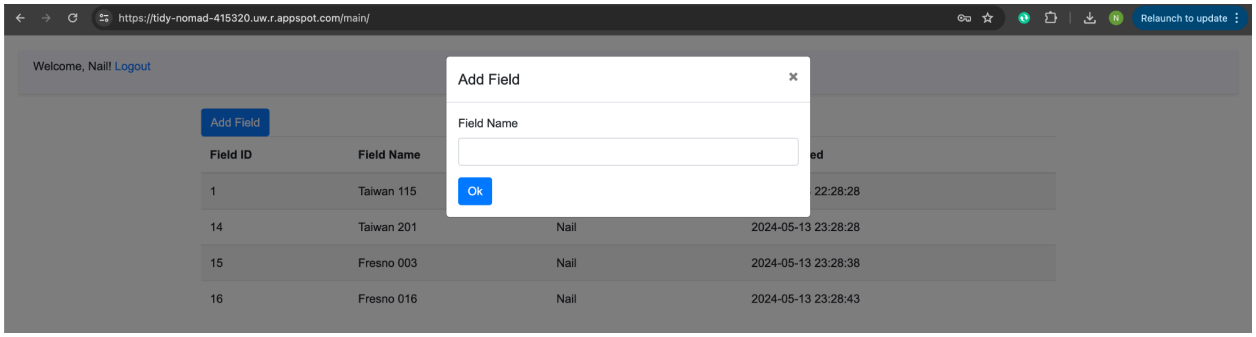
Water-Intensive Crop Disease Detection and Classification Home Page



Use Case 4: Adding a Field

Figure 65

Input a new Field into the system



