

Water-Intensive Crop Disease Detection and Classification Using Machine Learning

Nassim Ali-Chaouche, Jason Widjaja Djuandy,
Nail Enikeev, Vrushali Pravin Menche

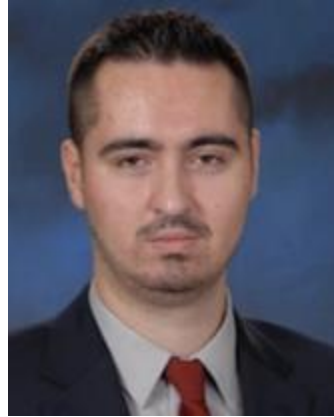
Department of Applied Data Science,
San Jose State University
DATA 298B: MSDA Project II
Dr. Jerry Gao



Team 1



Jason Djuandy



Nassim Ali-
Chaouche



Nail Enikeev



Vrushali Menthe

Table of Contents

- 🌾 Introduction
- 🌾 Data Engineering
- 🌾 Machine Learning Models
- 🌾 Machine Learning Results and Case Study
- 🌾 - System Requirements Analysis and Design
- Web System Design and Development
- 🌾 Project Demonstration





INTRODUCTION



Introduction: Project Background



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)



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)

“Rice prices surged to their highest in almost 12 years, after India’s rice export ban and adverse weather conditions dented production and supplies of Asia’s primary staple food, according to the UN’s food agency.”(Tan & Shan, 2023)

Introduction: Objective

Utilize Machine Learning Models for Water-Intensive Crop Disease Classification



Current Research:

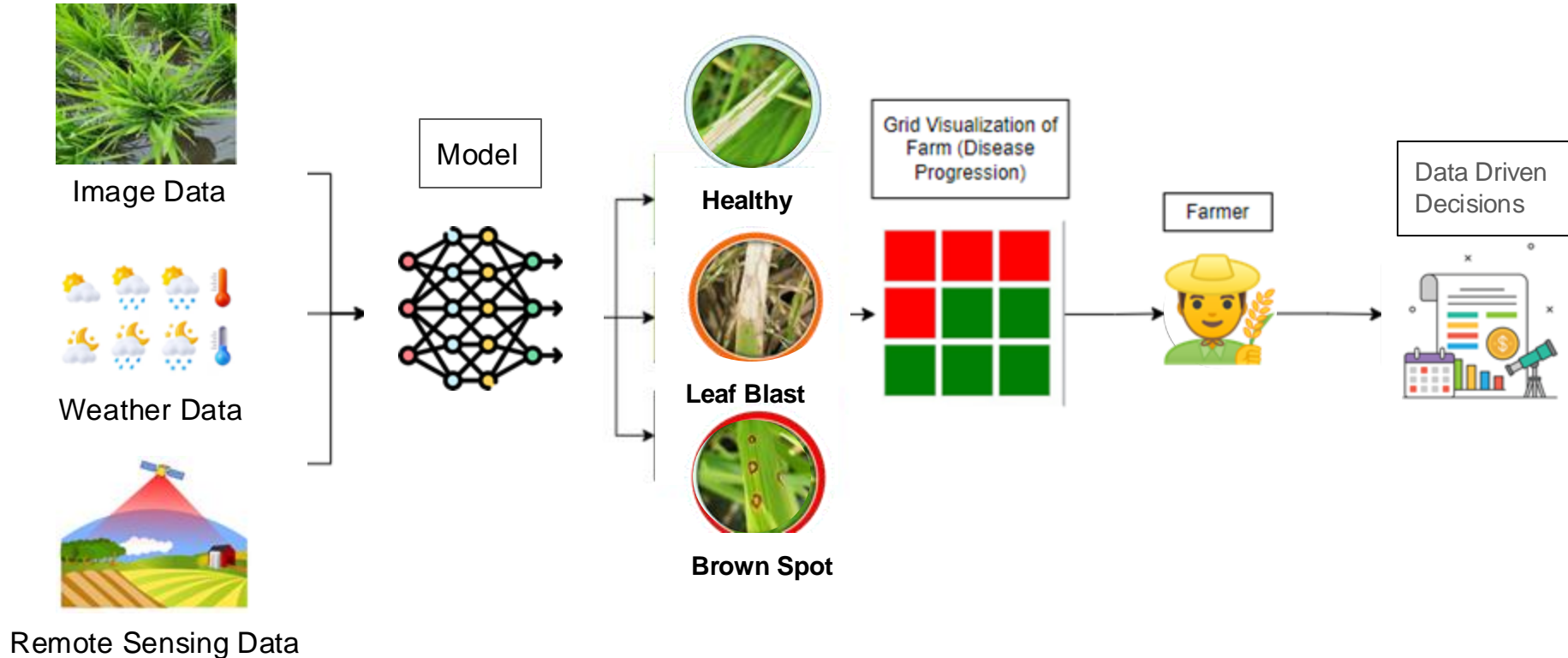
- Pure Image Classification Models
- Remote Sensing (Hyperspectral Readings) Classification Models



Our Approach:

- Disease Classification Model
- Combine: Image Data + Remote Sensing Data + Weather Data
 - Use Satellite for remote sensing data
- Compare with current approaches for effectiveness

Introduction: Problem Goal and Application



Tasks



Gather Data and Preprocess Data for Training

- Gather images for training
- Gather weather data and remote sensing data for each labeled image



Create a Disease Classification Model

- Using image classification models (Image Only)
- Using image + weather + remote sensing data for different models (Hybrid)
- Compare, Optimize and Fine tune selected model



Create a Interactive Web Portal

- Visualize disease classifications into grids.
- Allows users such as farmers to visualize the possible disease outbreaks on their farm
- Allows farmers to perform data driven decisions for the farm



DATA ENGINEERING



Data Engineering: Data Collection

Collected data must have combination of:

Labeled Image Dataset

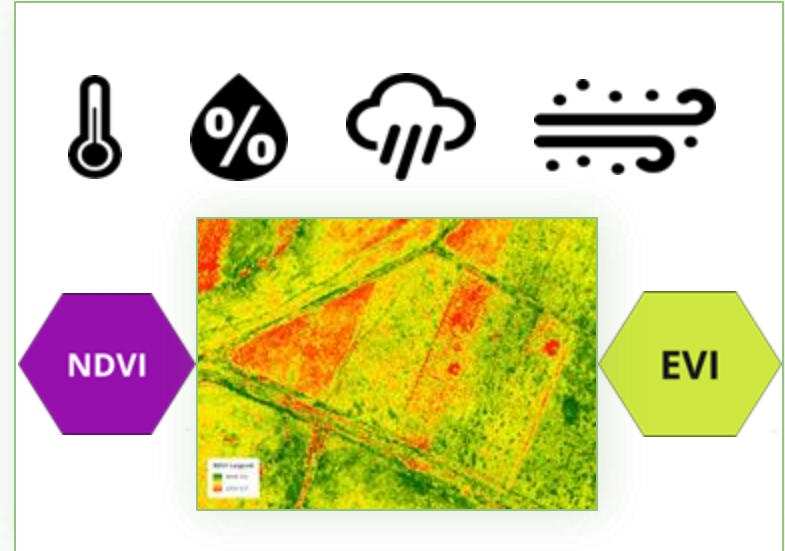
- Rice Leaf Diseases

Weather Data

- Temperature
- Humidity
- Precipitation
- Wind Speed

Remote Sensing Data

- NDVI
- EVI



Data Engineering: Data Collection

Kaggle Dataset: Rice leaf diseases in Taiwan (927 Images)

✎ Contains labeled rice leaf diseases images:

- Brown Spot
- Leaf Blast
- Healthy

✎ Majority of the Image Metadata contains:

- Location (Geotagged)
- Time

✎ Input Time and Location into API:

- Extract Weather data using Visual Crossing API
- Extract Remote Sensing data using Google Earth Engine API

Brown Spot



Leaf blast



Data Engineering: Data Collection



Healthy



Brown Spot



Leaf Blast

Data Engineering: Data Collection

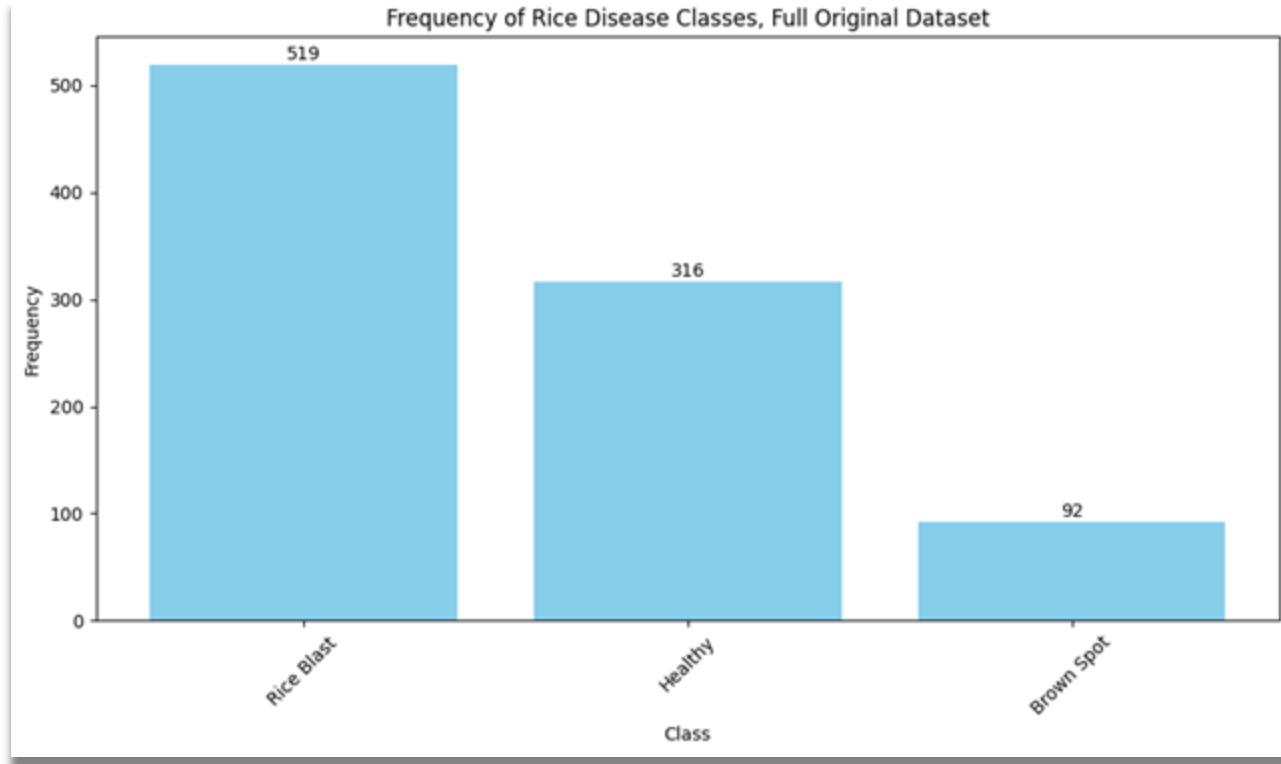
Collected and Combined Data:

- 👉 Image
- 👉 Tabular

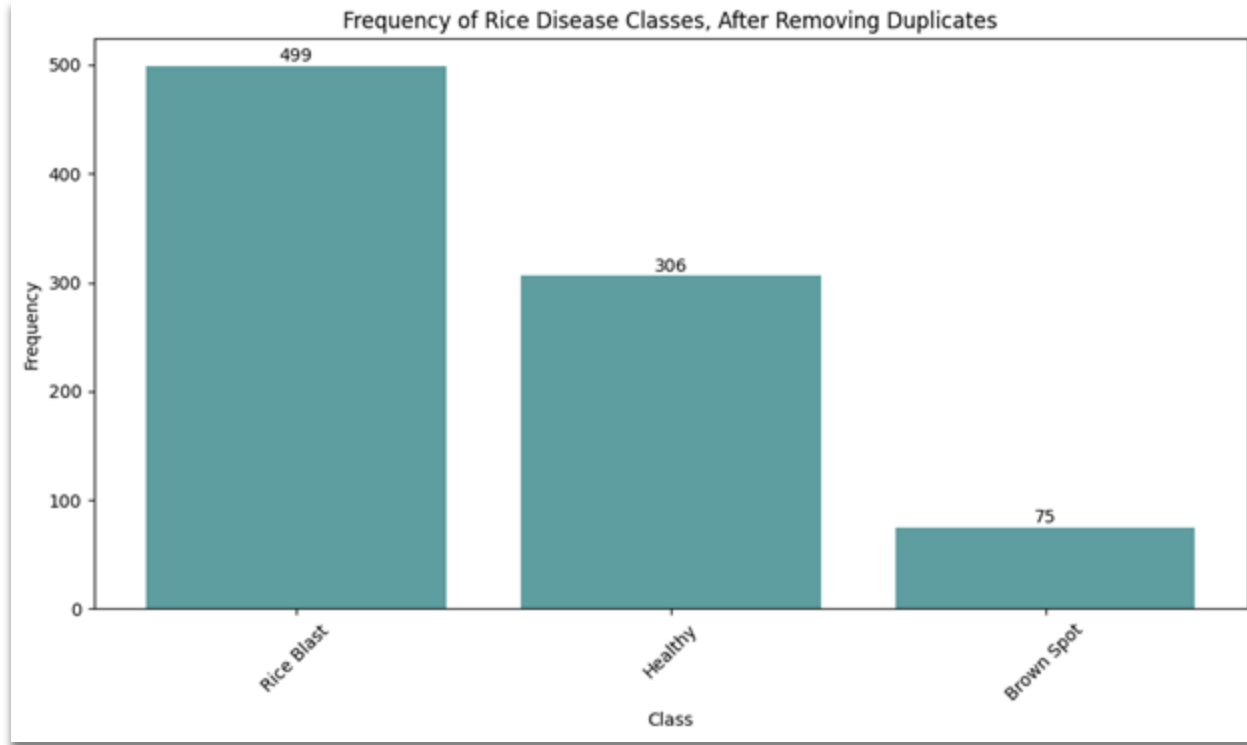


	Id	Latitude	Longitude	Date	Class	Date and Time	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS	NDVI - 1 MODIS	NDVI - 2 MODIS	EVI MODIS	EVI - 1 MODIS	EVI - 2 MODIS
0	P_20181227_153331_vHdR_Auto.jpg	24.073258	120.661451	2018-12-27	Brown Spot	2018:12:27 15:33:31	19.328571	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

Data Engineering: Data Preprocessing

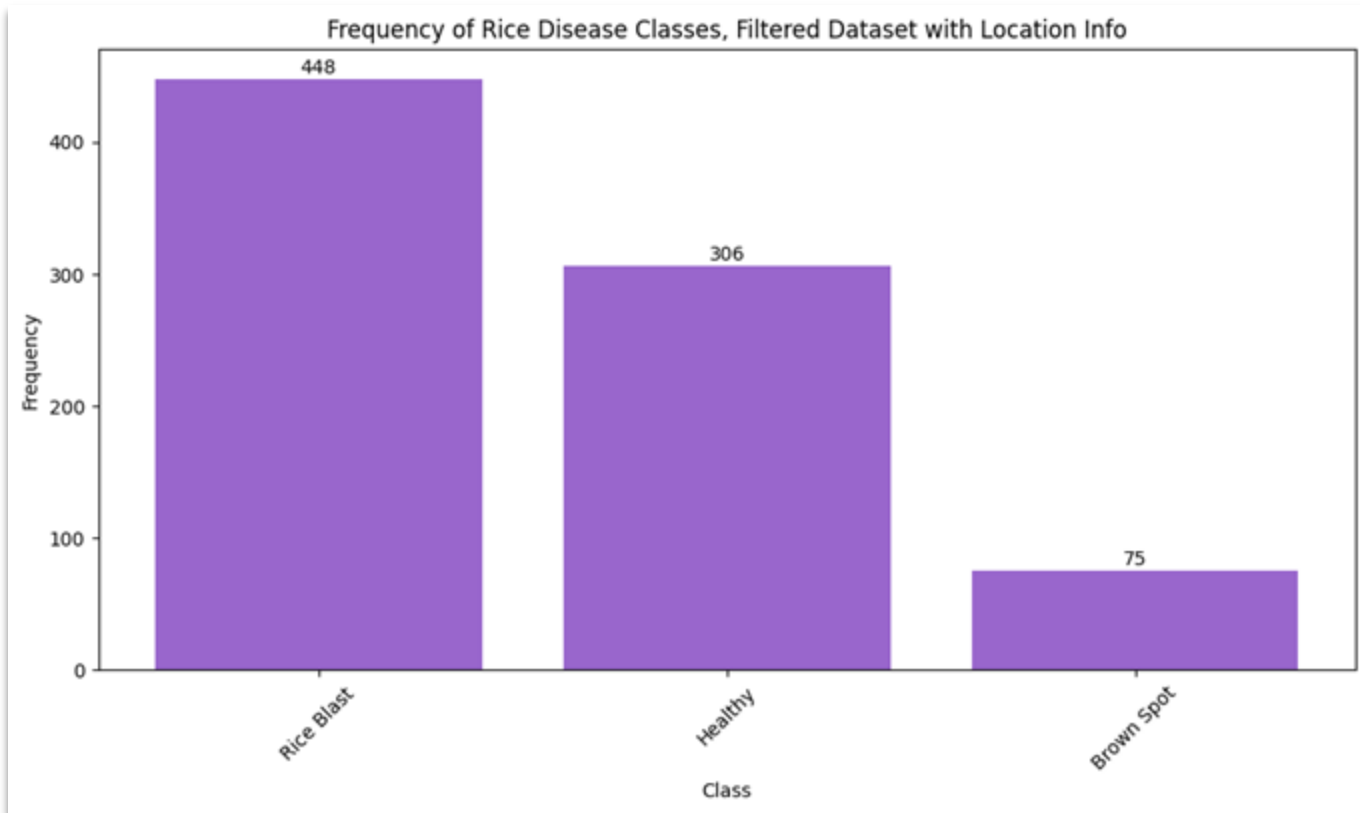


Data Engineering: Data Preprocessing



Duplicate images were detected and removed using the image's hash

Data Engineering: Data Preprocessing



Looked into each image's metadata and removed images without a Latitude, Longitude, and Date, which are needed later for modeling

Data Engineering: Data Preparation

👉 70/15/15 train/validation/test split

- The distribution of the class/target variable was maintained throughout the splitting of the data.

👉 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
- Prevents further training of the model when the performance on the validation set begins to degrade.

```
train_df['Class'].value_counts()
```

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

```
val_df['Class'].value_counts()
```

```
Rice Blast    67  
Healthy       46  
Brown Spot    11  
Name: Class, dtype: int64
```

```
test_df['Class'].value_counts()
```

```
Rice Blast    68  
Healthy       46  
Brown Spot    11  
Name: Class, dtype: int64
```

Data Engineering: Data Preparation



To counter the imbalance of the target class, and to increase the size of the data, employed data augmentation on images in the train set only.

- 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.

Data Engineering: Data Preparation




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

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

```
# 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
```

Data Engineering: Data Preparation

-  - Specific augmentations: horizontal & vertical rotations, brightness changes
-  Rotations: employed to make the model more robust to changes in rotation and to generalize better to a wider array of images.
-  - Brightness changes: images were randomly brightened or darkened with a maximum percentage change of 30.
 - Simulate various lighting conditions that may occur naturally due to environment changes, for instance due to the time of day.
 - Helps the model generalize better to images with a variety of brightness levels.

Original Image



Augmented Image (Left-Right Flip)



Original Image



Augmented Image (Up-Down Flip)



Original Image



Augmented Image (Brightness)



Data Engineering: Data Preparation - Image Data

- Some of the images were very high resolution (e.g. 4000x3000), far exceeding input sizes required for pre-trained models (e.g. 224x224 for VGG19).
 - Directly resizing these images can result in significant loss of detail and distortion of the aspect ratio.
- 'tf.image.resize_with_pad' - resizes the images while maintaining the original aspect ratio without distortion, preserving the fidelity of the image's details.
- Antialiasing smooths images to prevent jaggedness that occasionally occurs when scaled down.
 - Preserving image quality is crucial when training convolutional neural network models.
- After resizing, used a 'preprocess_input' function of the pre-trained module in TensorFlow
 - Standardize the pixel values to a format the pre-trained model expects.
 - Ex: for VGG-19, "tensorflow.keras.applications.vgg19.preprocess_input"

Data Engineering: Data Preparation - Tabular Data

👉 Standardized the weather data using StandardScaler()

- $(X - \mu) / \sigma$
- The remote sensing data were previously scaled during initial data collection
- All numerical data are on the same scale, and thus no bias will be introduced to the natural magnitude of the data

👉 Adding binary indicators to the remote sensing data that shows whether or not there was a decrease in these indices from the previous 2 periods, which might be indicative of disease

	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

Data Engineering: Data Preparation - TensorFlow Modeling



TensorFlow's 'Dataset' API was used to transform the datasets into a format suitable for training in TensorFlow

- 'from_tensor_slices' function with the image data, numerical data, and class labels to convert them into datasets compatible for use in TensorFlow
- Used Dataset.zip to combine the image and numerical datasets, ensuring that future models receive two inputs



Training, validation, and test sets were batched and prefetched to increase computational efficiency and the convergence speed of the model.

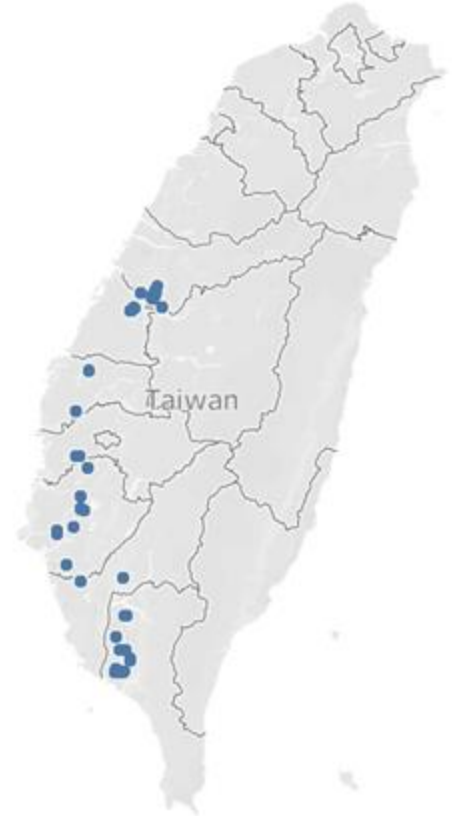
Data Engineering: Data Analytics Result

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	

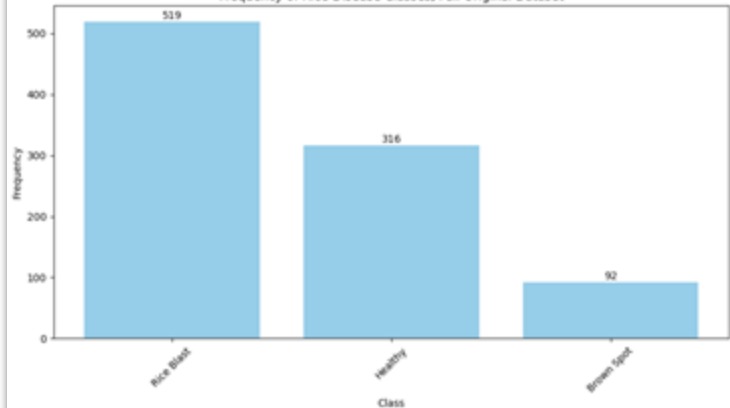
combined_data.describe()

	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	21.879152	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

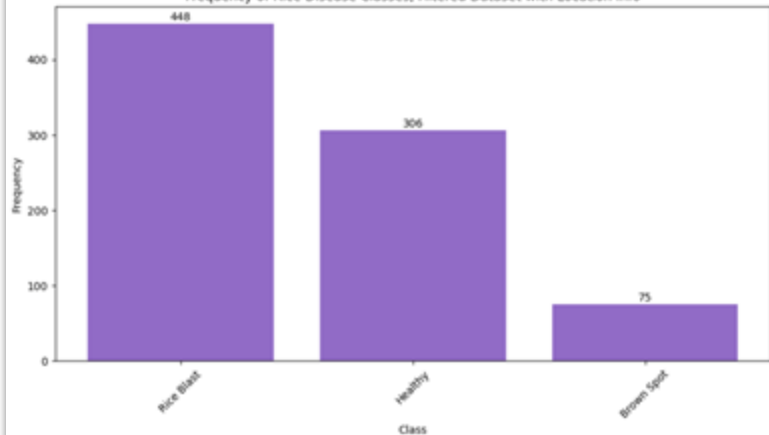


Data Engineering: Data Analytics Result

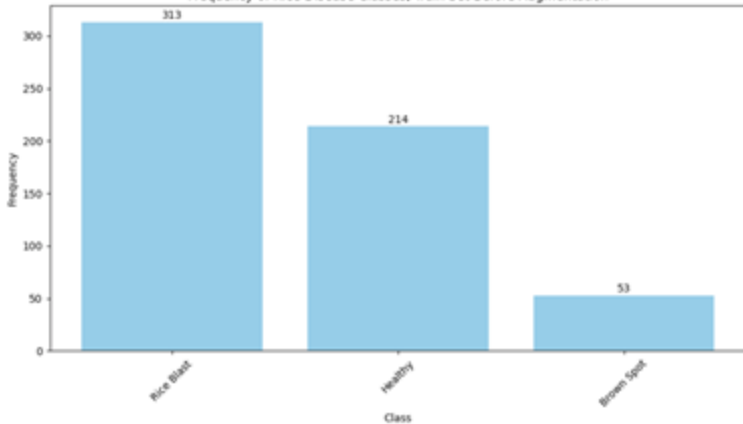
Frequency of Rice Disease Classes, Full Original Dataset



