Plant health monitoring with photos based on Deep Learning

Md Akram
Electronic Engineering
Hochshule Hamm Lippstadt
Hamm, Germany
md.akram@stud.hshl.de

***Abstract*—According with the increase of population growth the demand of food is also increase. Agriculture provide food for the human beings. Agriculture is the main occupation for a large portion of people in many Asian and African countries. Due to the plant diseases around one third of crops production losses occurs in united states. So controlling plants disease is one of the biggest challenges for the entire world. The improvement of uses deep learning in agriculture sector help to identify and classification of plant diseases. Deep learning help to monitoring and detecting current condition of plants with high accuracy. In this paper I discuss the process of plant health monitoring base on deep learning. I developed a system by using Convolutional Neural Networks (CNN) as an underlying deep learning engine for monitoring plant health and identify plant disease. I use a photo base dataset more than 87 thousand of picture of different plants and leaves base on healthy and infection for training, validating as well as testing in my CNN model. Finally, I created a web application as a user interface where user can upload images of leaves and then they can monitor the health condition of their plants. I think this system will help farmers to monitoring and identify plant diseases as well as help to reduce crops losses ratio.**

***Index Terms*—Plant health, CNN, Deep Learning, Plant Disease**

I. Introduction

Agriculture is necessary for maintaining ecosystems and providing base for human civilization. It is possible only for agriculture to manage food for billions of people. According the previous studies we can see agriculture has multiple problems like climate change, irrigation, diseases, lack of agricultural land and capital and many more. Among them plant disease is the worst obstacle and challenge for agriculture production [1]. For example, only for plant disease, between 20 and 40 percent of agricultural crop production is lost per year in the US [2]. Any wrong prediction about plant disease will result uses of inappropriate fertilisers, which may cause plant stress and nutritional deficiencies in the agricultural sector [3]. So it is very important to identify the accurate disease and uses the correct pesticide and fertilizer on time for a better agricultural production. According with time technological progress give us opportunity to monitoring plant health by using computer vision and deep learning. It is possible by analyzing visual data those methods provide us scalable and accurate solutions. Convolutional Neural Networks is a DL architecture for image classification and object detection which we can use for plant health monitoring [4]. A photo base plant health monitoring can use for identification different types of disease. This method can be effective for both small and large scale agriculture sector to identify different disease and make accurate decision about uses of fertilizers and pesticide on time. This paper presents a comprehensive system for plant health monitoring using photos analyzed through a CNN-based DL model. This method focusses on classifying and detecting diseases using a meticulously selected dataset of 87 thousand RGB images for 38 different disease categories in 14 crops species. I created a web application beside DL model where farmer can upload images and can see the prediction of disease. I also discuss about system design, dataset and implementation. End of this paper, I explained the evolution of my DL model result.

II. Related Work

Due to the developments in deep learning and image processing methods we can see in recent years the interest in plant health monitoring is increasing day by day. Convolutional Neural Networks (CNN) a most useful type of deep learning has been the widely uses in much recent research on automating the identification of plant diseases. Mohanty et al. [5] used a large number of dataset of leaf photos to show how CNNs may be used to classify plant diseases with a remarkable level of accuracy. Similar to this, Ferentinos [6] investigated CNN designs for detecting several illnesses in a range of plants, emphasising CNNs' versatility with regard to different datasets. The authors in [7] identified 27 crop illnesses found in China's difficult mountainous regions using CNN as an underlying DL engine. The user interface was implemented as a Java applet, which enables Chinese farmers to use the system conveniently. According to a series of tests the authors carried out 86.1 percent recognition accuracy. An apple leaf disease detection method based on the Mask Region-based CNN (R-CNN) model was proposed by Jiang et al. [8]. An object instance segmentation DL model called R-CNN can identify items of interest in an image and provide a segmentation mask for each instance. A CNN model for identifying common apple illnesses is trained using a dataset of 2029 photos of sick apple leaves in five different disease classes. The CNN model's classification accuracy was determined to be 78.8

In the final analysis about previous work, the use of computer vision and deep learning for plant disease detection reveals that the majority of these methods concentrate on certain crop species, disease classes or geographical areas. Furthermore, the majority of DL-based models are made to function offline, which makes them unsuitable for crop disease diagnosis in real time [3].