Use Case 5: Uploading a Batch

Figure 66

Uploading images Batch for the selected Field

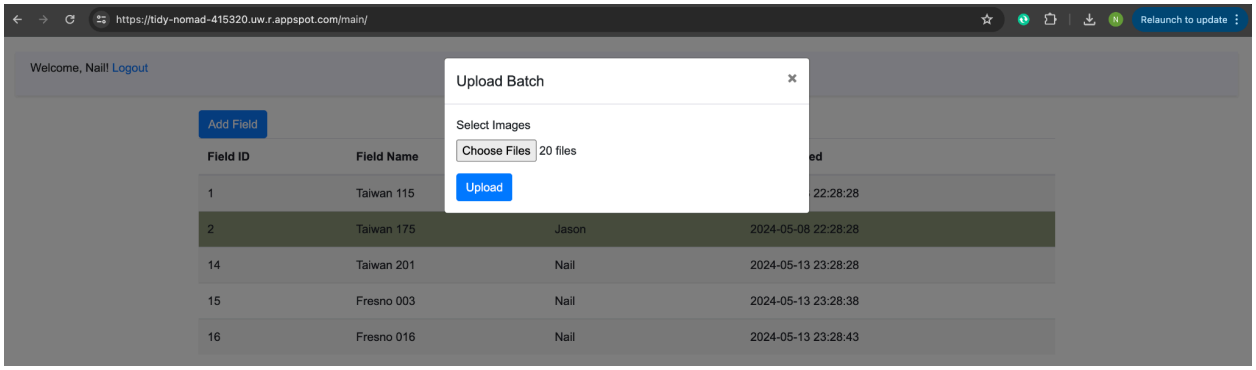
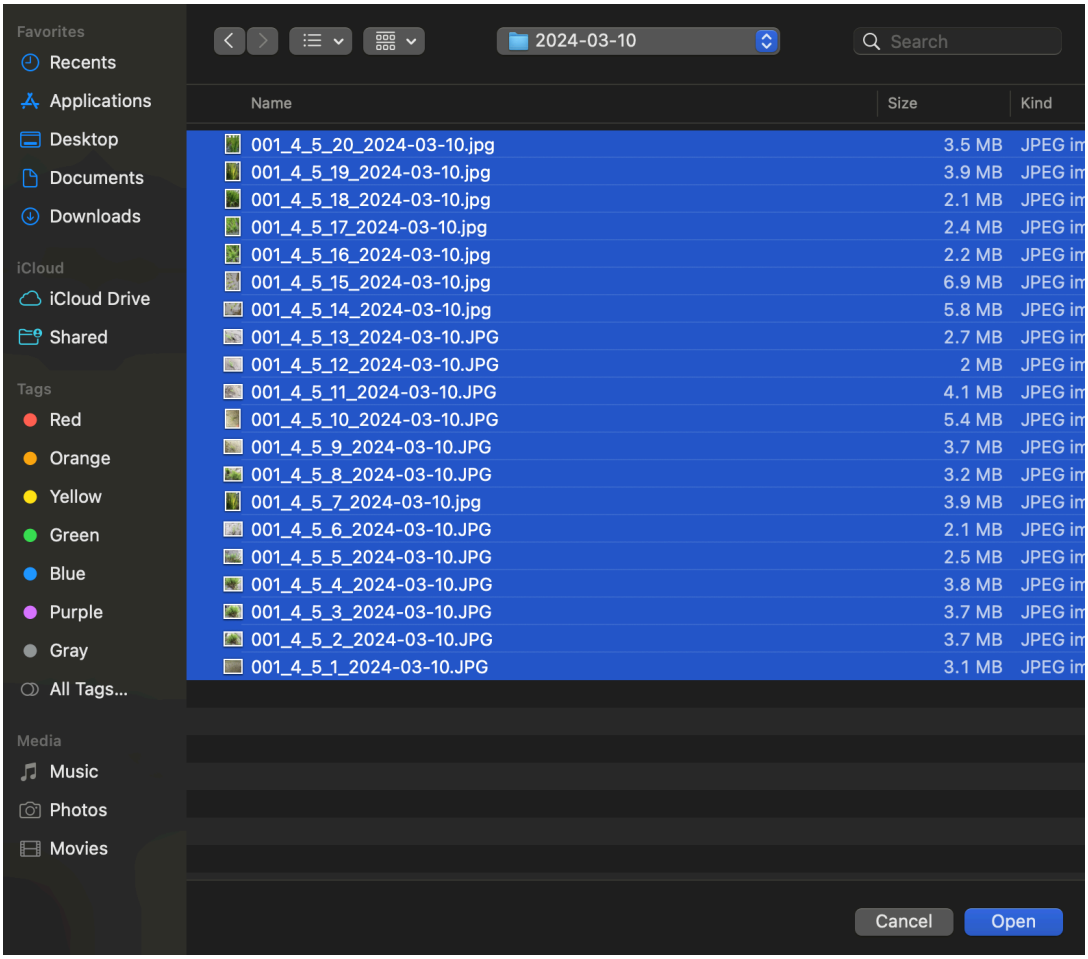