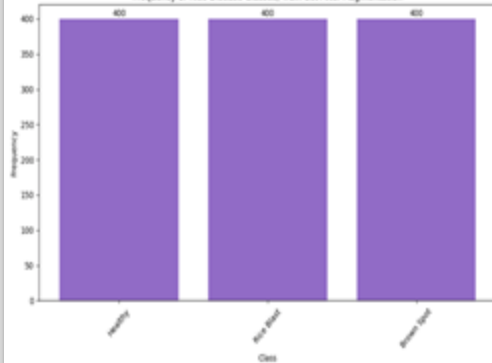
Frequency of Rice Disease Classes, Filtered Dataset with Location Info



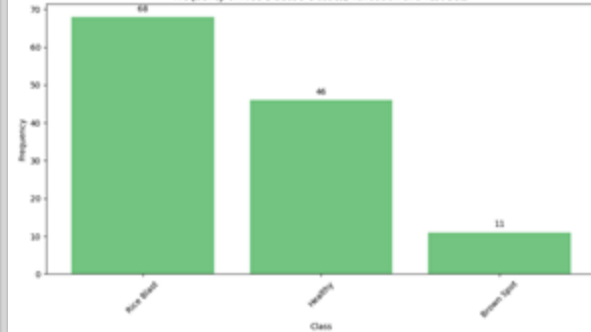
Frequency of Rice Disease Classes, Train Set Before Augmentation



Frequency of Rice Disease Classes, Train Set After Augmentation



Frequency of Rice Disease Classes, Validation and Test Sets





Machine Learning Models



Proposed Machine Learning Models



- **VGG-19**



- **ResNet-50**



- **InceptionV3**



- **DenseNet121**

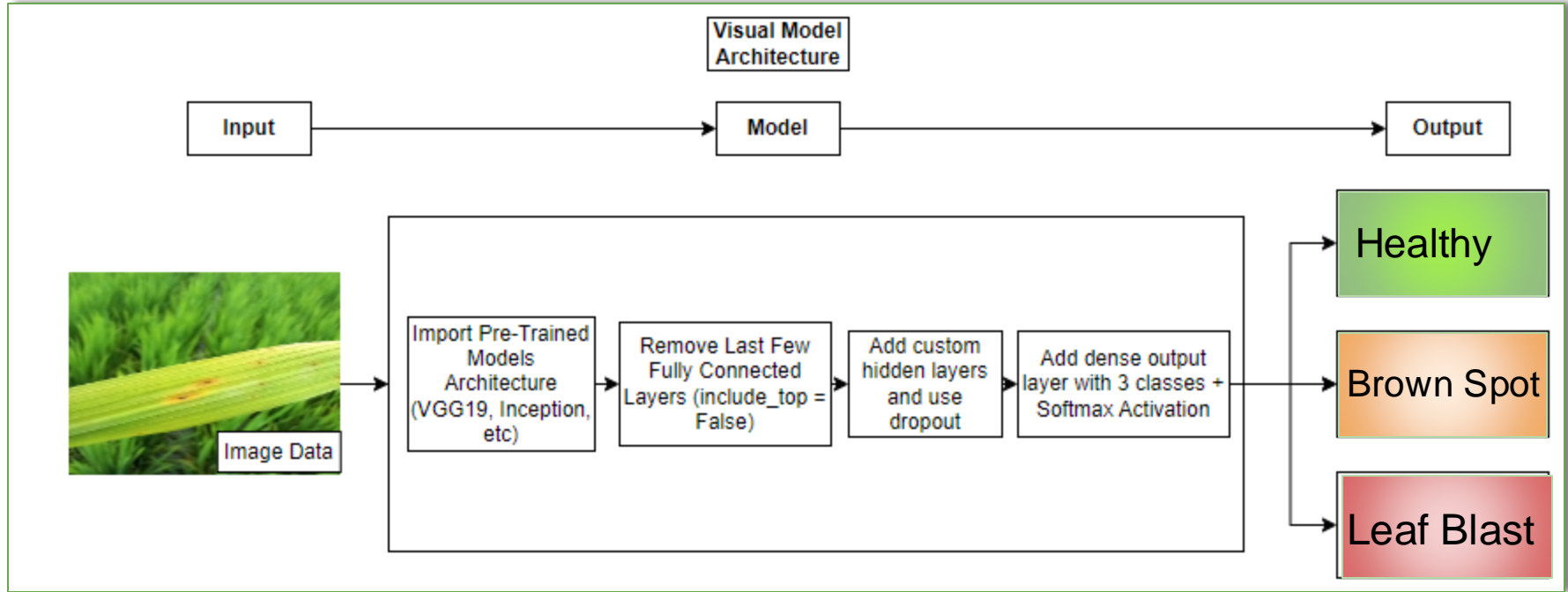


- **Hybrid Models**



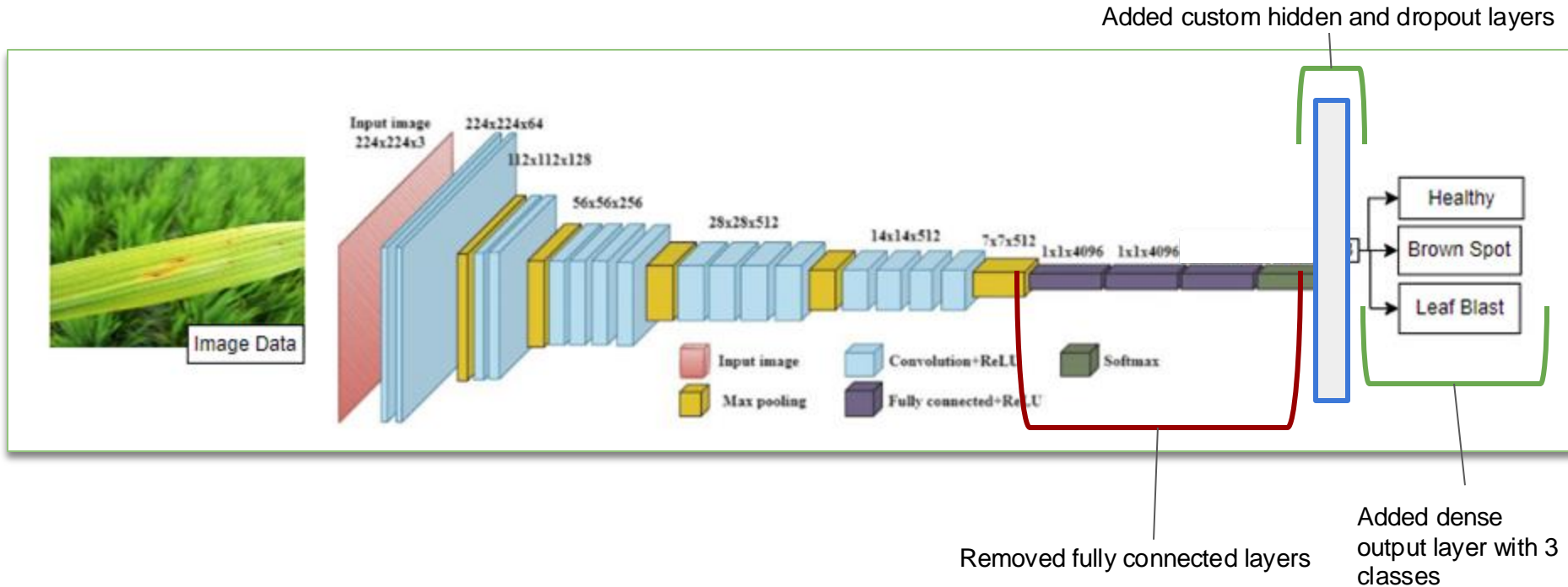
- **Ensemble Models**

Visual-Only Model Architecture



VGG-19 Architecture

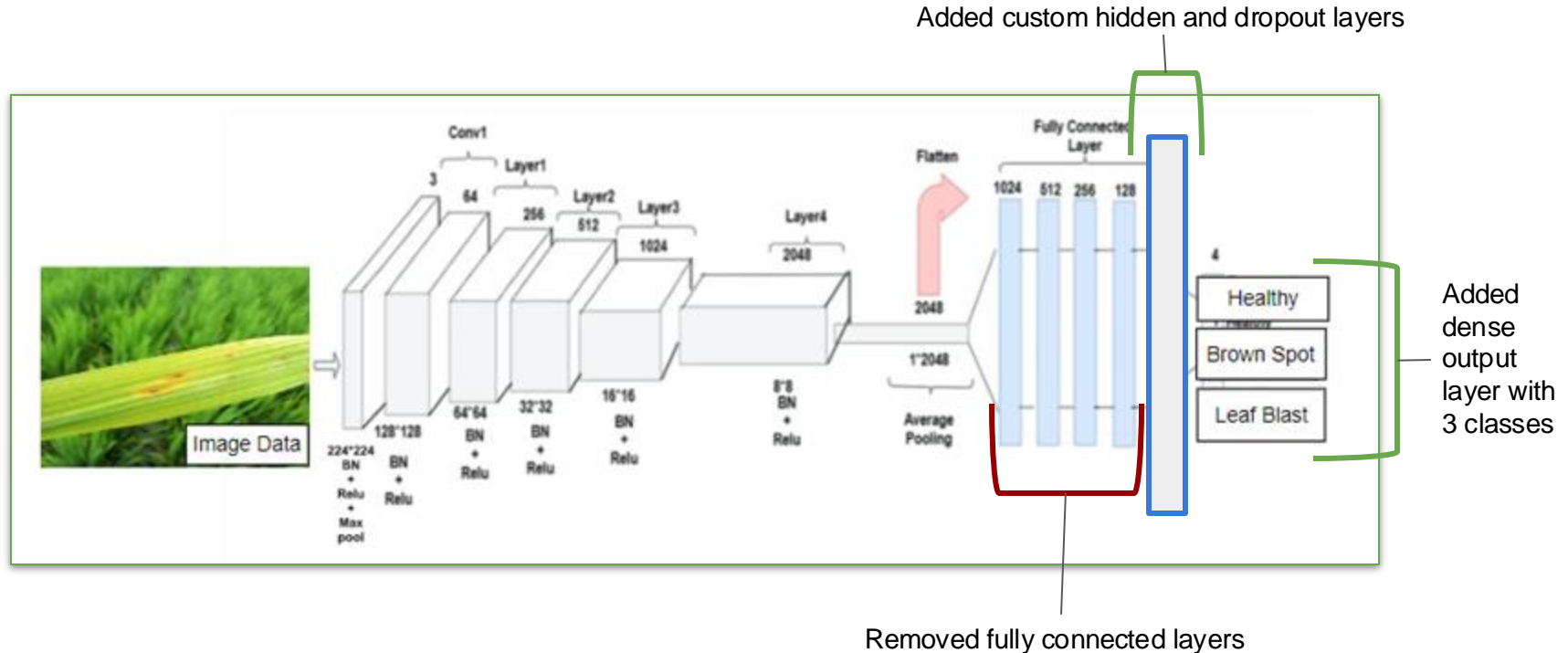
- Deep architecture, 19 layers
- Uniform 3x3 convolutions
- High computational cost



Note. Image credit: Adapted from
<https://doi.org/10.3390/agriengineering4040056>