III. System Design

The described plant health monitoring system analyses plant photos and identifies health issues using a Deep Learning (DL) model. We can see the architecture of this system on figure 1. There are total three primary component on the architecture including CNN modeling and training, Dataset and cloud storing and finally User interface layer. On the first layer I describe the CNN base Deep Learning model where I train that with 87k images of 38 different disease among 14 crops. On the second layer I describe about the cloud layer for storage web app and dataset. At the end, on third layer I showed how to use the model on web app by using mobile or laptop for monitoring plant health.

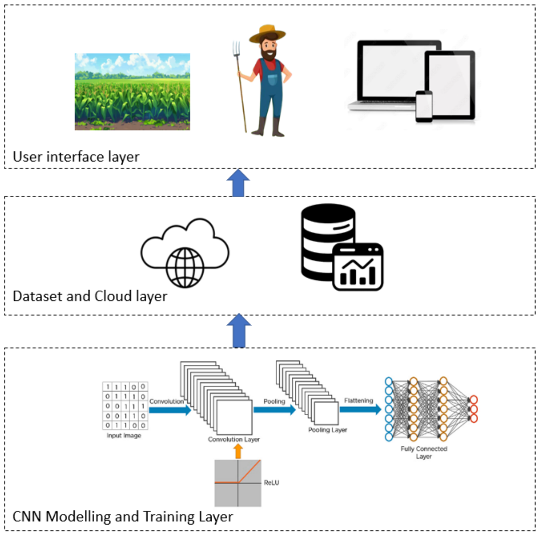
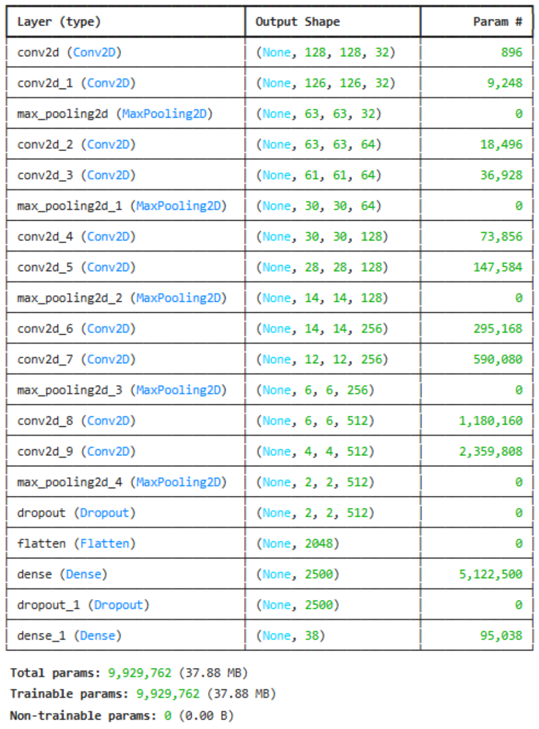


Fig. 1. System design

A. CNN Structure

The Convolutional Neural Network (CNN) designed for plant health monitoring processes input images of size 128x128x3 (height, width, and RGB channels). It features multiple convolutional layers with increasing filters (32, 64, 128, 256, and 512), each utilizing 3x3 kernels and ReLU activation to extract spatial and non-linear features, followed by max pooling layers to reduce spatial dimensions while retaining key features. After the convolutional and pooling layers, the feature maps of size 2x2x512 are flattened into a 1D vector of 2,048 values, which is passed to a fully connected Dense layer with 2,500 neurons. The output layer, consisting of 38 neurons with softmax activation, predicts probabilities for each class (plant disease or healthy category). Dropout regularization 25 percent and 40 percent is applied during training to prevent overfitting. With a total of 9,929,762 trainable parameters, this CNN effectively extracts features from plant images and classifies them into 38 categories, enabling robust plant health monitoring using deep learning.