Figure 67

Selecting images from the user's storage

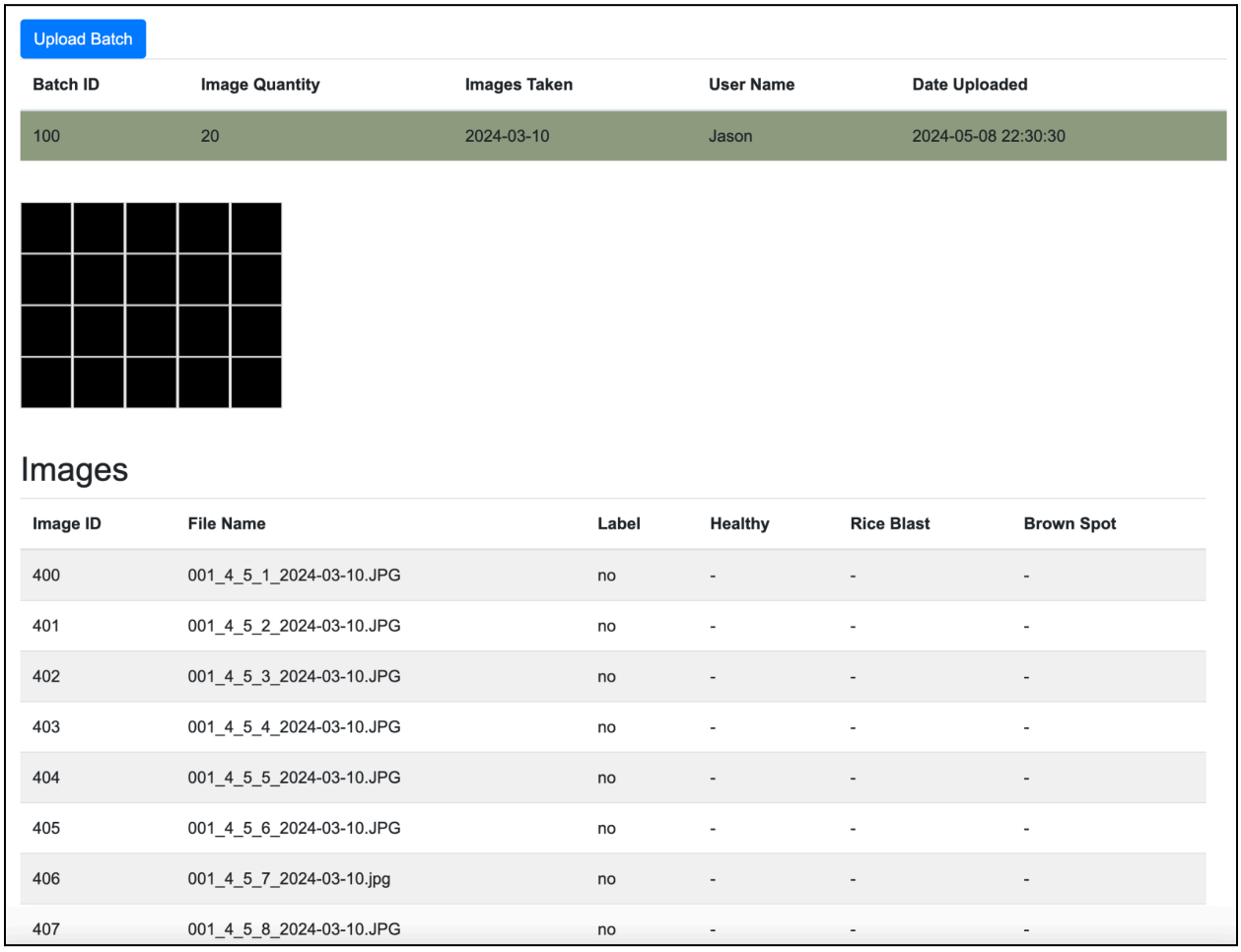


Use Case 6: Data preprocessing

After the Batch is uploaded the Data pipeline is triggered to run scripts to extract image metadata, fetch weather and remote sensing data, and consolidate datasets for the hybrid model.

Figure 68

Metadata extraction, weather and remote sensing data fetching are in progress.

