

Plant health monitoring with photos based on Deep Learning

Course Title: Deep Learning

SUBMITTED BY	SUBMITTED TO
<p>Md Akram Student id: 2210008 Hamm-Lippstadt University of Applied Science.</p>	<p>Prof. Dr. Achim Rettberg Hamm-Lippstadt University of Applied Science.</p>

Motivation

01

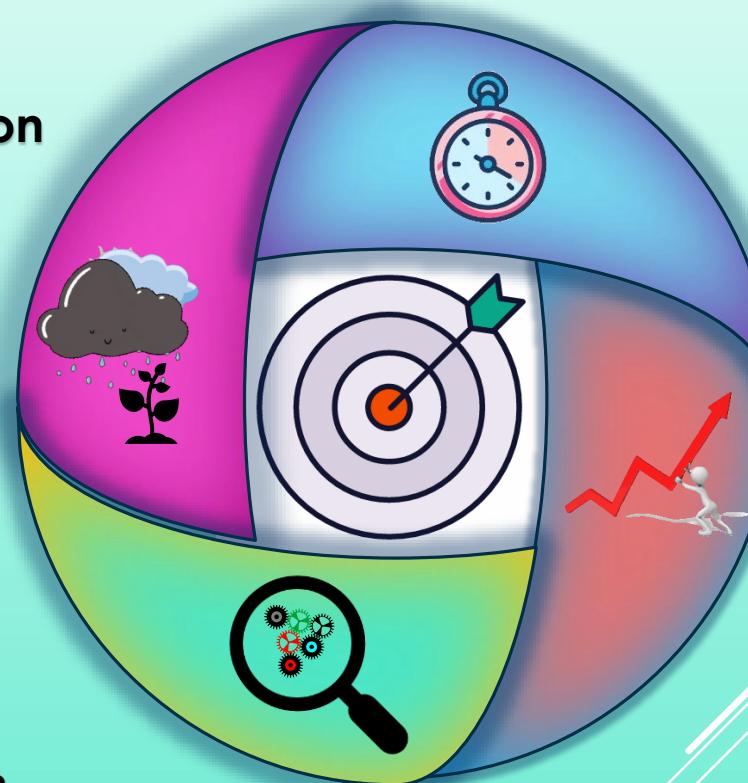
Plant, Weed & Parasite Identification

Monitoring plant leaves to identify plant categories, Weed and parasite.

02

Accurate Results Prediction

I want to find out a state forward results like name of the Plant, weed or parasite



03

Design a Model

For precision farming, I want to design a TAPAAL Model

04

Verification & Simulation

I want to check some verification process like deadlock free or not and Some simulation.

Motivation

01

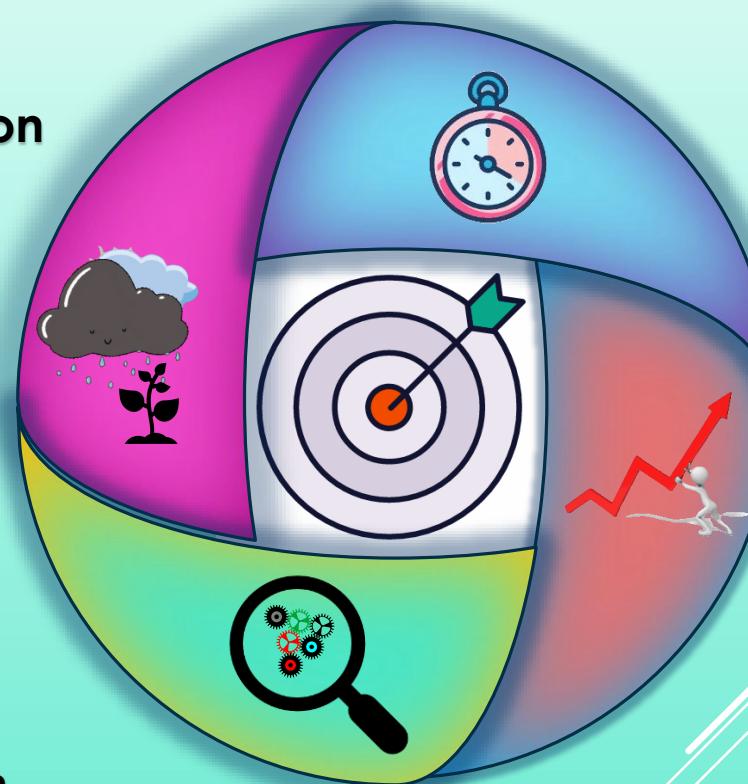
Plant, Weed & Parasite Identification

Monitoring plant leaves to identify plant categories, Weed and parasite.

02

Accurate Results Prediction

I want to find out a state forward results like name of the Plant, weed or parasite



03

Design a Model

For precision farming, I want to design a TAPAAL Model

04

Verification & Simulation

I want to check some verification process like deadlock free or not and Some simulation.

Motivation

01

Plant, Weed & Parasite Identification

Monitoring plant leaves to identify plant categories, Weed and parasite.

02

Accurate Results Prediction

I want to find out a state forward results like name of the Plant, weed or parasite



03

Design a Model

For precision farming, I want to design a TAPAAL Model

04

Verification & Simulation

I want to check some verification process like deadlock free or not and Some simulation.

Motivation

01

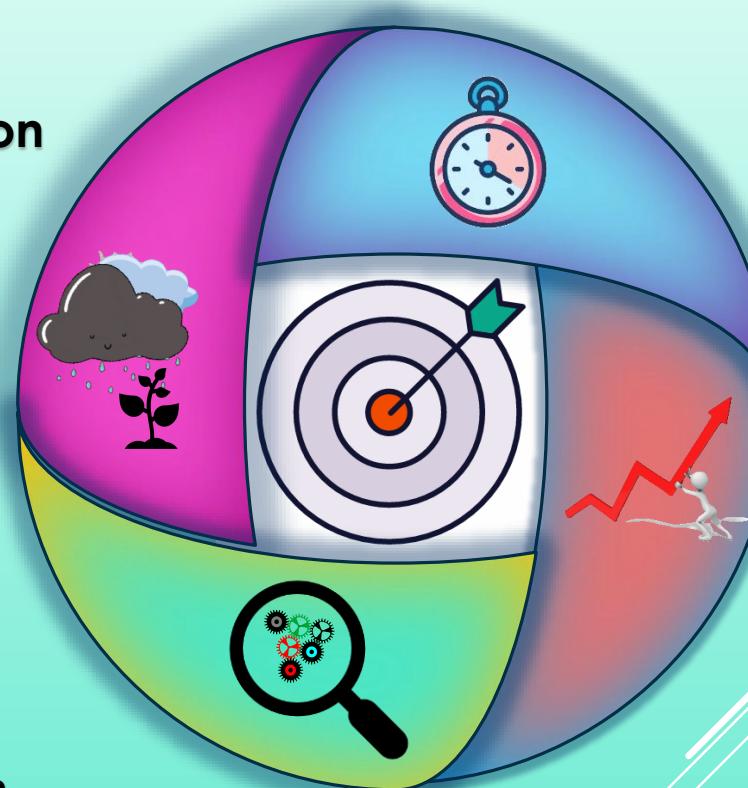
Plant, Weed & Parasite Identification

Monitoring plant leaves to identify plant categories, Weed and parasite.

02

Accurate Results Prediction

I want to find out a state forward results like name of the Plant, weed or parasite



03

Design a Model

For precision farming, I want to design a TAPAAL Model

04

Verification & Simulation

I want to check some verification process like deadlock free or not and Some simulation.

Plant Health & Deep Learning

Plant Health:

Refers to the condition of a plant including healthy or diseases affected.

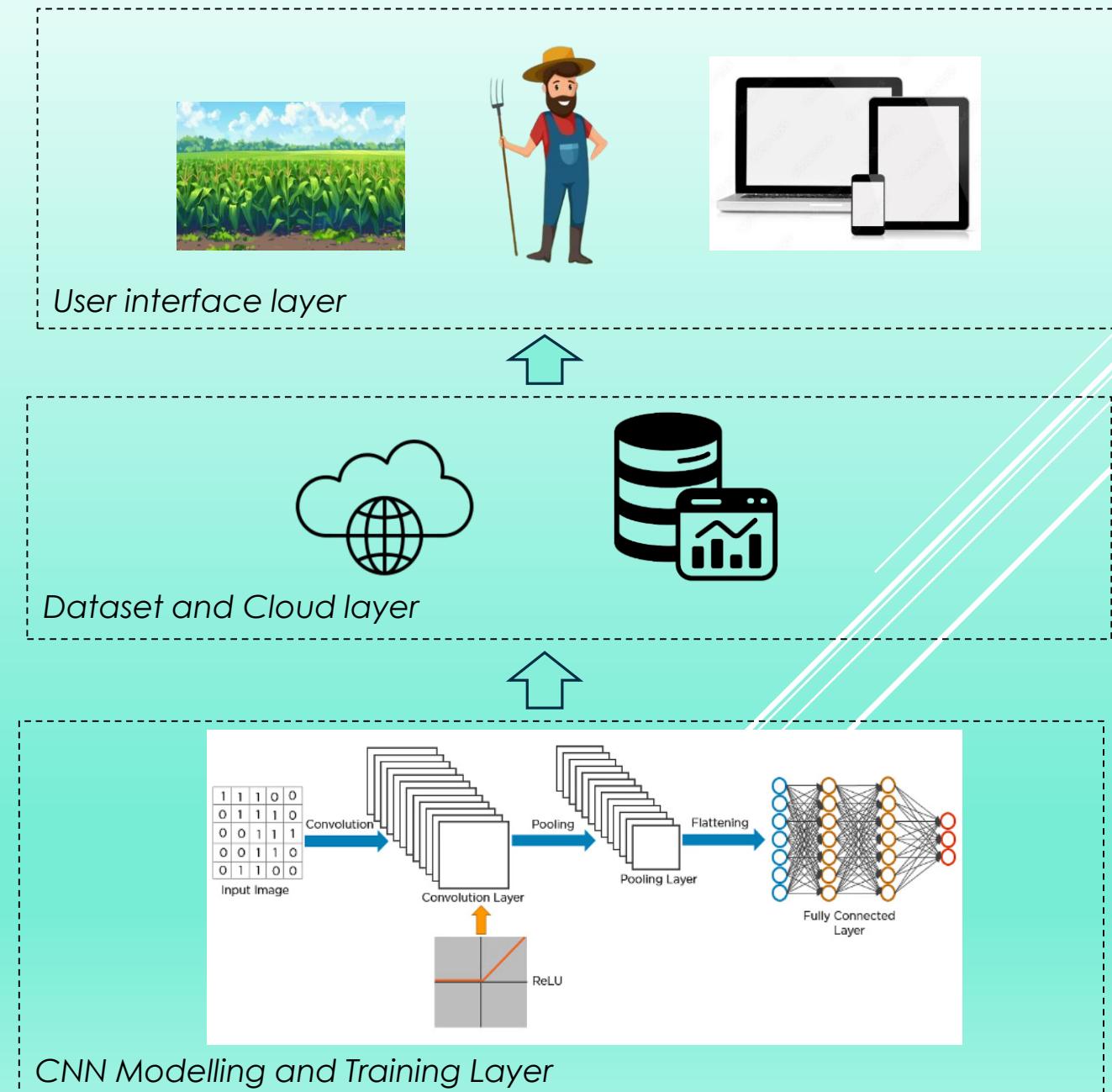


Deep Learning:

A subset of machine learning that uses neural networks to automatically learn patterns from data



System Design



CNN Structure

01

Convolution & ReLU

In the convolution layer according, several feature and filter we get several feature maps for each channel of RGB. Finally, after combining all kernel channel and bias we get final feature map. Then for making our feature map non-linear and make training process first we can use Relu (Rectified Linear Unit).

02

Pooling layer

This layer use for reduce image size and dimension as well as reduce over feeding since less parameter. There are total two polling method possible including max polling and average polling.

03

Fully connected Layer

This layer is responsible for classifying the images. In the Flatten layer we get one dimensional array and in this layer that is connected with neurons and finally give a output layer with prediction for each class.



CNN Structure

01

Convolution & ReLU

In the convolution layer according, several feature and filter we get several feature maps for each channel of RGB. Finally, after combining all kernel channel and bias we get final feature map. Then for making our feature map non-linear and make training process easy and first we can use Relu (Rectified Linear Unit).

02

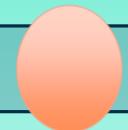
Pooling layer

This layer use for reduce image size and dimension as well as reduce over feeding since less parameter. There are total two polling method possible including max polling and average polling.

03

Fully connected Layer

This layer is responsible for classifying the images. In the Flatten layer we get one dimensional array and in this layer that is connected with neurons and finally give a output layer with prediction for each class.



CNN Structure

01

Convolution & ReLU

In the convolution layer according, several feature and filter we get several feature maps for each channel of RGB. Finally, after combining all kernel channel and bias we get final feature map. Then for making our feature map non-linear and make training process easy and first we can use Relu (Rectified Linear Unit).

02

Pooling layer

This layer use for reduce image size and dimension as well as reduce over feeding since less parameter. There are total two polling method possible including max polling and average polling.

03

Fully connected Layer

This layer is responsible for classifying the images. In the Flatten layer we get one dimensional array and in this layer that is connected with neurons and finally give a output layer with prediction for each classes.



CNN Structure of My Model

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	896
conv2d_1 (Conv2D)	(None, 126, 126, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_2 (Conv2D)	(None, 63, 63, 64)	18,496
conv2d_3 (Conv2D)	(None, 61, 61, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_4 (Conv2D)	(None, 30, 30, 128)	73,856
conv2d_5 (Conv2D)	(None, 28, 28, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_6 (Conv2D)	(None, 14, 14, 256)	295,168
conv2d_7 (Conv2D)	(None, 12, 12, 256)	590,080
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 256)	0
conv2d_8 (Conv2D)	(None, 6, 6, 512)	1,180,160
conv2d_9 (Conv2D)	(None, 4, 4, 512)	2,359,808
max_pooling2d_4 (MaxPooling2D)	(None, 2, 2, 512)	0
dropout (Dropout)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 2500)	5,122,500
dropout_1 (Dropout)	(None, 2500)	0
dense_1 (Dense)	(None, 38)	95,038

Total params: 9,929,762 (37.88 MB)
Trainable params: 9,929,762 (37.88 MB)
Non-trainable params: 0 (0.00 B)

Dataset

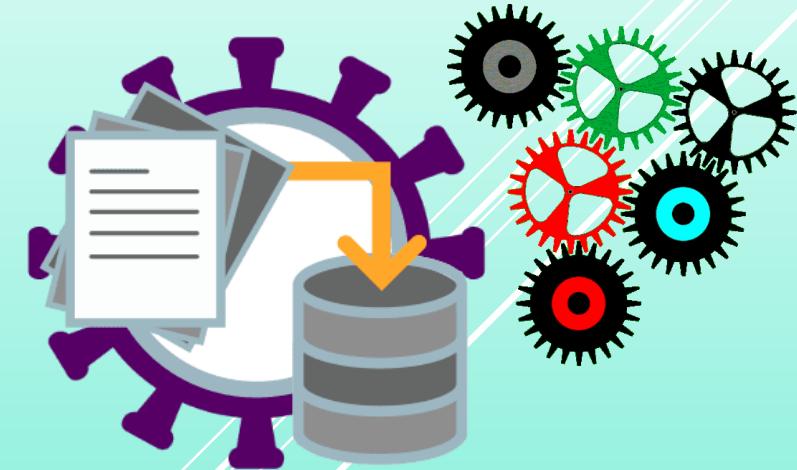
Dataset Source

- Kaggle

Dataset asset

- 14 Plants
- 38 Categories of disease
- 87.9K Training, Validation & Testing image

Available: <https://www.kaggle.com/datasets/vipooooool/new-plant-diseases-dataset/data>



CNN Implementation

Used tools

- Python
- TensorFlow
- Keras

Training Process

- Separate directory for training and validation
- Total 10 Epochs
- After getting desire accuracy and losses stopped training

```
training_history = cnn.fit(x=training_set,validation_data=validation_set,epochs=10)

Epoch 1/10
2197/2197 2809s 1s/step - accuracy: 0.4094 - loss: 2.1314 - val_accuracy: 0.8234 - val_loss: 0.5801
Epoch 2/10
2197/2197 2678s 1s/step - accuracy: 0.8400 - loss: 0.5077 - val_accuracy: 0.9095 - val_loss: 0.2814
Epoch 3/10
2197/2197 2629s 1s/step - accuracy: 0.9072 - loss: 0.2897 - val_accuracy: 0.9334 - val_loss: 0.2075
Epoch 4/10
2197/2197 2615s 1s/step - accuracy: 0.9364 - loss: 0.1961 - val_accuracy: 0.9537 - val_loss: 0.1506
Epoch 5/10
2197/2197 2619s 1s/step - accuracy: 0.9513 - loss: 0.1479 - val_accuracy: 0.9533 - val_loss: 0.1431
Epoch 6/10
2197/2197 2624s 1s/step - accuracy: 0.9656 - loss: 0.1053 - val_accuracy: 0.9535 - val_loss: 0.1536
Epoch 7/10
2197/2197 2623s 1s/step - accuracy: 0.9687 - loss: 0.0953 - val_accuracy: 0.9520 - val_loss: 0.1586
Epoch 8/10
2197/2197 2617s 1s/step - accuracy: 0.9750 - loss: 0.0741 - val_accuracy: 0.9612 - val_loss: 0.1289
Epoch 9/10
2197/2197 2627s 1s/step - accuracy: 0.9768 - loss: 0.0665 - val_accuracy: 0.9641 - val_loss: 0.1204
Epoch 10/10
2197/2197 2610s 1s/step - accuracy: 0.9813 - loss: 0.0585 - val_accuracy: 0.9717 - val_loss: 0.1038
```

Training Epochs

Web App Implementation

Used tools

- Streamlit
- Python
- Trained Model

Benefits

- Easy to Use
- Real-Time Results
- Free of Cost

The screenshot shows a web application interface for a Plant Health Monitoring System. On the left, there's a sidebar with a 'Dashboard' section and a 'Select Page' dropdown menu set to 'Home'. The main content area has a title 'PLANT HEALTH MONITORING SYSTEM' above a photograph of a hand holding a smartphone displaying a plant image. Surrounding the phone are several circular icons representing environmental factors: a sun, water droplets, a leaf, and a thermometer showing 20°. Below the image, a welcome message reads 'Welcome to the Plant Health Monitoring System! 🌱🔍' and a mission statement says 'Our mission is to help in plant health monitoring efficiently by using photos. After upload a image of a plant then our system will analyze it to give a current health condition.' To the right, there's a 'How It Works' section with three numbered steps: 1. Upload Image, 2. Analysis, 3. Results. At the bottom, there's a 'Why Choose Us?' section with a bullet point about accuracy.

PLANT HEALTH MONITORING SYSTEM

Welcome to the Plant Health Monitoring System! 🌱🔍

Our mission is to help in plant health monitoring efficiently by using photos. After upload a image of a plant then our system will analyze it to give a current health condition.

How It Works

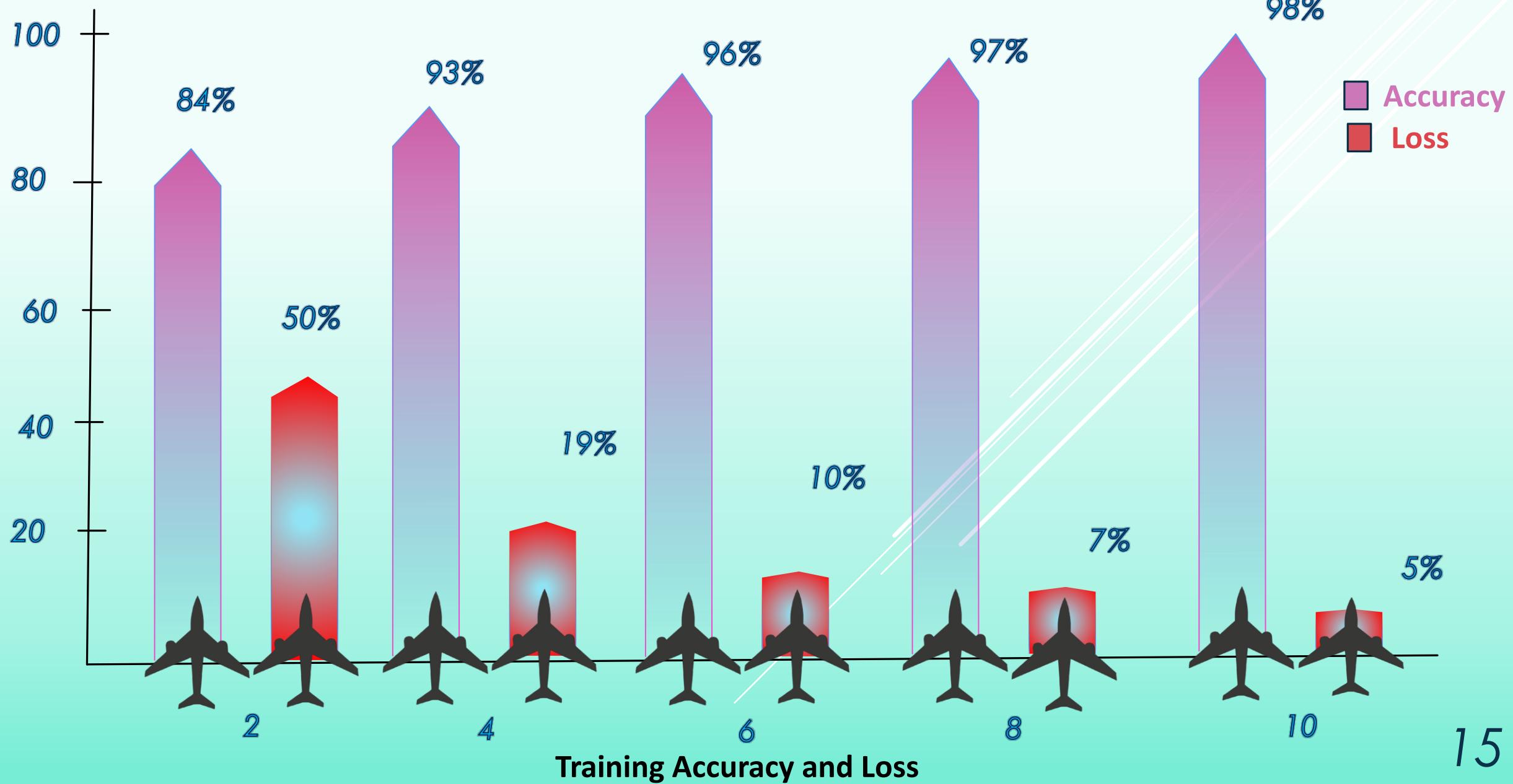
1. Upload Image: Go to the Plant Health Monitoring page and upload an image of a plant.
2. Analysis: Our system will process the image using advanced algorithms to monitor the current status.
3. Results: After upload image click predict button to see the plant health condition.

Why Choose Us?

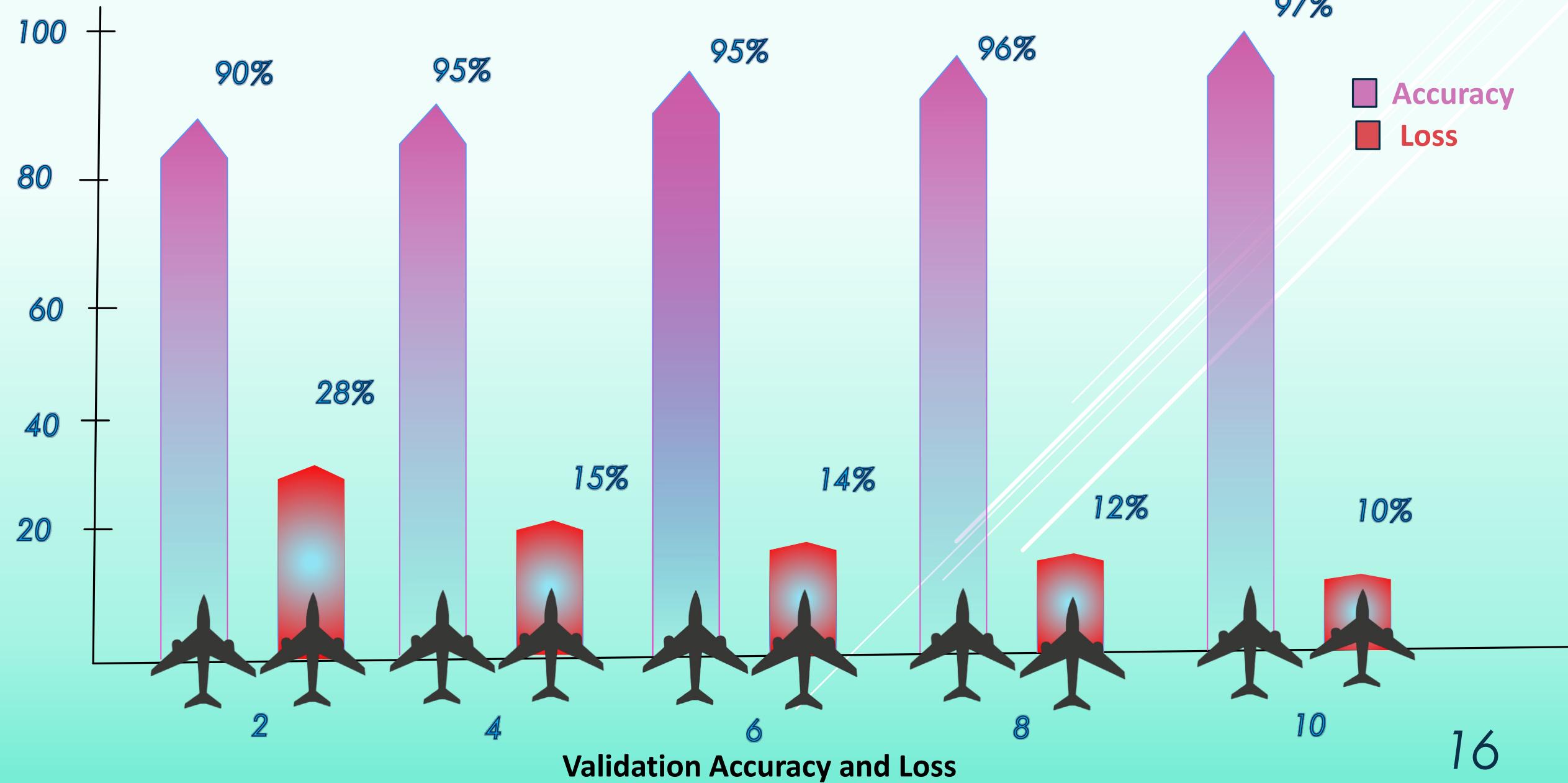
- Accuracy: Our system utilizes state-of-the-art deep learning learning techniques for accurate disease

Web App Interface

Experimental Evaluation



Experimental Evaluation



Experimental Evaluation

Test Image



Plant Health Condition: Potato__healthy

```
print(predictions)
[[6.6111966e-08 5.4390894e-06 1.5594614e-09 7.1823865e-04 2.6166587e-08
 3.0979113e-06 1.1432533e-02 7.3854933e-10 1.3162514e-07 9.4102792e-10
 1.0811150e-08 5.4947336e-08 1.0528570e-08 1.2353928e-08 7.9848093e-08
 2.5216629e-08 5.1544259e-08 2.6699020e-08 1.3879119e-06 9.9325948e-04
 5.8731601e-08 7.4177037e-04 9.8544484e-01 1.3231684e-07 3.8564927e-04
 3.0064040e-10 4.3257298e-10 9.2290684e-08 1.2734984e-07 7.3328607e-05
 1.4863815e-04 7.4040241e-07 1.1652932e-06 2.4268592e-08 4.8984628e-05
 1.1520246e-08 6.4506098e-09 5.7088261e-08]]
```

```
result_index = np.argmax(predictions) #Return index of max element
print(result_index)
```

22 ←

Test Image



Plant Health Condition: Corn_(maize)__Common_rust

```
print(predictions)
[[4.6991249e-16 2.5931651e-15 1.2982614e-16 5.2627477e-17 1.3401064e-16
 5.1188389e-18 5.7535750e-14 1.1435991e-15 9.9999821e-01 5.0928193e-17
 6.4495820e-16 1.2870855e-19 1.1679373e-19 2.7404891e-20 1.2820970e-19
 2.7717185e-17 5.1281388e-19 2.8329117e-20 2.0930750e-15 8.0030646e-15
 3.5939327e-12 1.2303068e-20 5.8832187e-16 1.2144660e-21 1.5307793e-20
 2.2895894e-20 1.4781300e-19 3.7306602e-20 4.3242981e-17 1.7065700e-14
 1.7690439e-06 1.4759901e-17 2.8917260e-15 1.3386201e-23 4.1805963e-20
 2.0074010e-22 4.2171895e-23 8.8589196e-16]]
```

```
result_index = np.argmax(predictions) #Return index of max element
print(result_index)
```

8 ←

Model Prediction Results

Experimental Evaluation

Dashboard

Select Page

Home

PLANT HEALTH MONITORING SYSTEM



Welcome to the Plant Health Monitoring System! 🌱🔍

Our mission is to help in plant health monitoring efficiently by using photos. After upload a image of a plant then our system will analyse it to give a current health condition.

How It Works

1. Upload Image: Go to the Plant Health Monitoring page and upload an image of a plant.
2. Analysis: Our system will process the image using advanced algorithms to monitor the current status.
3. Results: After upload image click predict button to see the plant health condition.

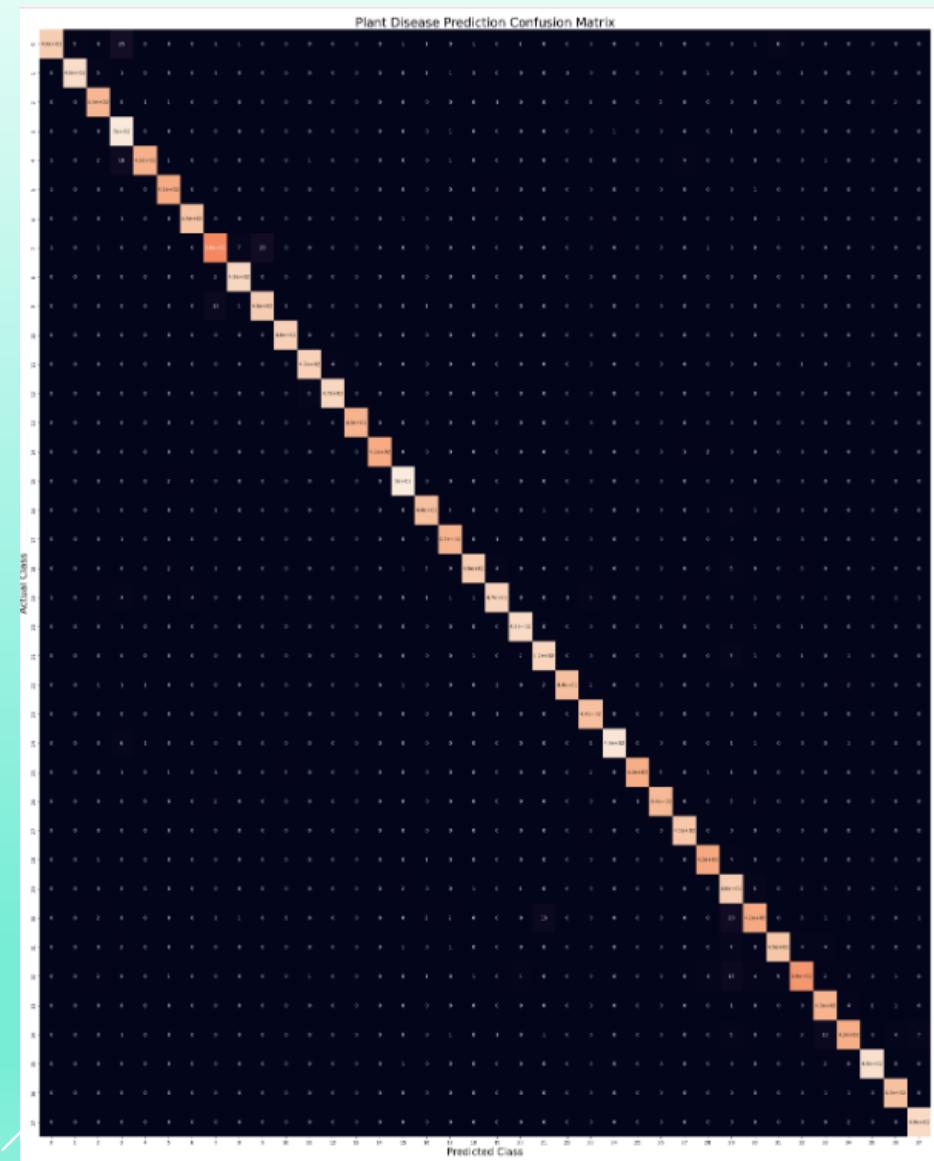
Why Choose Us?

- Accuracy: Our system utilises state-of-the-art deep learning learning techniques for accurate disease

Experimental Evaluation

		precision	recall	f1-score
1				
2	Apple__Apple_scab	0,991379	0,912698	0,950413
3	Apple__Black_rot	0,989733	0,969819	0,979675
4	Apple__Cedar_apple_rust	0,962138	0,981818	0,971879
5	Apple__healthy	0,892665	0,994024	0,940622
6	Blueberry__healthy	0,993007	0,938326	0,964892
7	Cherry_(including_sour)_Powdery_mildew	0,981176	0,990499	0,985816
8	Cherry_(including_sour)_healthy	0,991189	0,986842	0,989011
9	Corn_(maize)_Cercospora_leaf_spot_Gray_leaf_spot	0,9425	0,919512	0,930864
10	Corn_(maize)_Common_rust_	0,979381	0,995807	0,987526
11	Corn_(maize)_Northern_Leaf_Blight	0,952479	0,966457	0,959417
12	Corn_(maize)_healthy	0,989362	1	0,994652
13	Grape__Black_rot	0,978992	0,987288	0,983122
14	Grape__Esca_(Black_Measles)	0,991632	0,9875	0,989562
15	Grape__Leaf_blight_(Isariopsis_Leaf_Spot)	1	0,995349	0,997669
16	Grape__healthy	1	0,985816	0,992857
17	Orange__Haunglongbing_(Citrus_greening)	0,984283	0,996024	0,990119
18	Peach__Bacterial_spot	0,975771	0,965142	0,970427
19	Peach__healthy	0,975057	0,99537	0,985109
20	Pepper_bell__Bacterial_spot	0,982869	0,960251	0,971429
21	Pepper_bell__healthy	0,975155	0,947686	0,961224
22	Potato__Early_blight	0,983539	0,985567	0,984552
23	Potato__Late_blight	0,965235	0,973196	0,969199
24	Potato__healthy	0,995516	0,973684	0,984479
25	Raspberry__healthy	0,96732	0,997753	0,982301
26	Soybean__healthy	0,991968	0,978218	0,985045
27	Squash__Powdery_mildew	0,995272	0,970046	0,982497
28	Strawberry__Leaf_scorch	0,984199	0,981982	0,983089
29	Strawberry__healthy	0,980562	0,995614	0,98803
30	Tomato__Bacterial_spot	0,976526	0,978824	0,977673
31	Tomato__Early_blight	0,858473	0,960417	0,906588
32	Tomato__Late_blight	0,956221	0,896328	0,925307
33	Tomato__Leaf_Mold	0,969892	0,959574	0,964706
34	Tomato__Septoria_leaf_spot	0,953995	0,90367	0,928151
35	Tomato__Spider_mites_Two-spotted_spider_mite	0,938596	0,983908	0,960718
36	Tomato__Target_Spot	0,956818	0,921225	0,938685
37	Tomato__Tomato_Yellow_Leaf_Curl_Virus	0,991853	0,993878	0,992864
38	Tomato__Tomato_mosaic_virus	0,97807	0,995536	0,986726
39	Tomato__healthy	0,981557	0,995842	0,988648
40	accuracy	0,971659	0,971659	0,971659
41	macro avg	0,972484	0,971618	0,971725
42	weighted avg	0,97238	0,971659	0,971683

Precision, recall & f1-score



Confusion Matrix

Conclusion & Future Work

The System can successfully Identifies plant health issues with high accuracy

It prove a user friendly interactive web app for farmer

The System saving time and Money as well as increase production.

In future I want to integrate real-time monitoring and IOT.



**Do you have any
question?**