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)
```

Fig. 2. The structure of CNN

B. Dataset

Although for object detection there are many datasets available and easily possible to collect but for plant health monitoring not much dataset resources available. After some research about that I see few reliable websites provide data for plant health monitoring including Plant Village [9], Kaggle [10] and Google Web Scraper [11]. Among them I did collect data for my plant health monitoring from Kaggle. The dataset includes over 87K RGB photos of both healthy and affected crop leaves with 38 classes of 14 different plant. To maintain the structure of the dataset directory the entire dataset is split into training and validation sets in an 80/20 ratio. To predict the health condition there is a new a new directory called Test built with 33 photos of different plant including both healthy and affected. Before training my CNN model, I had normalise the dataset's leaf picture pixel intensity range. This step was required because feature vectors derived from input photos should have all dimensions within the same intensity range [3]. So that my CNN model converged more quickly during training. Here is the data set link (https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset/data)

IV. Implementation

According my system design there are total two implementation I did for my CNN model. Firstly I did implement CNN implementation and then I did create a web app by using python. So here I individually discuss both implementation.

A. CNN implementation

For implementation purpose I did use several required library. To implement this CNN model I did use Keras development environment 2.4 [21]. Keras is a popular open source neural network library written in Python and it uses TensorFlow 02 [22] environment as a backend engine. Keras usually run over the TensorFlow and provide a easy platform for user to create and test deep learning models using Python. I used tf.keras.utils.image-dataset-from-directory function for loading image from my directory. It is useful for image pre-processing and augmentation. It also helps to avoid overfitting and quickly learning. It is mandatory to keep same size of all input images before feeding them on the model. For train the model I did use 3 channels colored (RGB) images with resize dimension of 128 x 128 pixels. I did set 10 epochs of 2197 images respectably. The training process of this model was done by using Intel CoreTM i7 CPU processor which has 16 GB RAM. The training process time around 5 hours to complete 10 epochs because each epoch took around 30 minutes. Every one hours I did monitor about the progress of training process. My plant health monitoring is a multi-class CNN model where I did use softmax activation function in output layer cross-entropy as the loss function. I also added more Convolutional Layer to extract more feature from images and increased number of the neuron for higher accuracy and avoid over feeding. I did train my model more than five time then I get my desire accuracy and losses rate was become very low. In the figure 3 we can see the accuracy results for training and validation set in individual line graph for per epoch. For the training and validation set we can see the accuracy increased according with the number of epochs in both cases. Initially we can see at the first epoch the training accuracy was only 0.4 while validation accuracy was 0.82. After that for second epoch rapidly training accuracy was increased and reach at 0.84 beside that the validation accuracy also slightly increased to 0.90. From the third to ten epoch, training and validation both increased continually and the final training accuracy was 0.98 while validation accuracy was 0.97. That is really a great result for my plant health monitoring model.