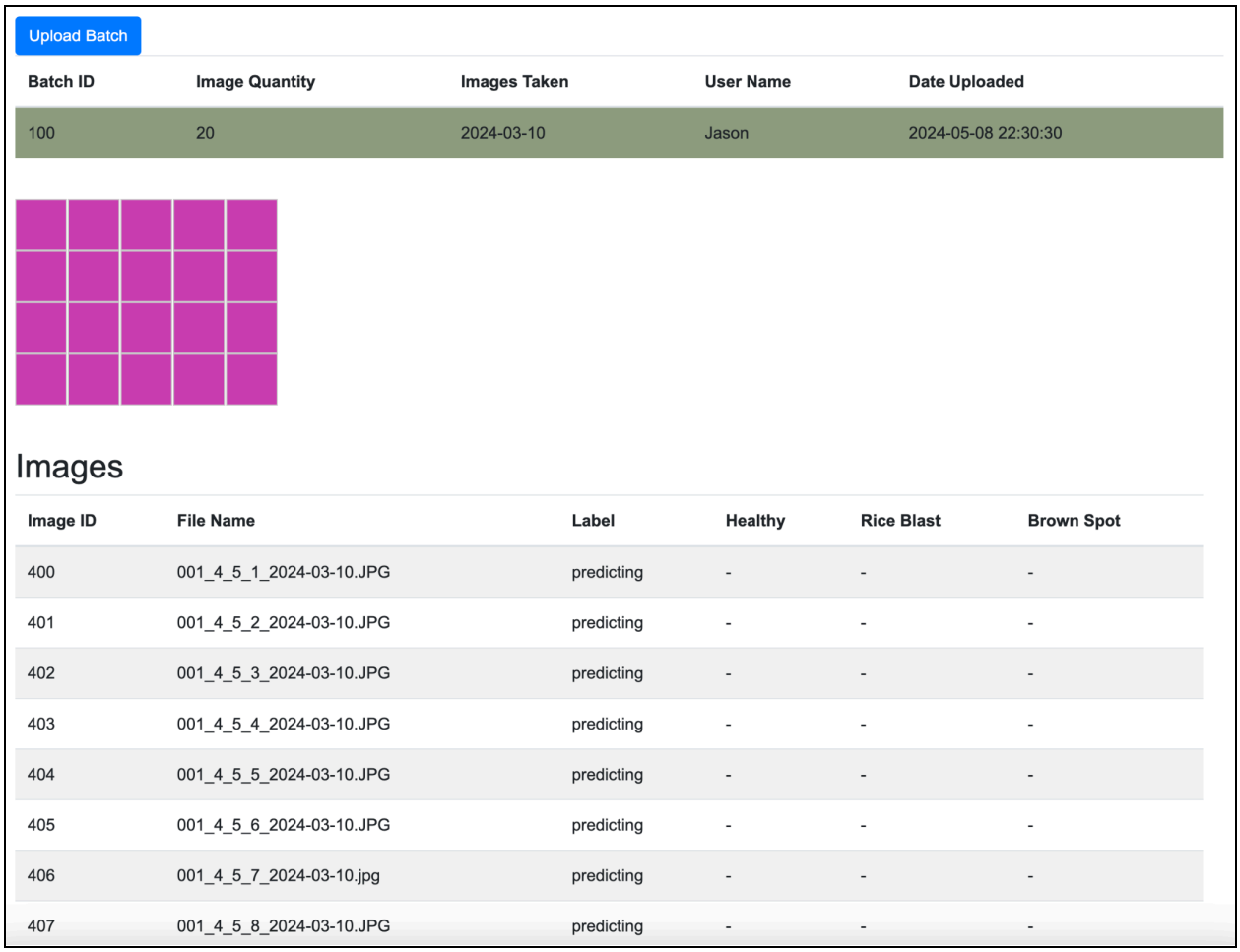


Use Case 7: Prediction

The consolidated dataset and batch images are passed to the hybrid model for prediction.

Figure 69

Hybrid Model prediction is in progress



Use Case 8: Predictions are made, Labels and Confidence Levels are assigned

The model outputs prediction JSON and CSV files, whose contents are written to the database. The Images table is being updated.

Figure 70

Observing the assigned labels and confidence levels after the hybrid model prediction