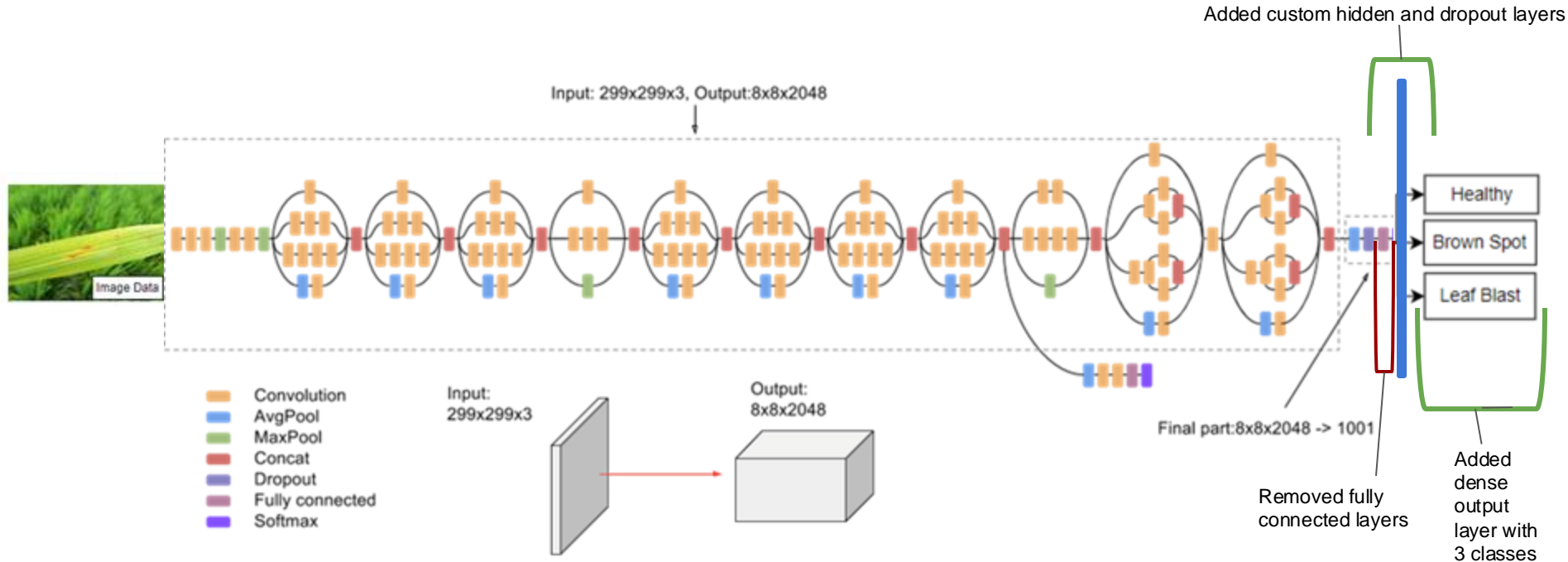
ResNet50 Architecture

- Residual learning, skip connections
- Solves vanishing gradient
- Efficient, deep (50 layers)



InceptionV3 Architecture

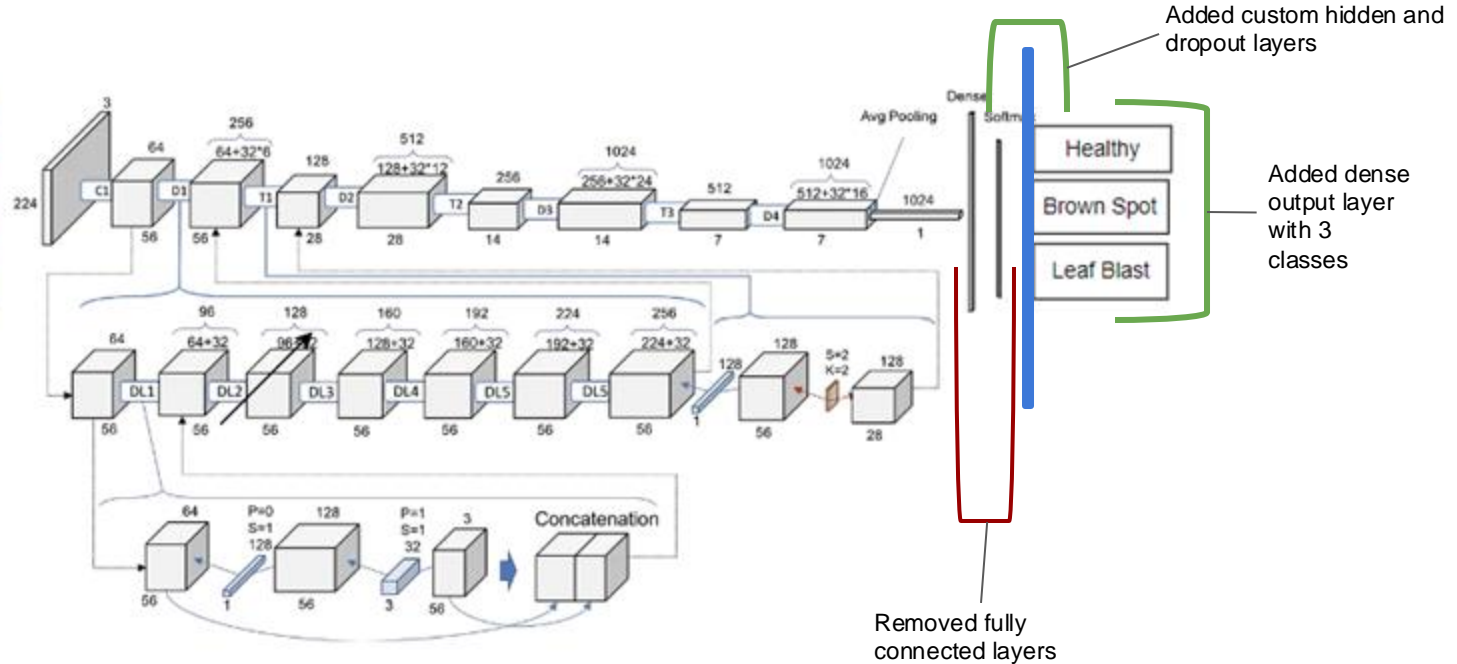
- Factorized convolutions, Inception modules
- Auxiliary classifiers, batch normalization
- Modular, scalable



Note. Image credit: Adapted from
<https://doi.org/10.14569/IJACSA.2019.0100107>

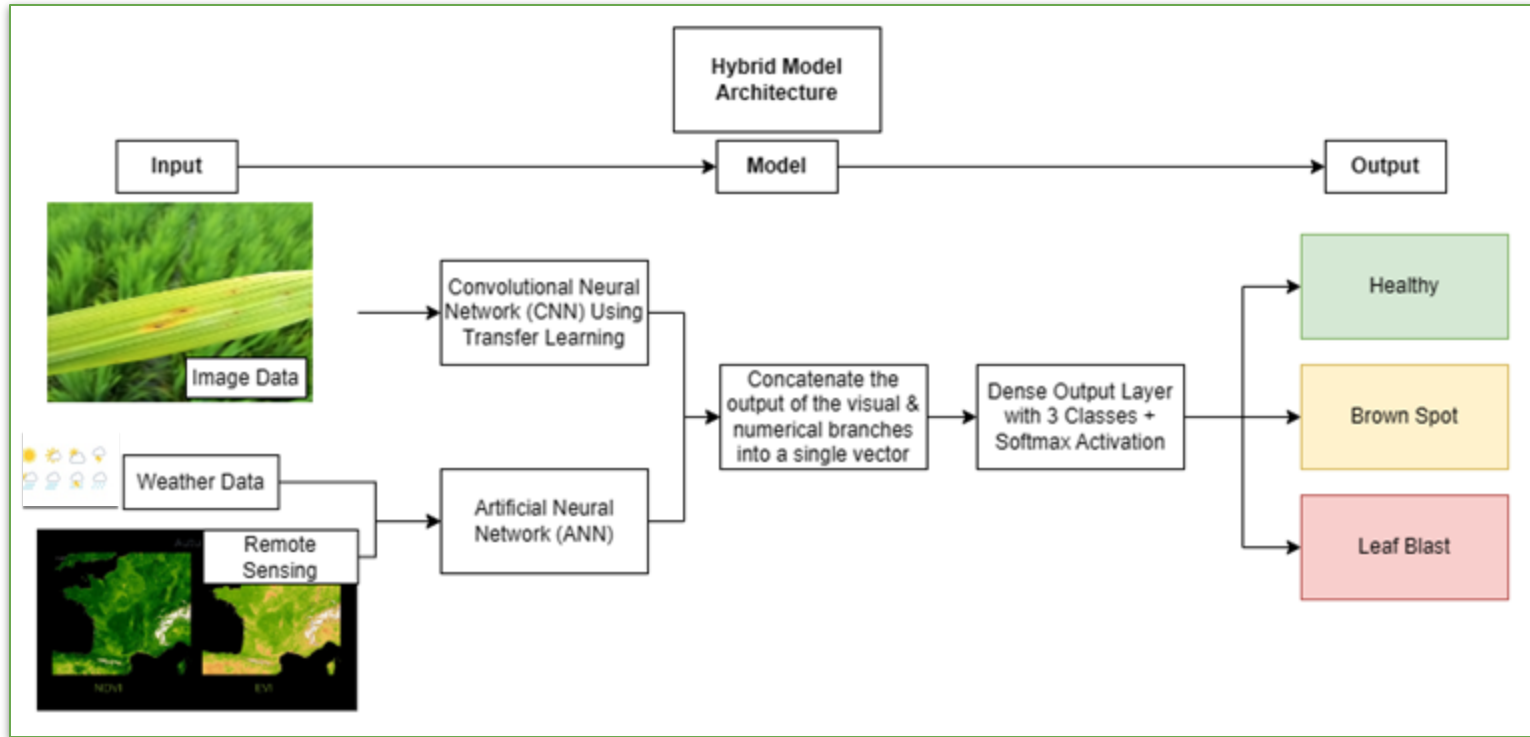
DenseNet121 Architecture

- Dense connectivity, layer-to-layer
- Parameter efficient, reduces overfitting
- Compact, effective for recognition tasks



Note. Image credit: Adapted from
<https://towardsdatascience.com/understanding-and-visualizing->

Hybrid Model Architecture



Ensemble Models

- ✍ - Ensembles of both the individual hybrid models and visual-only models will be created and evaluated
 - Aggregate predictions from each model to hopefully achieve a better accuracy
- ✍ - Majority Voting: each individual model makes a class prediction/vote for each instance. The class with the most votes is the final output prediction.
- ✍ - Soft Voting: the average probability of each class label using each classifier is obtained, and the class with the highest probability is the final prediction

Model Evaluation Results



Further tuning to achieve better model accuracies for both the visual-only and hybrid models

- A greater combination of dropout rates and neurons in the hidden layers



Modified the total number of frozen layers to further fine-tune the base transfer learning models.

- Unfrozen layers at the end to better adapt to the features of our specific dataset
 - In the hybrid model, the visual branch includes a pre-trained base model with the last few layers unfrozen.



The number of frozen layers involves careful consideration - more trainable layers increases computational demands and can heighten the risk of overfitting.

- Overly adapted to the training data, compromises its ability to generalize to new instances.

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%

Model Evaluation Results, cont.

- 👉 More extensive experimentation with the modeling hyperparameters and fine-tuning improved accuracy for all models on the test set, except the hybrid InceptionV3.
 - Classification task - metrics are obtained by comparing the predicted & true labels
- 👉 Hybrid models outperform their visual-only counterparts, with the hybrid ResNet50 & DenseNet121 models achieving the highest accuracies.
- 👉 The models are saved in .h5 format. The **hybrid DenseNet121** model, smaller than ResNet50, is chosen as the final model.
 - Have a more lightweight model for use in the application.
 - Macro-average ROC-AUC score, OvR scheme: 0.9994, Macro-average F1-score: 0.99
 - Further underscores the exceptional predictive ability of the model across all classes.