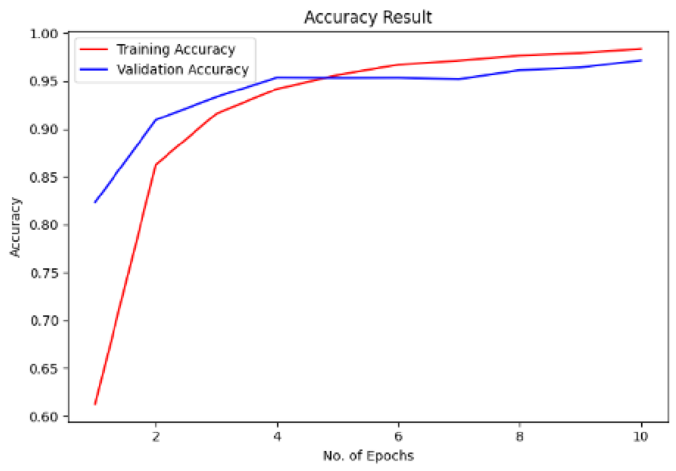


Fig. 3. Accuracy Results

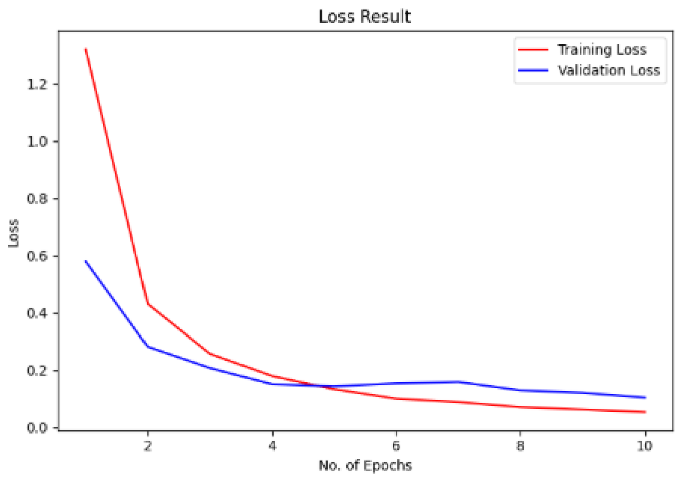
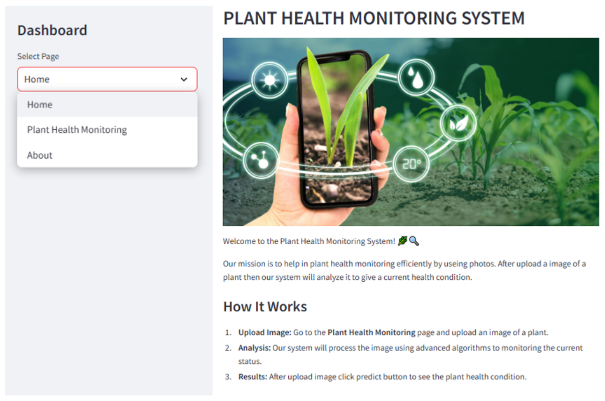


Fig. 4. Loss Results

In the figure 4 we can see Training and validation set loss results in individual line graph for per epoch. For the training and validation set we can see the loss decreased according with number of the epochs in both cases. Initially we can see at the first epoch the training loss was 2.13 while validation loss was 0.58 which is the maximum value in both cases. After that for second epoch rapidly training loss was decreased and reach at 0.5 beside that the validation loss also decreased to 0.28. From the third to ten epoch, training and validation both loss decreased continually and the final training loss was only 0.058 while validation loss was only 0.1 which is really very less in both cases.

B. Web App

I did build a web application with Streamlit to offer an user friendly interface for plant health monitoring. Streamlit is an open-source Python framework for deep learning and data science projects that makes it easier to create dynamic and accessible online apps [12]. Users can upload images of plants to the web app and the trained deep learning model uses those images to predict the health status of the plant. Streamlit is perfect for non-technical users because it is very easy to use and it automates result visualisation as well as it allows for real-time interaction. There are total three page of my web app including the Home, Plant Health Monitoring and About project. For get plant health status the user needs to go to the Plant Health Monitoring page and then need to upload desire photo after that needs to click on Check button. Finally within second the user can see the current status of the plant.



PLANT HEALTH MONITORING SYSTEM

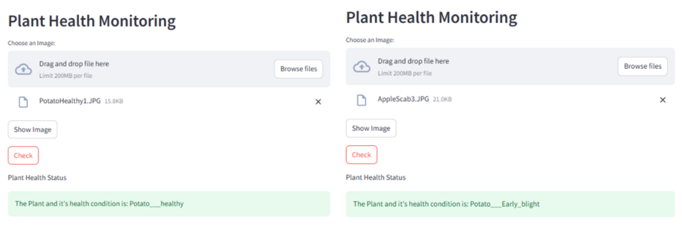
Welcome to the Plant Health Monitoring System! 🌿🔍

Our mission is to help in plant health monitoring efficiently by useing 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 monitoring the current status.
3. **Results:** After upload image click predict button to see the plant health condition.

Fig. 5. Web App user interface



The Plant and it's health condition is: Potato___healthy

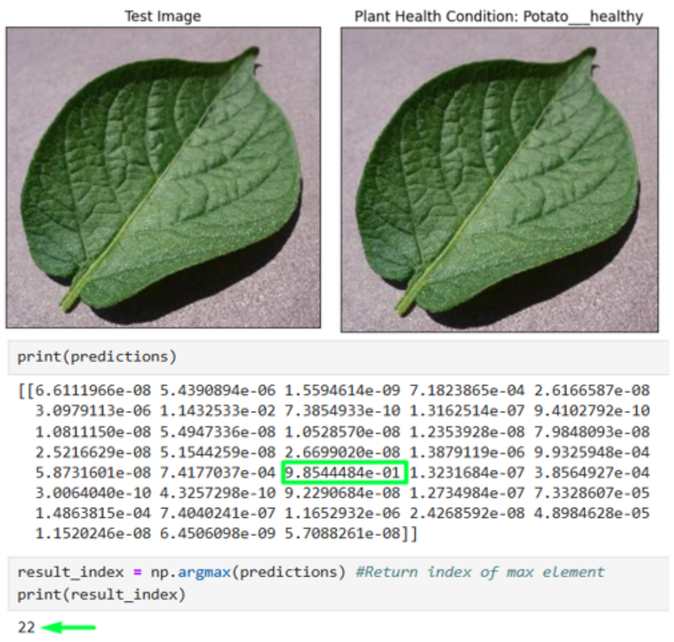
The Plant and it's health condition is: Potato___Early_blight

Fig. 6. Web App plant health Monitoring results

V. Experimental Evaluation

To evaluate the performance of my trained CNN model, I used several metrics, including accuracy, precision, recall, and F1-score. A dataset of more than 87k plant photos was used to train the model and then I stored that for use later. The saved model was loaded and predictions were made on the test set during the testing phase. After feeding image from test set we can see a good prediction of plant health. I used Softmax activation vector on outer layer that's why it converts the model's raw outputs into probabilities and helps in multi-class classification by identifying the class with the highest probability. Here is some example of my test and prediction output. In the Figure 7 we can see, I test Corn plant and the model is more than 99 percent confident that the plant is affected common rust virus while on Figure 8 we can see the model more than 98 percent sure that Potato is healthy.

For more closely evaluate the model I also observe precision value which measures the accuracy of positive predictions. Beside that I also check Recall metrics which measures the



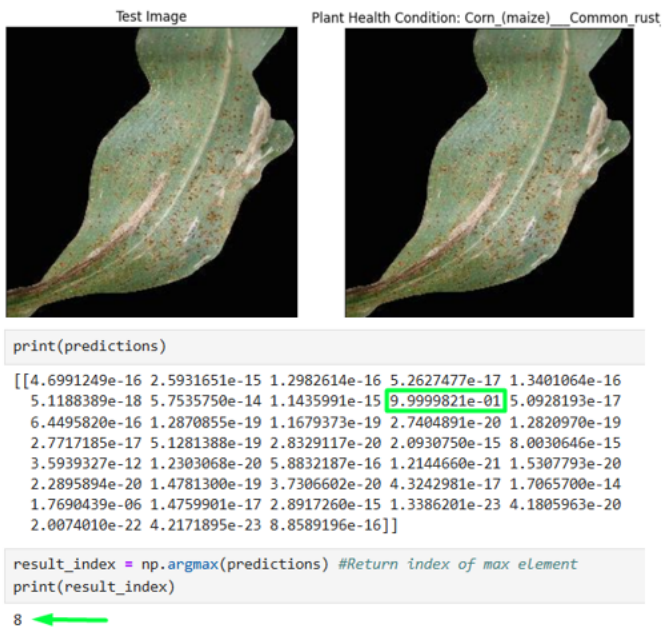
```
print(predictions)
```

```
[[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
```

Fig. 7. Model Prediction Result for Potato



```
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]]
```

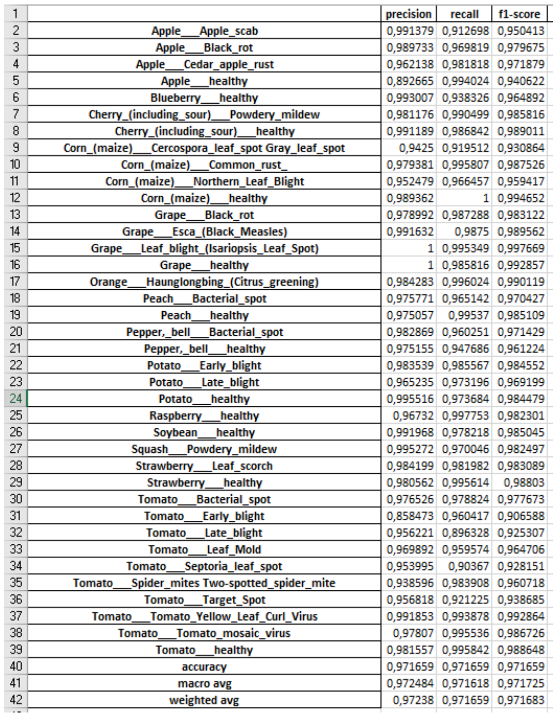
```
result_index = np.argmax(predictions) #Return index of max element
print(result_index)
```

```
8
```

Fig. 8. Model Prediction Result for Corn

model's ability to correctly identify all relevant true positives out of all actual positives. Finally for making balances between precision and recall I use F1-Score which offering a comprehensive measure of the model's overall performance. Figure 9 contains the detailed values of precision, recall, and F1-score for each class. These results demonstrate the model's effectiveness in classifying plant health conditions and its potential for real-world applications in agriculture.

I generated a confusion matrix in order to assess the



1		precision	recall	f1-score
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

Fig. 9. The Precision vs Recall values

CNN model's performance across different plant health and disease classes. A comprehensive representation of the model's predictions is given by the confusion matrix on Figure 10 which shows how many true positives (right classifications) and false positives (false negatives) each class received.

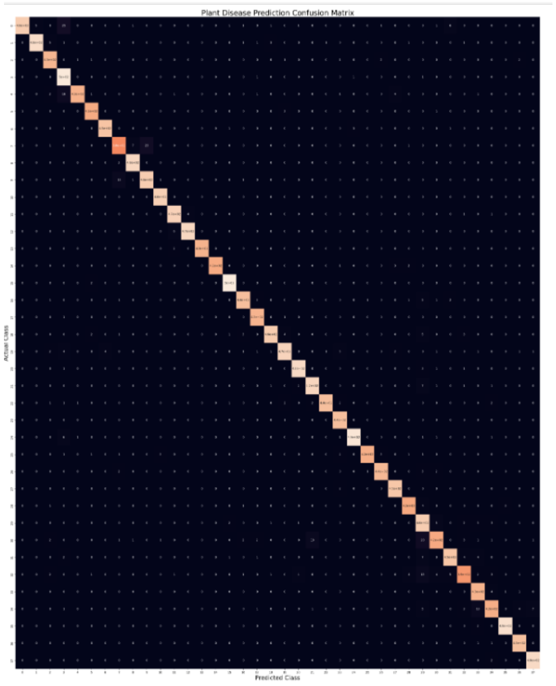


Fig. 10. The Confusion matrix for CNN model

VI. Conclusion and Future Work

We should develop our agricultural sector with the help of modern technology like Deep learning and AI for more production and reduce production cost. In this work I used picture data to construct a CNN-based system for monitoring plant health. Metrics including precision, recall and F1-score verified the model's excellent performance in differentiating between healthy and unhealthy plant conditions. The integration in Streamlit-based web application enhanced accessibility for real-world applications and make it more user friendly for plant health monitoring with second.

In future I will focus on expanding the dataset to include more plant species and diseases to improving model generalisation and enhanced in a variety of environmental settings so the system will be optimised for real-time implementation in agricultural processes.

References

[1] Krishna, Ujjwal Bharadwaj, Sreepada Kaswan, Vinit Kumar, Anuraj Kaur, Gursimran Rana, Pooja. (2024). Agri Watch: Precision Plant Health Monitoring using Deep Learning. E3S Web of Conferences. 556. 01028. 10.1051/e3sconf/202455601028.

[2] Crop Protection Network. 2021. Available online: https://cropprotectionnetwork.org/ (accessed on 25 June 2021).

[3] Ahmed, A.A.; Reddy, G.H. A Mobile-Based System for Detecting Plant Leaf Diseases Using Deep Learning. AgriEngineering 2021, 3, 478–493. https://10.3390/ agriengineering3030032

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[5] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," Frontiers in Plant Science, vol. 7, p. 1419, 2016.

[6] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," Computers and Electronics in Agriculture, vol. 145, pp. 311–318, 2018.

[7] Chen, W.; Lin, Y.; Ng, F.; Liu, C.; Lin, Y. Ricetalk: Rice blast detection using internet of things and artificial intelligence technologies. IEEE Internet Things J. 2020, 7, 1001–1010.

[8] Jiang, P.; Chen, Y.; Liu, B.; He, D.; Liang, C. Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. IEEE Access 2019, 7, 69–80.

[9] Hughes, D.P.; Salathe, M. An open access repository of images on plant health to enable the development of mobile disease diagnostics. Comput. Soc. 2016, 11, 1–13.

[10] Kaggle: Machine Learning and Data Science Community. 2021. Available online: https://www.kaggle.com/ (accessed on 25 June 2021).

[11] Google Web Scraper. 2021. Available online: https://chrome.google.com/webstore/detail/web-scraper/jnhgnonknehpejjnehehllk liplmbmhn?hl=en (accessed on 25 June 2021).

[12] Streamlit, "Streamlit: The fastest way to build and share data apps," Streamlit Inc., 2024. [Online]. Available: https://streamlit.io. [Accessed: Dec. 8, 2024].