Images

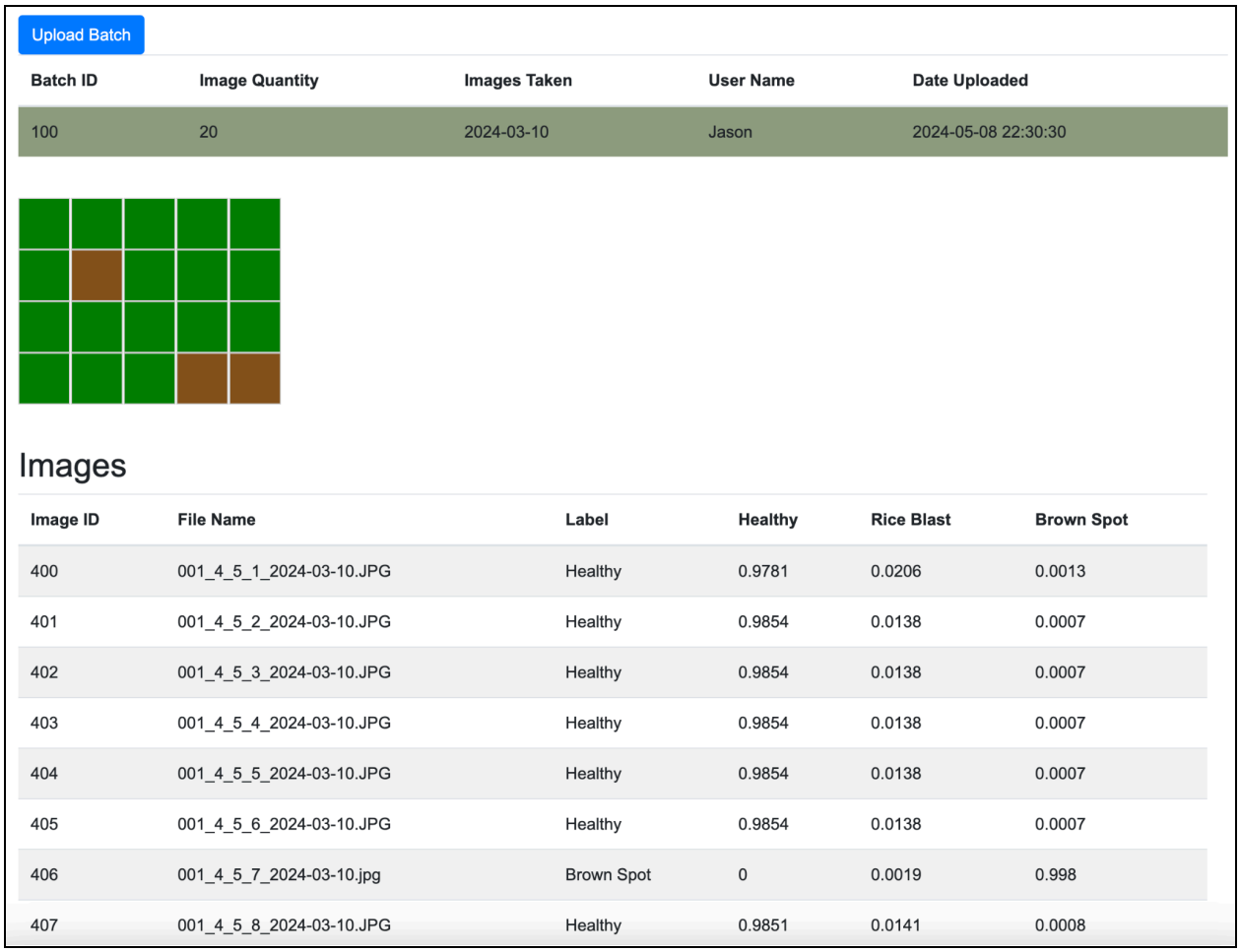
Image ID	File Name	Label	Healthy	Rice Blast	Brown Spot
400	001_4_5_1_2024-03-10.JPG	Healthy	0.9781	0.0206	0.0013
401	001_4_5_2_2024-03-10.JPG	Healthy	0.9854	0.0138	0.0007
402	001_4_5_3_2024-03-10.JPG	Healthy	0.9854	0.0138	0.0007
403	001_4_5_4_2024-03-10.JPG	Healthy	0.9854	0.0138	0.0007
404	001_4_5_5_2024-03-10.JPG	Healthy	0.9854	0.0138	0.0007
405	001_4_5_6_2024-03-10.JPG	Healthy	0.9854	0.0138	0.0007
406	001_4_5_7_2024-03-10.jpg	Brown Spot	0	0.0019	0.998
407	001_4_5_8_2024-03-10.JPG	Healthy	0.9851	0.0141	0.0008
408	001_4_5_9_2024-03-10.JPG	Healthy	0.9862	0.0131	0.0007
409	001_4_5_10_2024-03-10.JPG	Healthy	0.9862	0.0131	0.0007
410	001_4_5_11_2024-03-10.JPG	Healthy	0.9799	0.0189	0.0012
411	001_4_5_12_2024-03-10.JPG	Healthy	0.9799	0.0189	0.0012
412	001_4_5_13_2024-03-10.JPG	Healthy	0.9292	0.063	0.0079
413	001_4_5_14_2024-03-10.jpg	Healthy	0.9128	0.0869	0.0003
414	001_4_5_15_2024-03-10.jpg	Healthy	0.694	0.304	0.002
415	001_4_5_16_2024-03-10.JPG	Healthy	0.966	0.0339	0.0001
416	001_4_5_17_2024-03-10.jpg	Healthy	0.966	0.0339	0.0001
417	001_4_5_18_2024-03-10.jpg	Healthy	0.966	0.0339	0.0001
418	001_4_5_19_2024-03-10.jpg	Brown Spot	0	0.0019	0.998
419	001_4_5_20_2024-03-10.jpg	Brown Spot	0	0.0012	0.9988

Use Case 9: Observing the Batch Grid after predictions are made

After the images in the Batch have been updated by the prediction JSON file, the Grid becomes available to the user. The sector color indicates the label: green - healthy, brown - brown spot, red - rice blast.