Sample Testing Results

Raw Image + Numerical Data



Id	Latitude	Longitude	Date and Time	Class	Date	Avg Temp 14d	Avg Humidity 14d	Total Precipitation 14d	Avg Wind Speed 14d	NDVI MODIS
IMG_20190330_120129.jpg	22.50264222222220	120.52227713888900	2019-03:30 12:01:29	Rice Blast	2019-03-30	24.385714285714300	75.05	7.297	17.807142857142900	0.6890000000000000
IMG_20190308_114943.jpg	24.098481027777800	120.66923794444400	2019-03:08 11:49:43	Healthy	2019-03-08	18.457142857142900	80.80714285714290	75.8	26.428571428571400	0.351
P_20181227_162244_vHDR_Auto_HP.jpg	24.069349222222200	120.65324747222200	2018-12:27 16:22:44	Brown Spot	2018-12-27	19.314285714285700	76.67142857142860	6.7	29.171428571428600	0.404

Model Predictions

Id	Latitude	Longitude	Date	Class Confidence Levels	Class Prediction
IMG_20190330_120129.jpg	22.50264222222220	120.52227713888900	2019-03-30	{'Rice Blast': 1.0, 'Brown Spot': 0.0, 'Healthy': 0.0}	Rice Blast
IMG_20190308_114943.jpg	24.098481027777800	120.66923794444400	2019-03-08	{'Healthy': 0.966, 'Rice Blast': 0.0339, 'Brown Spot': 0.0001}	Healthy
P_20181227_162244_vHDR_Auto_HP.jpg	24.069349222222200	120.65324747222200	2018-12-27	{'Brown Spot': 0.9988, 'Rice Blast': 0.0012, 'Healthy': 0.0}	Brown Spot

Predicted: Healthy (96.60%), True: Healthy



Predicted: Rice Blast (100.00%), True: Rice Blast

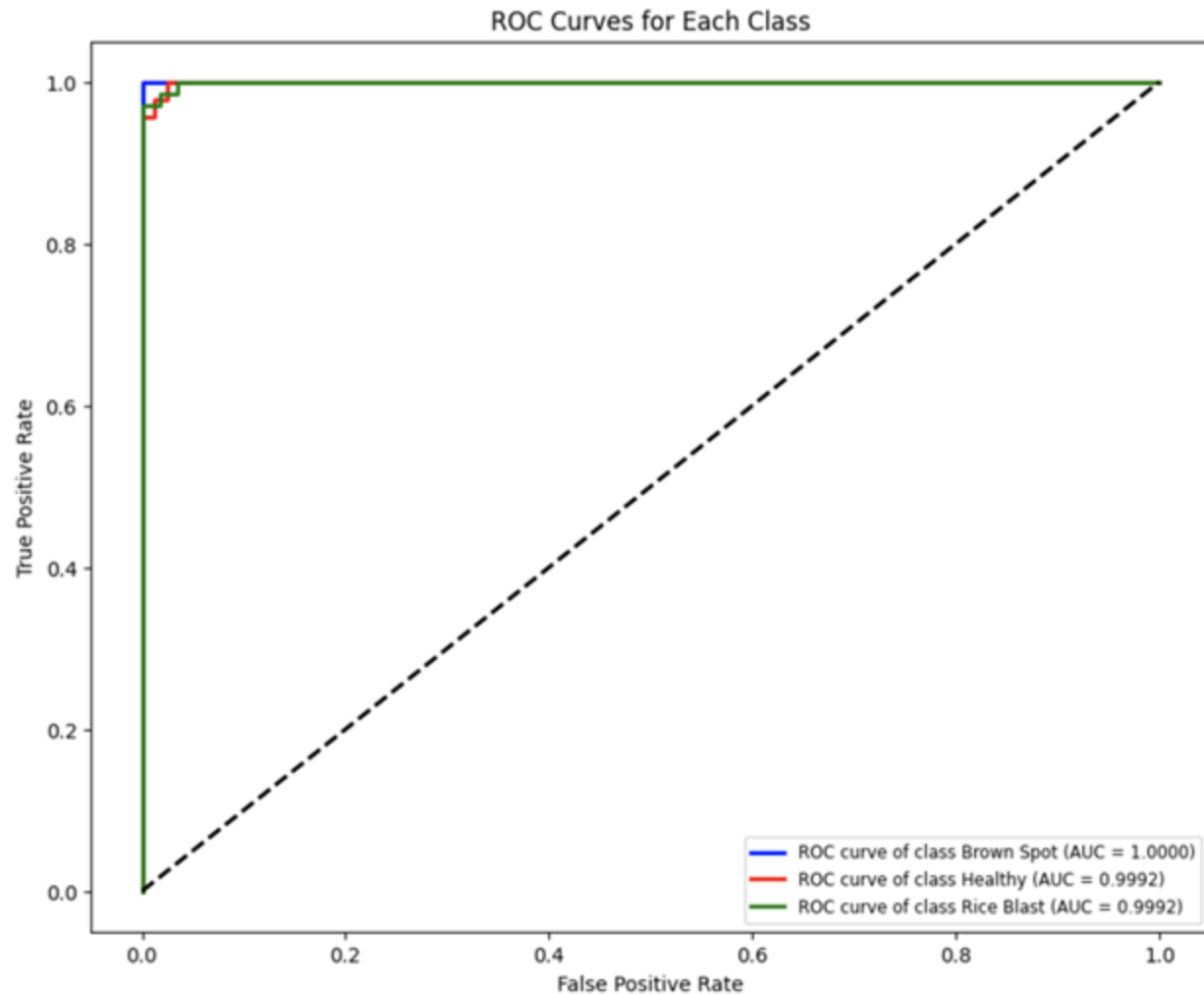


Predicted: Brown Spot (99.88%), True: Brown Spot



ROC Curves, OvR, Final Hybrid DenseNet121 Model

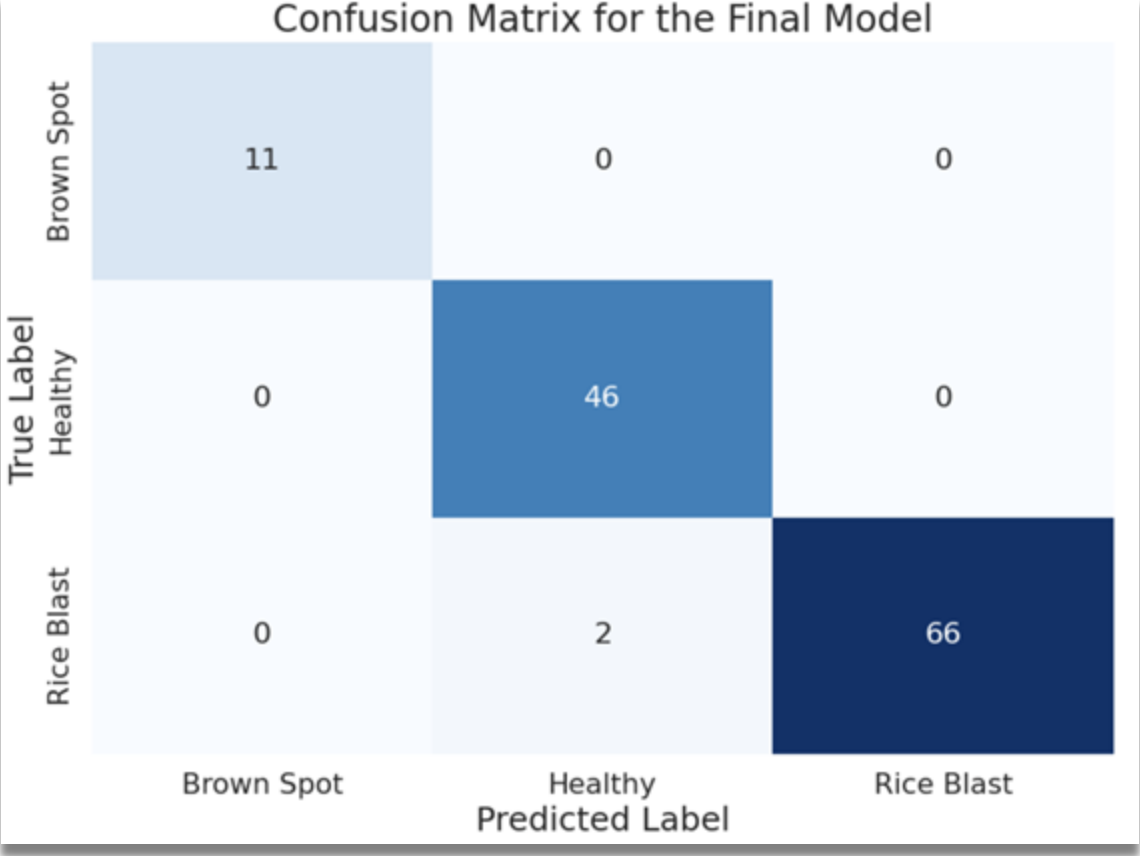
Macro- Average ROC-AUC
Score: 0.9994



Classification Report, Final Hybrid DenseNet121 Model

	precision	recall	f1-score	support
Brown Spot	1.00	1.00	1.00	11
Healthy	0.96	1.00	0.98	46
Rice Blast	1.00	0.97	0.99	68
accuracy			0.98	125
macro avg	0.99	0.99	0.99	125
weighted avg	0.98	0.98	0.98	125

Confusion Matrix, Final Hybrid DenseNet121 Model



Run-Time Evaluation

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

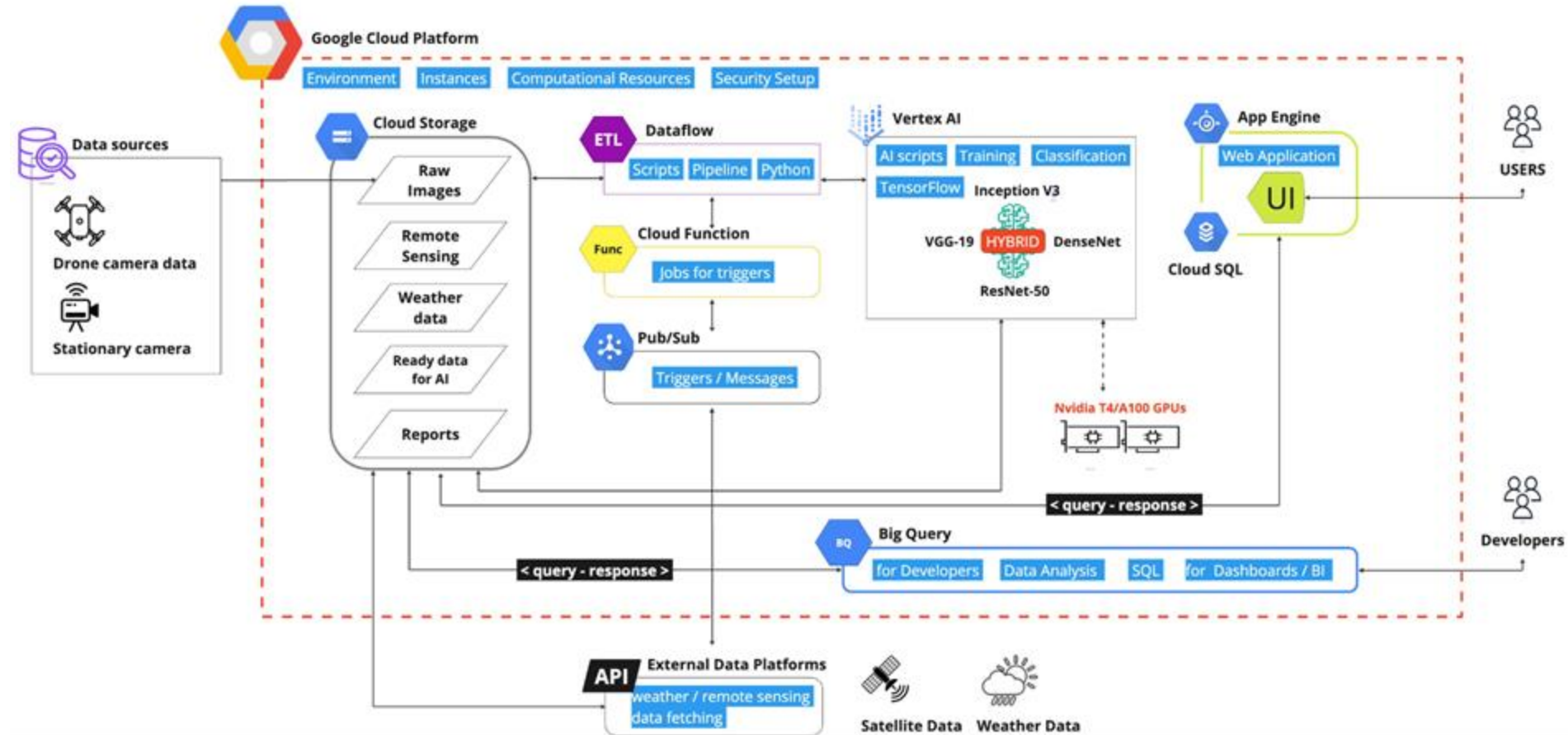
Feature	Device Specs	Average Run Time (seconds)
Extract Metadata from Images	MacBook Pro Processor: 2.3 GHz 8-Core Intel Core i9 Memory: 16GB 2667 MHz DDR4	0.1085 seconds
Obtaining Remote Sensing Data		34.4002 seconds
Obtaining Weather Data		1.4923 seconds
Combine Image Metadata, Remote Sensing and Weather 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



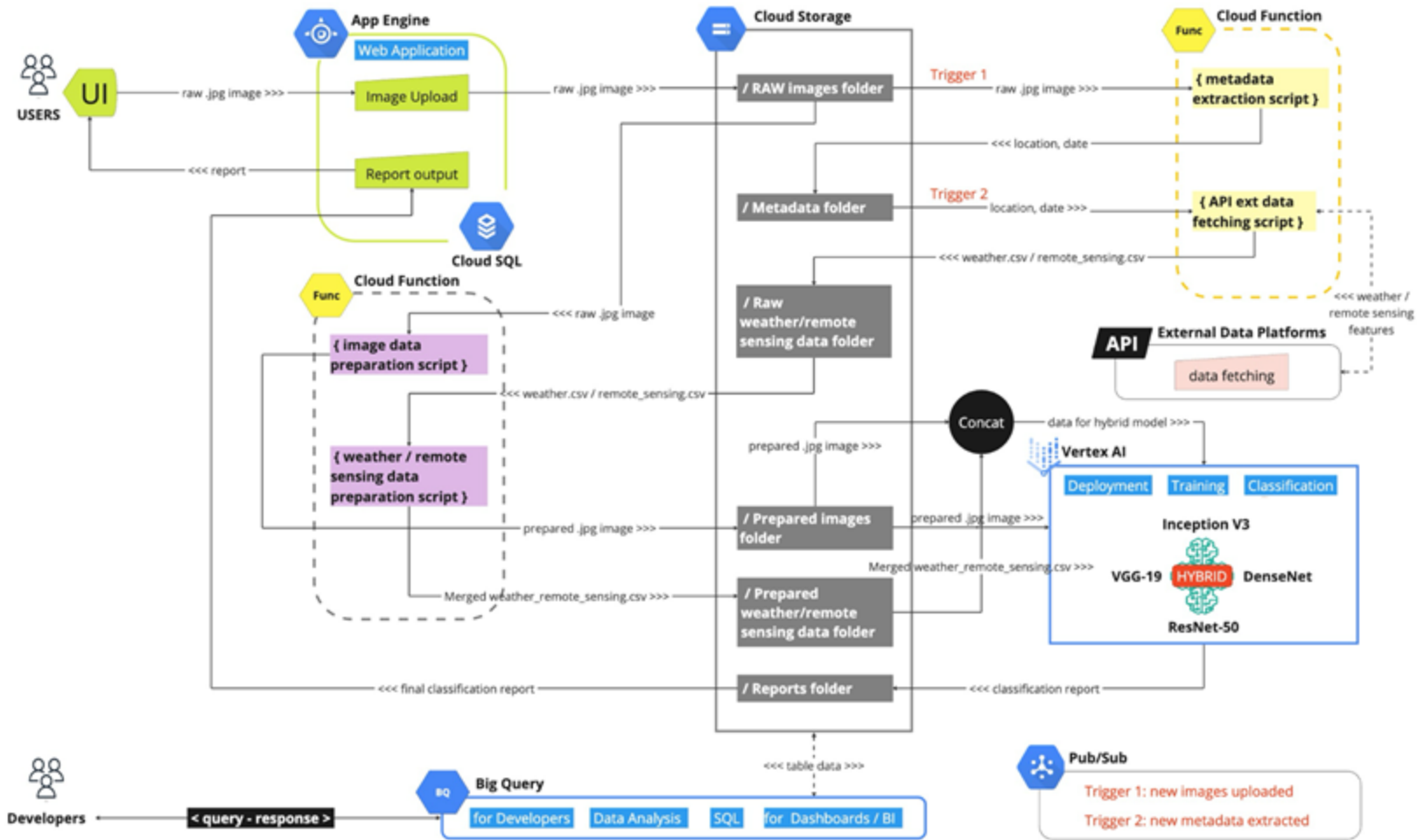
System Requirements and Design



System Architecture



System Data Flow





Web System Design and Development





Google IAM and Secret Manager

IAM & Admin

IAM

PAM **NEW**

Identity & Organization

Policy Troubleshooter

Policy Analyzer

Organization Policies

Service Accounts

Workload Identity Federat...

Workforce Identity Federa...

Labels

Tags

Settings

Service accounts [+ CREATE SERVICE ACCOUNT](#)

Service accounts for project "Crop Disease Classification AI"

A service account represents a Google Cloud service identity, such as code running on Compute En

Organization policies can be used to secure service accounts and block risky service account featu

Filter Enter property name or value

<input type="checkbox"/>	Email	Status	Name ↑
<input type="checkbox"/>	tidy-nomad-415320@appspot.gserviceaccount.com	✓ Enabled	App Engine default service account
<input type="checkbox"/>	cloud-sql@tidy-nomad-415320.iam.gserviceaccount.com	✓ Enabled	Cloud SQL
<input type="checkbox"/>	57814283461-compute@developer.gserviceaccount.com	✓ Enabled	Compute Engine default service account
<input type="checkbox"/>	vertex-ai@tidy-nomad-415320.iam.gserviceaccount.com	✓ Enabled	Vertex AI

Security

Secret Manager

Security Command Center ^

Risk Overview

Threats

Vulnerabilities

Compliance

Assets

Findings

Sources

Posture Management

Detections and Controls ^

SECRETS LOGS

Secret Manager lets you store, manage, and retrieve secrets in a secure and scalable way. [Learn more](#)

Filter Enter property name or value

<input type="checkbox"/>	Name ↑
<input type="checkbox"/>	DB_CONN_NAME
<input type="checkbox"/>	DB_HOST
<input type="checkbox"/>	DB_NAME
<input type="checkbox"/>	DB_PASS
<input type="checkbox"/>	DB_USER
<input type="checkbox"/>	flask_secret_key
<input type="checkbox"/>	main-service-acc-key

Database tables

User Data

Image Field Batch User ×					
⏪ < 4 rows > ⏩ ↺ ⌚ ■ + − ↶ ↷ ⬆ Tx: Auto ▾ DDL 🔍 📄					
🔍 WHERE			≡ ORDER BY		
	🔗 id	🔗 name	🔗 email	🔗 password	🔗 datetime
1	4	Nail	nail@cropai.com	pbkdf2:sha256:600000\$Vyo10UKgEkPpsXx7\$53fcdfc819a3775540a4e91e8...	2024-03-26 01:36:42.129171
2	5	Jason	jason@crop.ai	pbkdf2:sha256:600000\$yprplrGxLdEW04QH\$28e56a89e9a9848181f9f208f...	2024-03-26 02:44:56.508050
3	6	Nassim	nassim@cropai.com	pbkdf2:sha256:600000\$nRJFjE6uK11yfNnF\$b539207b7ef1c5f36fcd9a168...	2024-03-26 02:45:11.404888
4	7	Vrushali	vrushali@cropai.com	pbkdf2:sha256:600000\$eEsbIRfsGcsfvTfz\$cbba9cdc643cc769a582a6611...	2024-03-26 02:45:32.394781

Field Data

Image Field × Batch User				
⏪ < 4 rows > ⏩ ↺ ⌚ ■ + − ↶ ↷ ⬆ Tx: Auto ▾ DDL 🔍 📄				
🔍 WHERE		≡ ORDER BY		
	🔗 id	🔗 user_id	🔗 name	🔗 datetime
1	84	4	002 Taiwan	2024-04-08 19:35:28.789763
2	85	4	003 Taiwan	2024-04-08 20:12:25.211833
3	86	4	004 Taiwan	2024-04-08 20:12:29.128151
4	87	4	001 Fresno	2024-04-08 20:12:34.596959

Batch Data

Image

Field

Batch

User

<

>

3 rows

<

>

↺

⌚

■

+

−

↶

🌀

⬆

Tx: Auto

DDL

🔍

📄

WHERE

ORDER BY

	🔗 id	÷	🔗 user_id	÷	🔗 field_id	÷	÷	÷	🔗 datetime	÷	🔗 date_taken
1	100		4		84		20	4 5	2024-04-08 19:48:41.240163		2024-02-01
2	101		4		84		20	4 5	2024-04-08 19:56:14.679689		2024-03-01
3	102		4		84		20	4 5	2024-04-08 20:00:40.866585		2024-04-01

Image Data

Image

Field

Batch

User

<<60 rows>>

↺⌚■

+

−

↶

↷

↕

Tx: Auto

DDL

🔍

📄

WHERE

ORDER BY

	id	batch_id	filename	label	path	datetime	healthy	rice_blast	brown_spot	order	date_taken
1	895	100	001_4_5_1_202...	Healthy	userdata/84/10...	2024-04-08 19...	0.9781	0.0206	0.0013	1	2024-02-01
2	896	100	001_4_5_2_202...	Healthy	userdata/84/10...	2024-04-08 19...	0.9854	0.0138	0.0007	2	2024-02-01
3	897	100	001_4_5_3_202...	Healthy	userdata/84/10...	2024-04-08 19...	0.9854	0.0138	0.0007	3	2024-02-01
4	898	100	001_4_5_4_202...	Healthy	userdata/84/10...	2024-04-08 19...	0.9854	0.0138	0.0007	4	2024-02-01
5	899	100	001_4_5_5_202...	Healthy	userdata/84/10...	2024-04-08 19...	0.9854	0.0138	0.0007	5	2024-02-01
6	900	100	001_4_5_6_202...	Healthy	userdata/84/10...	2024-04-08 19...	0.9854	0.0138	0.0007	6	2024-02-01



Google Cloud SQL

All instances > mainsql

PRIM...

Explorer

Databases 1

database (Default)

Tables 4

Batch

Columns 8

Indexes 3

Keys 3

Triggers 0

Field

Columns 5

Indexes 2

Keys 2

Triggers 0

Image

Columns 11

Indexes 2

Keys 2

Triggers 0

User

Columns 5

id

Tr name

Tr email

Tr password

RUN

FORMAT

CLEAR

1 SELECT * FROM

2 | 'database'. 'Image' LIMIT 1000;

3

RESULTS

id	batch_id	filename	label	path	datetime	healthy	rice_blast	brown_spot	order	date_taken
1	1	001_4_5_1_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_1_2024-03-10.JPG	2024-05-09 22:44:01	0.9781	0.0206	0.0013	1	2024-03-10
2	1	001_4_5_2_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_2_2024-03-10.JPG	2024-05-09 22:45:01	0.9854	0.0138	0.0007	2	2024-03-10
3	1	001_4_5_3_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_3_2024-03-10.JPG	2024-05-09 22:46:01	0.9854	0.0138	0.0007	3	2024-03-10
4	1	001_4_5_4_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_4_2024-03-10.JPG	2024-05-09 22:47:01	0.9854	0.0138	0.0007	4	2024-03-10
5	1	001_4_5_5_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_5_2024-03-10.JPG	2024-05-09 22:48:01	0.9854	0.0138	0.0007	5	2024-03-10
6	1	001_4_5_6_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_6_2024-03-10.JPG	2024-05-09 22:49:01	0.9854	0.0138	0.0007	6	2024-03-10
7	1	001_4_5_7_2024-03-10.jpg	Brown Spot	userdata/1/1/001_4_5_7_2024-03-10.jpg	2024-05-09 22:50:01	0	0.0019	0.998	7	2024-03-10
8	1	001_4_5_8_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_8_2024-03-10.JPG	2024-05-09 22:51:01	0.9851	0.0141	0.0008	8	2024-03-10
9	1	001_4_5_9_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_9_2024-03-10.JPG	2024-05-09 22:52:01	0.9862	0.0131	0.0007	9	2024-03-10
10	1	001_4_5_10_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_10_2024-03-10.JPG	2024-05-09 22:53:01	0.9862	0.0131	0.0007	10	2024-03-10
11	1	001_4_5_11_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_11_2024-03-10.JPG	2024-05-09 22:54:01	0.9799	0.0189	0.0012	11	2024-03-10
12	1	001_4_5_12_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_12_2024-03-10.JPG	2024-05-09 22:55:01	0.9799	0.0189	0.0012	12	2024-03-10
13	1	001_4_5_13_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_13_2024-03-10.JPG	2024-05-09 22:56:01	0.9292	0.063	0.0079	13	2024-03-10
14	1	001_4_5_14_2024-03-10.jpg	Healthy	userdata/1/1/001_4_5_14_2024-03-10.jpg	2024-05-09 22:57:01	0.9128	0.0869	0.0003	14	2024-03-10
15	1	001_4_5_15_2024-03-10.jpg	Healthy	userdata/1/1/001_4_5_15_2024-03-10.jpg	2024-05-09 22:58:01	0.694	0.304	0.002	15	2024-03-10
16	1	001_4_5_16_2024-03-10.JPG	Healthy	userdata/1/1/001_4_5_16_2024-03-10.JPG	2024-05-09 22:59:01	0.966	0.0339	0.0001	16	2024-03-10
17	1	001_4_5_17_2024-03-10.jpg	Healthy	userdata/1/1/001_4_5_17_2024-03-10.jpg	2024-05-09 23:00:01	0.966	0.0339	0.0001	17	2024-03-10

Valid



Google Cloud Storage

Cloud Storage



Bucket details

GO TO PATH

REFRESH

LEARN



Buckets



Monitoring



Settings

Location: us-west1 (Oregon)
Storage class: Standard
Public access: Not public
Protection: Soft Delete

OBJECTS

CONFIGURATION

PERMISSIONS

PROTECTION

LIFECYCLE

OBSERVABILITY

INVENTORY REPORTS

OPERATIONS

Folder browser



userdata-tidy-nomad-415320

log/

model_artifacts/

userdata/

1/

1/

2/

3/

4/

5/

6/

7/

9/

Buckets > userdata-tidy-nomad-415320 > userdata > 9 > 35

UPLOAD FILES

UPLOAD FOLDER

CREATE FOLDER

TRANSFER DATA

MANAGE HOLDS

EDIT RETENTION

DOWNLOAD

DELETE

Filter by name prefix only



Filter objects and folders

Show Live objects only



<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public
<input type="checkbox"/>	001_3_3_1_2024-02-01.JPG	3 MB	image/jpeg	May 10, 2024, 9:00:25 AM	Standard	May 10, 2024, 9:00:25 AM	Not p
<input type="checkbox"/>	001_3_3_2_2024-02-01.JPG	3.6 MB	image/jpeg	May 10, 2024, 9:00:25 AM	Standard	May 10, 2024, 9:00:25 AM	Not p
<input type="checkbox"/>	001_3_3_3_2024-02-01.JPG	3.5 MB	image/jpeg	May 10, 2024, 9:00:25 AM	Standard	May 10, 2024, 9:00:25 AM	Not p
<input type="checkbox"/>	001_3_3_4_2024-02-01.JPG	3.7 MB	image/jpeg	May 10, 2024, 9:00:25 AM	Standard	May 10, 2024, 9:00:25 AM	Not p
<input type="checkbox"/>	001_3_3_5_2024-02-01.JPG	2.4 MB	image/jpeg	May 10, 2024, 9:00:26 AM	Standard	May 10, 2024, 9:00:26 AM	Not p
<input type="checkbox"/>	001_3_3_6_2024-02-01.JPG	2 MB	image/jpeg	May 10, 2024, 9:00:26 AM	Standard	May 10, 2024, 9:00:26 AM	Not p
<input type="checkbox"/>	001_3_3_7_2024-02-01.jpg	3.7 MB	image/jpeg	May 10, 2024, 9:00:26 AM	Standard	May 10, 2024, 9:00:26 AM	Not p
<input type="checkbox"/>	001_3_3_8_2024-02-01.jpg	3.4 MB	image/jpeg	May 10, 2024, 9:00:27 AM	Standard	May 10, 2024, 9:00:27 AM	Not p
<input type="checkbox"/>	001_3_3_9_2024-02-01.JPG	2.3 MB	image/jpeg	May 10, 2024, 9:00:27 AM	Standard	May 10, 2024, 9:00:27 AM	Not p
<input type="checkbox"/>	combined_data.csv	2 KB	text/csv	May 10, 2024, 9:01:04 AM	Standard	May 10, 2024, 9:01:04 AM	Not p
<input type="checkbox"/>	image_metadata.csv	879 B	text/csv	May 10, 2024, 9:00:33 AM	Standard	May 10, 2024, 9:00:33 AM	Not p
<input type="checkbox"/>	placeholder	0 B	text/plain	May 10, 2024, 9:00:24 AM	Standard	May 10, 2024, 9:00:24 AM	Not p
<input type="checkbox"/>	predictions_with_confidences.csv	1.8 KB	text/csv	May 10, 2024, 9:01:18 AM	Standard	May 10, 2024, 9:01:18 AM	Not p
<input type="checkbox"/>	predictions_with_confidences.json	2.4 KB	application/json	May 10, 2024, 9:01:18 AM	Standard	May 10, 2024, 9:01:18 AM	Not p
<input type="checkbox"/>	remote_sensing_data.csv	758 B	text/csv	May 10, 2024, 9:00:54 AM	Standard	May 10, 2024, 9:00:54 AM	Not p
<input type="checkbox"/>	weather_data.csv	353 B	text/csv	May 10, 2024, 9:00:39 AM	Standard	May 10, 2024, 9:00:39 AM	Not p



Marketplace



Release Notes

Rows per page: 100

1 - 16 of 16





Google Cloud Functions



Cloud Functions

Functions



Filter Filter functions

<input type="checkbox"/>		Environment	Name ↑
<input type="checkbox"/>	✓	1st gen	dataset_consolidator
<input type="checkbox"/>	✓	1st gen	metadata_extractor
<input type="checkbox"/>	✓	1st gen	remote_sensing_data_fetcher
<input type="checkbox"/>	✓	1st gen	weather_data_fetcher

```
59 def fetch_remote_sensing_data(event, context):
60     """Triggered by the message from metadata_extractor cloud function"""
61     print("Raw event data:", event['data']) # Log the raw base64 message
62     pubsub_message = base64.b64decode(event['data']).decode('utf-8')
63     print("Decoded message:", pubsub_message) # Log the decoded string
64     message_data = json.loads(pubsub_message)
65     print("Message data:", message_data) # Log the dictionary
66
67     bucket_name = message_data["bucket"]
68     field_id = message_data["field_id"]
69     batch_id = message_data["batch_id"]
70
71     try:
72         # Initialize Earth Engine and Storage Client
73         ee.Initialize()
74         storage_client = storage.Client()
75
76         # Setup paths and download the metadata CSV to /tmp directory
77         bucket = storage_client.bucket(bucket_name)
78         local_dir = f"/tmp/{field_id}/{batch_id}/"
79         os.makedirs(local_dir, exist_ok=True)
80         local_path = os.path.join(local_dir, "image_metadata.csv")
81         blob = bucket.blob(f"userdata/{field_id}/{batch_id}/image_metadata.csv")
82         blob.download_to_filename(local_path)
83
84         # Read the CSV and process data
85         df = pd.read_csv(local_path)
86
87         # List which contains tuples which contain the Latitude, Longitude, and D
88         coordinates_and_dates = []
89
90         for _, row in df.iterrows():
91             lat = row['Latitude']
```



Docker container for hybrid model

The screenshot displays a code editor interface with three main panels. The left panel shows the 'EXPLORER' view with a file tree for a project named 'projectai'. The tree includes folders like 'app', 'cloud_functions', 'dataset_consolidator', 'metadata_extractor', 'remote_sensing_data_fetcher', 'weather_data_fetcher', 'etl', 'google-cloud-sdk', 'instance', 'old', 'userdata', and files like '.flaskenv', '.gcloudignore', '.gitignore', '.python-version', 'app.yaml', 'cert.py', 'cors-config.json', 'Dockerfile', 'fixed_Best_DenseNet121_Hybrid_Mod...', and 'h5file.py'. The 'Dockerfile' is selected and highlighted in blue.

The middle panel shows the 'app.py' file with the following code:


```
1 from flask import Flask, request, jsonify
2 import os
3 import pandas as pd
4 import numpy as np
5 import joblib
6 import tensorflow as tf
7 from tensorflow.keras.applications.densenet import preprocess_input as preprocess
8 from tensorflow.keras.models import load_model
9 from google.cloud import storage, pubsub_v1
10 import json
11 import shutil
12
13 app = Flask(__name__)
14
15 # Function to run the model with field_id and batch_id
16 def run_hybrid_model(field_id, batch_id, bucket_name):
17
18     gcs_base_path = f"gs://{bucket_name}"
19
20     # Initialize Google Cloud Storage client
21     storage_client = storage.Client()
22     bucket = storage_client.bucket(bucket_name)
23
24     # Set up directories
25     model_dir = "/tmp/model_artifacts"
26     os.makedirs(model_dir, exist_ok=True)
27     local_tmp_dir = f"/tmp/userdata/{field_id}/{batch_id}" # Local directory
28     os.makedirs(local_tmp_dir, exist_ok=True)
29
30     # Downloading model artifacts from GCS
31     blob = bucket.blob("model_artifacts/label_encoder_v2_hybrid_model.joblib")
32     blob.download_to_filename(os.path.join(model_dir, "label_encoder_v2_hybrid_model.joblib"))
33     blob = bucket.blob("model_artifacts/scaler.joblib")
34     blob.download_to_filename(os.path.join(model_dir, "scaler.joblib"))
35     blob = bucket.blob("model_artifacts/Best_DenseNet121_Hybrid_Model.h5")
36     blob.download_to_filename(os.path.join(model_dir, "Best_DenseNet121_Hybrid_Model.h5"))
37
38     # Loading model components
39     label_encoder = joblib.load(os.path.join(model_dir, 'label_encoder_v2_hybrid_model.joblib'))
40     scaler = joblib.load(os.path.join(model_dir, 'scaler.joblib'))
41     model = load_model(os.path.join(model_dir, "Best_DenseNet121_Hybrid_Model.h5"))
```


The right panel shows the 'Dockerfile' with the following code:

```
1 # Use an official Python runtime as a base image
2 FROM python:3.11-slim
3
4 # Install necessary tools
5 RUN apt-get update && apt-get install -y curl && rm -rf /var/lib/apt/lists/*
6
7 # Set the working directory in the container
8 WORKDIR /app
9
10 # Copy the 'ai_gcp' directory contents into the container at /app
11 COPY ai_gcp/ /app/
12
13 # Install any needed packages specified in requirements.txt
14 RUN pip install --no-cache-dir -r requirements.txt
15
16 # Make port 8080 available to the world outside this container
17 EXPOSE 8080
18
19 # Define environment variable
20 ENV FLASK_APP app.py
21
22 # Optionally define a command for health checks
23 HEALTHCHECK --interval=30s --timeout=30s --start-period=5s --retries=3
24 CMD curl -f http://localhost:8080/health || exit 1
25
26 # Run Gunicorn to serve the Flask app
27 CMD ["gunicorn", "-b", "0.0.0.0:8080", "app:app"]
```





Google Vertex AI


Vertex AI





BUILD WITH GEN AI

Extensions


DATA


Feature Store

Datasets

Labeling tasks

MODEL DEVELOPMENT

Training

[<](#) hybrid-model > Version 1  [EXPORT](#)

EVALUATE

DEPLOY & TEST



BATCH PREDICT

VERSION DET

Deploy your model

Endpoints are machine learning models made available for online prediction requests. Endpoints are useful for timely predictions from many users (for example, in response to an application request). You can also request batch predictions if you don't need immediate results.

DEPLOY TO ENDPOINT

	Name	ID	Status	Models
	hybrid-model-endpoint	4526755342251458560	 Active	1



YAML deployment configuration for App Engine

The screenshot shows a VS Code editor with three panels. The left panel displays the Explorer view with a project structure. The middle panel shows the Python code for `__init__.py`. The right panel shows the `Dockerfile`.

EXPLORER

- PROJECT (WORKSPACE)
- project
- app
 - auth.py
 - config.py
 - file_utils.py
 - main.py
 - models.py
 - services.py
- cloud_functions
 - dataset_consolidator
 - main.py
 - requirements.txt
- metadata_extractor
 - main.py
 - requirements.txt
- remote_sensing_data_fetcher
 - main.py
 - requirements.txt
- weather_data_fetcher
 - main.py
 - requirements.txt

- etl
- google-cloud-sdk
- instance
- old
- userdata
- .flaskenv
- .gcloudignore
- .gcloudignore_
- .gitignore
- .python-version
- app.yaml

project > app > __init__.py

```
1 from flask import Flask, render_template, flash, redirect, url_for
2 from flask_sqlalchemy import SQLAlchemy
3 from flask_login import LoginManager
4 from .config import Config
5 import logging
6 from google.cloud import storage
7 import sqlalchemy
8 from sqlalchemy import create_engine
9 from sqlalchemy.orm import scoped_session, sessionmaker
10 from google.cloud.sql.connector import Connector, IPTypes
11 import pymysql
12 import google.cloud.logging
13
14 db = SQLAlchemy()
15
16 # Initialize login manager outside the create_app function to be accessible
17 login_manager = LoginManager()
18
19 # Set up Google Cloud Logging
20 def setup_logging():
21     client = google.cloud.logging.Client()
22     # Attaches a Google Cloud Logging handler to the root logger
23     client.setup_logging()
24     logging.basicConfig(level=logging.DEBUG, format='%(asctime)s %(levelname)s')
25
26 def connect_with_connector() -> sqlalchemy.engine.base.Engine:
27     """
28     Initializes a connection pool for a Cloud SQL instance of MySQL.
29
30     Uses the Cloud SQL Python Connector package.
31     """
32     connector = Connector(IPTypes.PUBLIC)
33
34     def getconn() -> pymysql.connections.Connection:
35         return connector.connect(
36             Config.DB_CONN_NAME,
37             "pymysql",
38             user=Config.DB_USER,
39             password=Config.DB_PASS,
40             db=Config.DB_NAME
```

project > app.yaml

```
1 # Runtime environment specifying Python 3.11
2 runtime: python311
3
4 # Command to start your Flask application with Gunicorn as the WSGI server
5 entrypoint: gunicorn -b :$PORT 'app:app'
6
7 # Environment variables configuration
8 env_variables:
9     FLASK_ENV: 'production' # Specifies the environment (production, dev)
10     ENV_MODE: 'production' # Custom environment variable to specify operation
11
12 # Handlers define rules for serving static files and routing URLs
13 handlers:
14     - url: /static # URL path that should be treated as static
15       static_dir: static/ # Directory in the project that contains static files
16     - url: /* # Catch-all pattern for all other URLs
17       script: auto # Automatically route requests to your Flask app
18
19 # Instance class specification for App Engine
20 instance_class: F1 # Specifies the F1 class which is covered under the
21
22 # Automatic scaling configuration for managing the scaling behavior of your
23 automatic_scaling:
24     target_cpu_utilization: 0.90 # Target CPU utilization before scaling
25     max_instances: 1 # Maximum number of instances to scale to
26     min_idle_instances: 0 # Minimum number of instances that are always running
27     max_idle_instances: 1 # Maximum number of idle instances
28     min_pending_latency: 30s # Minimum time a request waits in the queue
29     max_pending_latency: automatic # Maximum time a request is allowed to wait
30
31 # Environment type, standard or flexible; standard does not allow custom
32 env: standard
33
34 # Beta settings are used to access beta features
35 #beta_settings:
36     # cloud_sql_instances: "tidy-nomad-415320:us-west1:mainsql" # Connect to
37     # cloud_sql_instances: "tidy-nomad-415320:us-west1:mainsql"
```




Google App Engine

App Engine

Dashboard

Services

Versions

Instances

Task queues

Cron jobs

Security scans

Firewall rules

Quotas

Memcache

Search

Settings

Versions

REFRESH

DELETE

STOP

START

MIGRATE TRAFFIC

SPLIT TRAFFIC

Filter

Filter versions

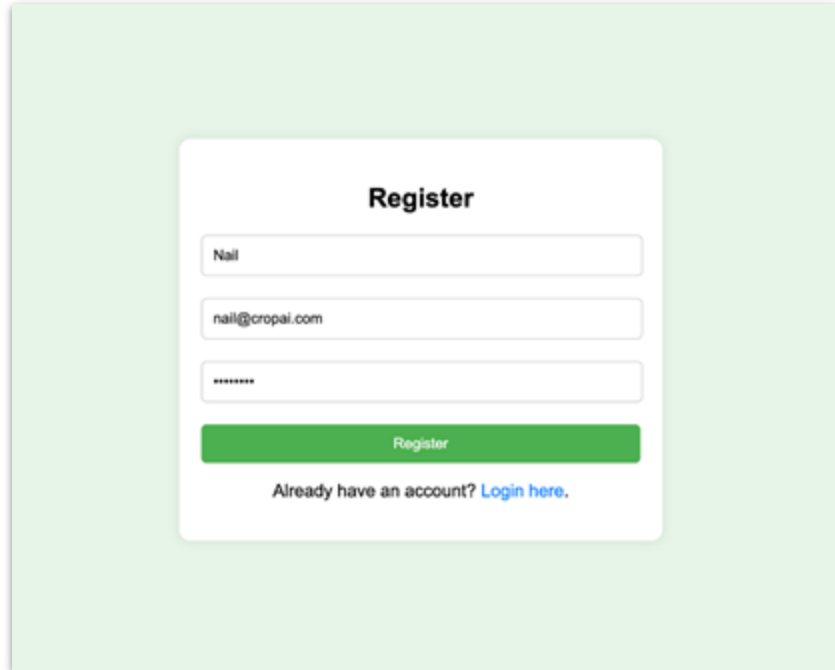
<input type="checkbox"/>	Version	Status	Traffic Allocation	Instances ?	Runtime	Environment
<input type="checkbox"/>	20240510t092951	Serving	<div><div></div></div> 100%	1	python311	Standard
<input type="checkbox"/>	20240510t091525	Serving	<div><div></div></div> 0%	0	python311	Standard
<input type="checkbox"/>	20240510t085815	Serving	<div><div></div></div> 0%	0	python311	Standard
<input type="checkbox"/>	20240509t212704	Serving	<div><div></div></div> 0%	0	python311	Standard
<input type="checkbox"/>	20240509t202955	Serving	<div><div></div></div> 0%	0	python311	Standard
<input type="checkbox"/>	20240509t200800	Serving	<div><div></div></div> 0%	0	python311	Standard

Extensive Logging

The screenshot displays the Google Cloud Logs Explorer interface. On the left sidebar, there are navigation icons and a list of resource types with their respective counts: Cloud Build (731), GAE Application (223), Cloud Function (56), Vertex AI Endpoint (20), Cloud SQL Database (9), Audited Resource (3), and GCS Bucket (1). The main panel is titled 'Logs Explorer' and includes a 'Refine scope' button. Below this, there are tabs for 'Query', 'Recent (8)', 'Saved (0)', 'Suggested (3)', and 'Library'. A search bar is present with the text 'Search all fields'. A timeline view is active, showing a time range from 5/9/24, 6:46 PM to 7:46 PM. A search bar within the timeline says 'Search fields and values'. Below the timeline, a table displays 1,043 results. The table has columns for SEVERITY, TIME, and SUMMARY. The first few rows show log entries for a Cloud Function starting an application, including details like session ID (R5K6WPG) and application start info.

User Interface

Registration Page



The registration page features a white card on a light green background. The card has a title 'Register' in bold black text. Below the title are three input fields: the first contains 'Nail', the second contains 'nail@cropai.com', and the third contains masked characters '*****'. A green button with the text 'Register' is positioned below the input fields. At the bottom of the card, there is a link that says 'Already have an account? [Login here.](#)'.

Register

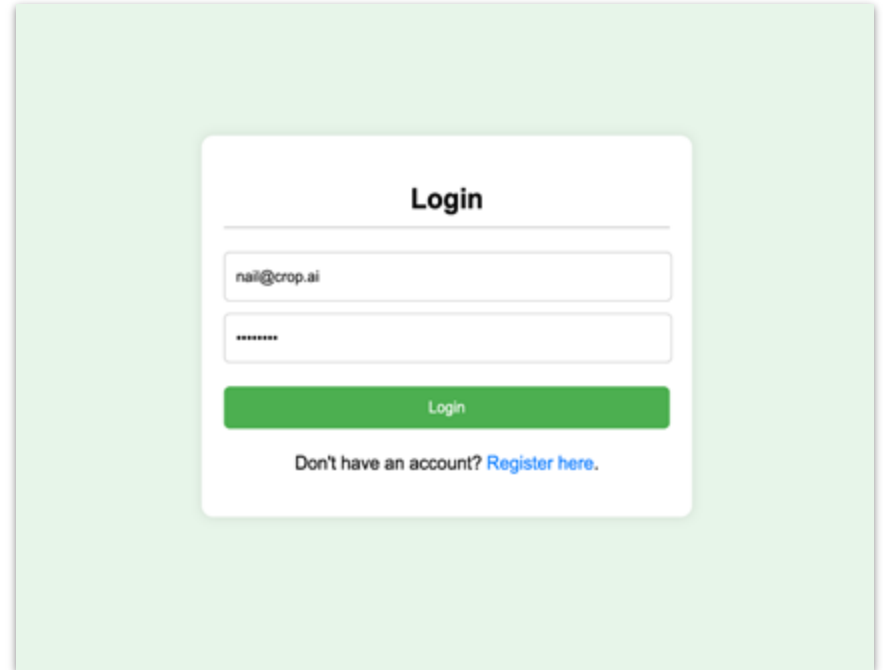
Nail

nail@cropai.com

Register

Already have an account? [Login here.](#)

Login Page



The login page features a white card on a light green background. The card has a title 'Login' in bold black text. Below the title are two input fields: the first contains 'nail@cropai' and the second contains masked characters '*****'. A green button with the text 'Login' is positioned below the input fields. At the bottom of the card, there is a link that says 'Don't have an account? [Register here.](#)'.

Login

nail@cropai

Login

Don't have an account? [Register here.](#)

Main Page

Field Section

Welcome, Nail! [Logout](#)

Add Field

Field ID	Field Name	User Name	Date Created
----------	------------	-----------	--------------

Add Field Form

Welcome, Nail! [Logout](#)

Add Field

Field ID	Field Name	User Name	Date Created
----------	------------	-----------	--------------

Add Field

Field Name

001 Taiwan

Ok

Batch Section

Welcome, Nail! [Logout](#)

Add Field

Field ID	Field Name	User Name	Date Created
82	001 Taiwan	Nail	2024-04-08 19:09:23

Upload Batch

Batch ID	Image Quantity	Images Taken	User Name	Date Uploaded
----------	----------------	--------------	-----------	---------------

Upload Batch Form

Welcome, Nail! [Logout](#)

Add Field

Field ID	Field Name		
82	001 Taiwan		

Upload Batch

Select Images

Choose Files 20 files

Upload

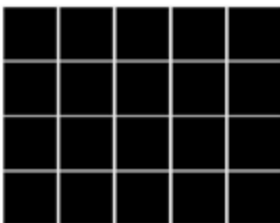
Upload Batch

Batch ID	Image Quantity	Images Taken	User Name	Date Uploaded
----------	----------------	--------------	-----------	---------------

Data Extraction + Connecting the Model

Upload Batch

Batch ID	Image Quantity	Images Taken	User Name	Date Uploaded
95	20	2024-02-01	Nail	2024-04-08 19:26:00



Images

Image ID	File Name	Label	Healthy	Rice Blast	Brown Spot
795	001_4_5_1_2024-02-01.JPG	no	-	-	-
796	001_4_5_2_2024-02-01.JPG	no	-	-	-
797	001_4_5_3_2024-02-01.JPG	no	-	-	-
798	001_4_5_4_2024-02-01.JPG	no	-	-	-

Data Fetching in Progress

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.

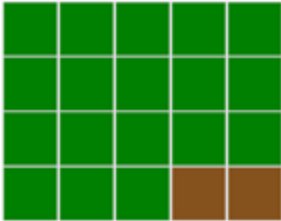
Finally, the consolidated dataset is prepared and along with the images passed to the hybrid model for predictions.

Classification Result

Healthy
Brown Spot
Leaf Blast

Upload Batch

Batch ID	Image Quantity	Images Taken	User Name	Date Uploaded
100	20	2024-02-01	Nail	2024-04-08 19:48:41



Images

Image ID	File Name	Label	Healthy	Rice Blast	Brown Spot
911	001_4_5_17_2024-02-01.jpg	Healthy	0.966	0.0339	0.0001
912	001_4_5_18_2024-02-01.jpg	Healthy	0.966	0.0339	0.0001
913	001_4_5_19_2024-02-01.jpg	Brown Spot	0	0.0019	0.998
914	001_4_5_20_2024-02-01.jpg	Brown Spot	0	0.0012	0.9988

Disease Progression Results

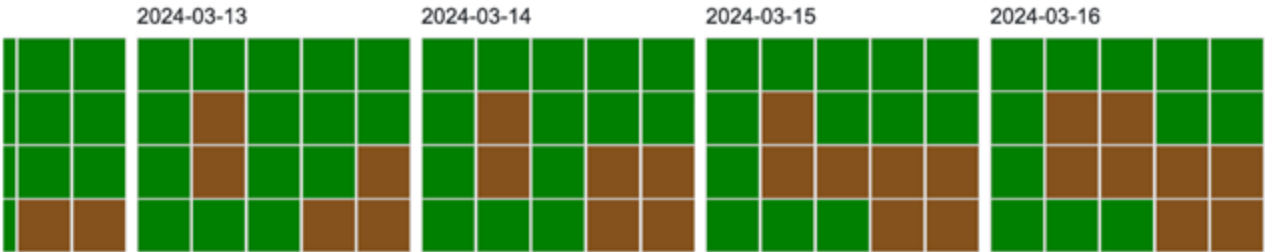
 Healthy  Brown Spot  Leaf Blast

Welcome, Nail! [Logout](#)

Add Field

Field ID	Field Name	User Name	Date Created
1	Taiwan 115	Jason	2024-05-08 22:28:28

Disease progression



Upload Batch

Batch ID	Image Quantity	Images Taken	User Name	Date Uploaded
1	20	2024-03-10	Jason	2024-05-08 22:30:30
2	20	2024-03-11	Jason	2024-05-09 22:31:59
3	20	2024-03-12	Jason	2024-05-09 22:32:32
4	20	2024-03-13	Jason	2024-05-09 22:32:58

A single, elongated green leaf with a pointed tip and a small notch at the base, oriented horizontally. The leaf has a vibrant green color with visible veins. The text "Demo Video" is centered on the leaf in a bold, black, sans-serif font.

Demo Video

A single, elongated green leaf with a pointed tip and a small notch at the base, oriented horizontally. The leaf has a vibrant green color with visible veins. The text "Thank You!" is written in a bold, black, sans-serif font across the center of the leaf.

Thank You!

References found in report