Figure 71

Obser*ving the Batch Grid - representation of the physical crop field's sectors, colored with disease label*

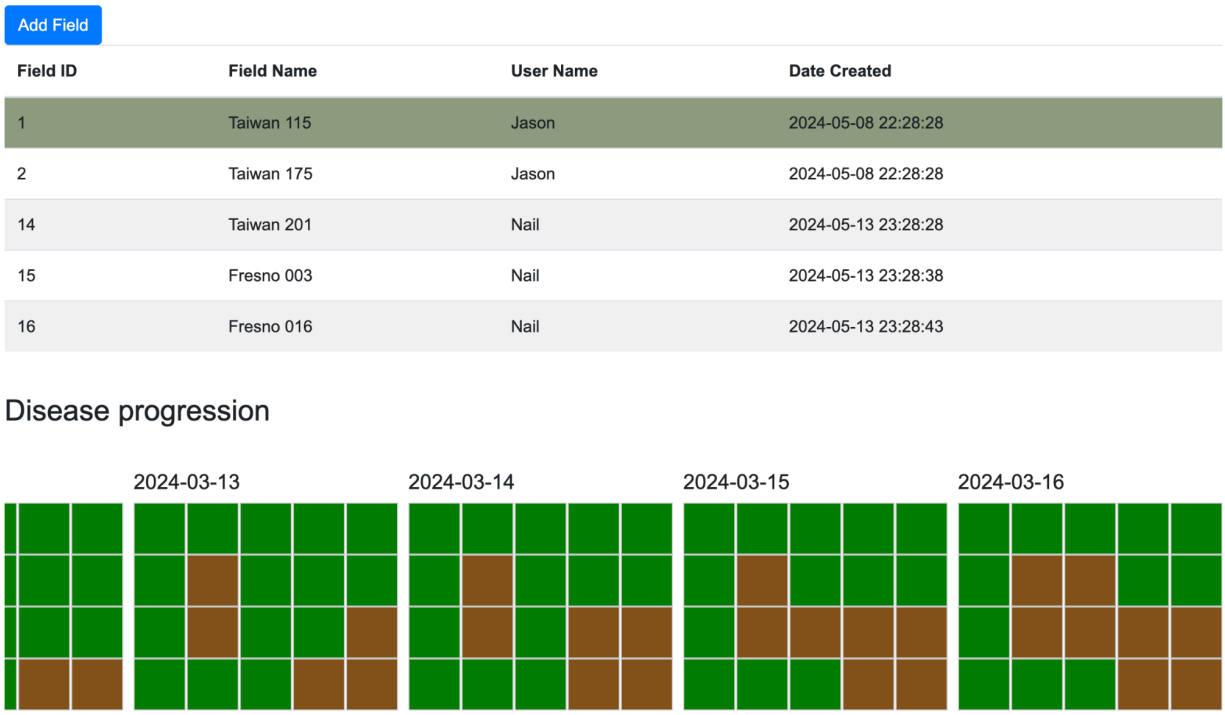


Use Case 10: Observing Disease Progression Grids

After multiple Batches are uploaded and predictions are made the Disease Progression Grids become available to the user. Grids are sorted from left to right in Date Uploaded ascending order.

Figure 72

Observing the Disease Progression Grids for the selected Field, where each Grid represents an uploaded Batch.



Appendix B – Project Data Source and Management Store

The comprehensive datasets used across modules of this project are provided in the below drive link. The dataset comprises three distinct sets: training (along with the augmented images), validation and test data sets, for both the image and numerical data. All of the additional numerical datasets used in modeling are also stored.

Appendix B can be located at the following Google Drive link:

[redacted]

Appendix C – Project Program Source Library, Presentation, and Demonstration

A final package contains Workbook 1, Workbook 2, Final Reports, Presentation 1, Presentation 2, Final Presentation, Demo Video 1, Demo Video 2, Final Demo Video, Final

Presentation Recording, Preprocessing scripts, Prediction scripts, Trained model artifacts, Web Application modules, and other project source code files.

Appendix C can be located at the following Google Drive link: