Plant Health monitoring with Sensors

Nnaemeka Valentine Eze
Electronic Engineering Department
Hochschule Hamm-Lippstadt
Lippstadt, Germany
Nnaemeka-valentine.eze@stud.hshl.de

***Abstract*—The vitality of plant and animal life is essential to human nutrition, highlighting the ecosystem's interdependence. In turn, plants need nutrients from the soil and surrounding environment to survive. This research explores the critical role that plant health plays in maintaining life, and it suggests a novel way to monitor and improve plant health by integrating sensors and Deep Learning.**

The use of deep learning (DL) and sensors in plant health monitoring is a paradigm change in environmental and agricultural practices. By utilizing cutting-edge technology, deep learning algorithms analyze complex data collected by sensors to provide precise and quick plant health assessments. When completely implemented, this methodology has the power to transform farming practices in a way that promotes production and sustainability.

When it comes to monitoring plant health, the combination of sensors and deep learning offers an effective tool that enhances agricultural operations, cut down on resource waste, and support sustainable food production. In line with the Sustainable Development Goals (SDGs), this revolutionary method has the potential to completely change plant care in many ways.

One of the main advantages of this integrated system is the early detection of stress and sickness, which allows for quick action to reduce loss and damage. Targeted interventions are made easier by timely and relevant information, which maximizes the use of resources like pesticides, fertilizers, and water. This method gives farmers the ability to modify illumination, irrigation, and dietary needs, which improves crop quality and yields.

In addition to its immediate benefits, this system minimizes chemical usage through efficient monitoring, thereby addressing global environmental problems. Deep learning and sensor integration promotes further technological and agricultural progress, resulting in the creation of novel concepts and tools to further enhance plant health.

This project aims to provide a comprehensive and efficient method for assessing the health of plants in agricultural and environmental settings. It accomplishes this by fusing cutting-edge neural network technology with complex algorithms. The study also aims to further the conversation on using technology such as Deep Learning, to maximize crop yield and environmental responsibility.

***Index Terms*—Precision farming, plant phenotyping, smart Agriculture, Neural network, Algorithm**

I. Introduction

Human beings depend on plant and animals primarily for food, just as plant depends on soil and environmental nutrients to be alive. By derived meaning, overall life generally depends on plant. And for a plant to be alive, there must be need to prioritise its health. Plant health monitoring with sensors and machine learning(ML) is a novel method of analyzing plant health in agricultural and environmental settings for overall safety of man and animals. It leverages the power of cutting-edge technology using deep learning algorithms, to interpret the detailed data that is gathered by a variety of sensors. As a result, an accurate and timely assessment of the health and welfare of plants could be reliably made [13]. This innovative technology has the potential to fundamentally enhance further how we think about and approach plant health, when fully utilized. Hence, farming techniques that are more sustainable and productive.

When sensors and deep learning(DL) work together to monitor plant health; it could provide an efficient tool that can improve farming practices, cut down on resource waste, and eventually contribute to the efficient and sustainable production of crops. This approach has the capacity to fundamentally alter how we tend to and care for plants in many different contexts, for commercial farmers and peasants in accordance with the SDG (sustainable development goal) vision.

Deep learning and sensor-based plant health monitoring are important for a number of principal reasons: Early Detection of Stress and Disease in plant promotes prompt intervention, thereby minimizing damage and loss. With detailed, real-time information about plant health known, farmers can implement targeted interventions capable of enhancing efficient use of resources like water, fertilizers, and pesticides, reducing waste and environmental impact. Farmers can increase crop yields and quality by altering irrigation, illumination, and nutritional requirements in response to real-time data simply by adequate monitoring. In the face of growing global environmental concerns, it is essential to minimize the use of chemicals, this can be achieved by effective monitoring. Finally, Deep learning and sensor integration in plant health monitoring encourages continued study and advancement in the fields of technology and agriculture. It promotes the creation of new ideas and technology to improve plant health even more.

Farmers used to keep an eye on the health of their plants by just observing them. Since the eye can only identify the condition after it has begun to harm plants, it has been noted that most of the time this is caused by late detection. Even among specialists, the accuracy of detection and diagnosis based only on visual observation is compromised by subjective impressions [5]. Therefore, using these methods in the face of mounting concerns about plant health is counterproductive.

This work aims at investigating how deep learning methods and sensors can be combined to effectively monitor plant health. This study tends to develop a thorough and effective approach for evaluating the health of plants in agricultural and environmental settings, as well as to evaluate the efficacy of this approach, identify potential applications, and assess its impact on agricultural practices and sustainability. It does this by combining sophisticated algorithms with advanced technology on neural networks. The study also aims to further the conversation on using technology to maximize crop yield and environmental responsibility.

II. Related Work

When we talk about plant health, we often limit it to plant diseases and pest. However, these are tiny part of what makes a plant unhealthy, and as such the resultant effect of health conditions. It is a common knowledge that illnesses, particularly those that affect plants, always have a window of infection during which they typically exhibit chemical or biological symptoms. [2] At what stage these changes were observed makes a lot of difference in the disease severity estimation. The term ”infection window” describes a certain stage of a plant's growth when it is vulnerable to infection by a given pathogen. Several factors, such as the plant's maturation stage and the surrounding environmental conditions such as temperature, humidity, nutrients, light intensity affect this window. Hence the need to learn these pattern, in other to be able to observe the changes. Observable morphological changes in the plant, such as wilting, discolouration, lesions, or the appearance of aberrant growths, might be considered biological signals. Chemical indicators could be changes in biochemical markers or signaling molecules released by the plant in reaction to a disease invasion. For instance, alterations in the concentrations of particular metabolites or the initiation of compounds linked to defense.

Therefore, plant health monitoring involves observing the chances of a disease, the changes that may suggest unfavorable plant condition with a view to eliminate them or reduce the level of damage. So, it is important to employ a mechanism capable of getting these biological, environmental or chemical indicators.

In this chapter, we aim to examine several studies that identify and forecast these changes, along with the sensor technologies that are employed.

It is crucial to note that various plant diseases manifest at different phases and target distinct plant organs. There may also be differences in the methods used to find them. Plant diseases that manifest as symptoms on leaves have the most distinctive characteristic for identification out of all the diseases. [2] The most common of them is the photo-based approach using imaging devices like cameras. The prevelence of sensors became necessary for its precision.

One of such is the work recently carried out by ”Sefali Acharya of College of Life Sciences and Agriculture, University of New Hampshire, Durham, NH, United States” in his paper ”Plant monitoring using nanosensors. [1] Analytical tools called nano-biosensors were created especially to track the health of plants. They function by identifying modifications in the pathways and metabolism of plants without interfering with their inherent activities. Throughout his work, he emphasized that the information obtained can also be used to evaluate the extent of the plant's harm or overall welfare phytohormones and microRNAs.

For a variety of uses, such as medical diagnostics and environmental monitoring, numerous biosensors have been developed for use. A sensor based on optical, magnetic, chemical, electrochemical, vibrational, or optical signals may be used to detect the analytes, depending on the sensor's working principle. Biorecognition components like DNA, antibodies, and enzymes can be used to increase specificity and increase the limit of detection by acting as transducers made of nanomaterial matrices giving rise to different types of biosensors. Affinity biosensors are created with extremely specialized biorecognition components, such as DNA and/or antibodies. Others may include; Antibody-based biosensors, Nucleic acid-based affinity biosensors, Enzymatic-based biosensors and bacteriophage-based biosensors.

Broadly speaking, biosensors based on nanotechnology, also known as nano-biosensors, provide an advanced way to track the health of plants. Their capacity to transform biological impulses into electrical signals, along with their categorization according to transducer kinds and dependence on particular biorecognition components, renders them invaluable instruments for the instantaneous assessment of plant stress and comprehensive health management in the agricultural sector.

As earlier highlighted, plant health can easily be identified on their leaves. The use of image for analysis has become the de-facto approach for obtaining data for training deep learning model and severity estimation. This is because of availability of dataset needed for this. Then evolution of picture-based has transitioned from ordinary image capturing devices like camera, to a precision sensor devices. One advantage of sensors is that it can be helpful in detecting signs when they are yet to be picked by conventional picture-base devices.

Some biotic and abiotic factors could lead to stress on the plant. Plant stress could be one parameter to monitor plant health, and can best be captured using various sensors. Plant stress could be in varied form; water stress, environmental stress [9], mechanical stress. [**?**] In previous work, multispectral sensors has been one of the imaging sensors used to track the health of plants. Stress makes plants more visible, which makes it easier for multispectral sensors to identify and detect high stress levels in plants across a variety of wavelengths, including red, green, red-edge, and near-infrared. [4]
In this review work done by ”Zhang, L.; Zhang, H.; Niu, Y.; Han, W.”, these sensors were used in Mapping maize water stress based on UAV multispectral remote sensing. One of the first applications of deep learning to the analysis of plant stress photos was image classification. To achieve this, one or more photos are often utilized as input data in crop stress image classification, and an output is used to make a diagnosis. These output could be healthy or unhealthy plant. In contrast to computer vision, where thousands or millions of samples

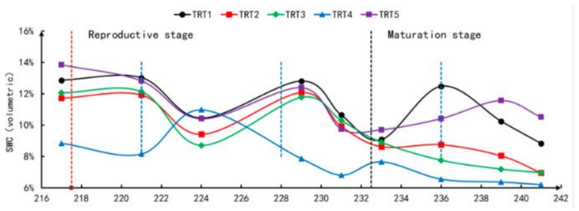


Fig. 1. Changing curve of the average SWC at varied depths [16]

are available, each diagnosis in this scenario is a sample, and the dataset size is typically less. Thus, researchers ought to find the transfer learning useful for these kinds of applications. [4] In order to determine the suitability of the data for mapping the water stress status of maize under various levels of deficit irrigation at the late vegetative, reproductive, and maturation growth stages, the researchers employed high-resolution multispectral images taken from an unmanned aerial vehicle (UAV). The UAV multispectral imagery yielded nine vegetation indices (VIs) associated with crop water stress, which were then utilized to create CWSI inversion models. This was carefully observed and reported. The findings indicated that significant differences in non-water-stressed baselines were observed during the reproductive and maturation stages. The study also identified specific vegetation indices, such as TCARI/RDVI and TCARI/SAVI, that exhibited the best correlations with CWSI at different growth stages. Good correlations with CWSI values were shown by regression models based on TCARI/RDVI and TCARI/SAVI. The R2 results demonstrated how strongly the chosen vegetation indices and CWSI were correlated. The CWSI values calculated by on-site measurements when compared with those retrieved by VI-CWSI regression models suggested that CWSI values from the regression models had better abilities to assess the field variability of both crops and soil. The study shows that mapping maize water stress at various growth stages may be accomplished with high-resolution UAV multispectral ensors.

Although multispectral sensors offer benefits in terms of affordability, simplicity, and readability, they might not be able to accurately detect and discriminate between various illnesses in various crops. One explanation for this might be the small number of bands. [2] Hyperspectral sensors, which can detect more bands, are better suited to reduce that problem and may be more helpful in differentiating between various illnesses.

The use of classic machine learning solutions is restricted because of the requirements for feature extraction and definition, despite the fact that these techniques might accurately identify diseases. Consequently, because Deep learning models enable automatic feature extraction, deep learning can aid in enhancing generalization.

III. Deep Learning

Deep Learning is a subset of machine learning, which is a subset of Artificial Intelligence . It is about using different convolutions to create ”deeper” neural networks that give a hierarchical representation of the data which enables greater learning capacities, which in turn enable improved accuracy and performance. [6] Simply put, artificial intelligence (AI) is the broad field that includes the creation of intelligent machines. A branch of artificial intelligence called machine learning (AI) is dedicated to creating systems that can learn from data. Deep learning, on the other hand, is a subset of machine learning that focuses on using deep neural networks to model and solve complicated problems. Within the more general topic of artificial intelligence, each level in this hierarchy denotes a more focused and specialized area of study.

About 20 years ago, interest started to grow in machine learning applications for tracking plant health and diagnosing diseases. [2]
Conventional machine learning methods such as Naïve Bayes, Random Forests, K-Nearest Neighbors (KNN), and Support Vector Machines (SVM) were frequently utilized. Traditional machine learning methods, however, have certain drawbacks that make them less practical. In order to train a model, traditional machine learning algorithms needed to extract significant features, which was a laborious procedure. [14]
Furthermore, conventional image processing and machine learning had limitations, and success was frequently restricted to particular circumstances. [11]
Thus, there has been a trend in the past ten years toward the use of deep learning approaches. [2] The automatic feature extraction that deep learning conducts by default removes the need for human feature engineering. Higher accuracy has since been achieved with the use of deep learning techniques in contrast to conventional methods. Convolutional Neural Network(CNN) is the most widely used deep learning framework. Another framework that has gained application is the Recurrent Neural Network(RNN).

A. *Convolutional Neural Network(CNN)*

CNNs are a particular class of neural network architecture that are intended for the processing and analysis of visual input, including pictures and videos. Compared with standard fully connected neural networks, the architecture of a CNN is more complicated. Usually, it is composed of a number of layers, such as input layer, fully connected, pooling, convolutional layers and output layer. [8]

Convolution is a mathematical process that creates feature maps by combining input data with a kernel or filter. Convolutional layers in CNNs employ a variety of filters to look for patterns in the input data (picture), such as edges, textures, or higher-level features. The network may automatically learn hierarchical representations of the input by to this mechanism. CNNs' capacity to automatically learn structural characteristics is one of their advantages. [8] This is because while higher layers in the network often capture more complicated and abstract elements, the lower layers typically catch simple features like edges and textures. For applications like picture recognition, where patterns at many levels of abstraction help recognize objects, hierarchical feature learning is essential. In a typical CNN platform, features are extracted

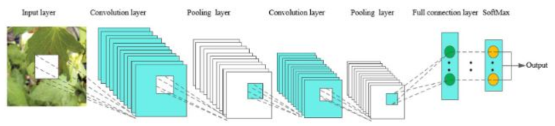


Fig. 2. Structure of CNN
[] [8]

by convolutional layers, spatial dimensions are decreased by pooling layers, while the final classification or regression is carried out by fully connected layers. Fig below shows the basic Architecture of a Convolutional Neural Network.

1) —Input Layer: Each neuron in the input layer, which is in charge of receiving and representing the raw input data, is associated with a particular pixel in the input image for the purposes of computer vision. In the forward-pass phase of the neural network, it serves as the beginning point for information flow. After then, the network's later layers will gradually learn hierarchical representations of the input data, which will bring the network to its final output layer, which is where the predictions or classifications are made.

2) —Convolutional Layer: Convolutional operations with filters or kernels are applied to the input data at the Convolutional Layer, a crucial part of CNNs. This layer is very good at capturing elements such as textures, edges, and intricate patterns. It learns several features simultaneously by using numerous filters. The network learns to extract hierarchical features from the input data and can comprehend and depict complex patterns as it moves through successive convolutional layers. The convolution core slides on the feature map during data processing in order to extract a portion of the feature information. This results in a pixel matrix that bears the manipulated or multiplied object pixel. Then, the neurons are fed into the pooling layer to retrieve the feature once more following the convolution layer's feature extraction. [8]

3) —Pooling Layer: The main goal of the pooling layer is to reduce the width and height of the input volume by using the input feature maps that were acquired from the previous Convolutional Layer. This helps to improve efficiency and control computing complexity. Typical pooling techniques consist of: Max Pooling; where the maximum value is kept and the remaining values are deleted, within a predetermined zone that is usually a 2x2 window. It works well to keep the most significant features intact when using max pooling. [10]

Average pooling; by calculating the mean value across a given area, thereby producing smoothed representation of the input. [15]

Additionally, pooling facilitates the achievement of certain translation invariances. That is, the pooled representation stays mostly unaltered even if a feature or object in the input is moved significantly. Overfitting, the situation where a model performs well on training data but is unable to generalize to new, unknown data, is another issue that pooling helps to control. By imposing a kind of spatial hierarchy and lowering the spatial dimensions, pooling helps to prevent overfitting by lowering the number of parameters in the layers that follow. By lowering the quantity of data that must be processed while keeping the crucial elements, pooling improves overall computational efficiency.

4) —Fully connected Layer: Particularly in the last phases, the Fully Connected Layer is an essential part of neural network topology. Every neuron is linked to every activation in the layer before it. The activation in the previous layer and all of the neurons in the present layer are therefore densely or fully connected. Flattening the high-level filtered input acquired from the preceding layers is the purpose of the layer. For example, the early layers of convolutional neural networks (CNNs) learn hierarchical elements including textures, edges, and intricate patterns. These high-level properties are transformed into a vector form via the Fully Connected Layer. The fully connected layer in a network fills the space between the final output layer and the convolutional or recurrent layers.

B. Recurrent Neural Network(RNN)

Recurrent Neural Network (RNN) is a specialized type of artificial neural network capable of handling sequential data. It does so by maintaining a hidden state or memory of previous inputs. No doubt why this architecture are effective for a range of tasks where understanding the temporal dependencies in the data is essential. A network with the same set of weights reproduced for every time step is what an RNN is. The network can handle sequences of any length because to this stretching over time.

We may discuss some few features of RNN, to understand its uniqueness. This may include;

1) Long Short-Term Memory(LSTM): In LSTMs Information flow is regulated by gating mechanisms , which have a more intricate structure. They can learn from sequences with long-range dependencies because they have a memory cell that has the capacity to store and retrieve information for extended periods of time.

By adding gating mechanisms, LSTMs overcome the drawbacks of conventional RNNs and enable them to recognize and retain long-range dependencies in sequential data. This makes them especially helpful for tasks involving time-series or sequential data.

2) Gate Recurrent Unit(GRU: Gated Recurrent Unit (GRU) is another form of recurrent neural network (RNN) architecture that is used in the field of deep learning for its own purpose. Long short-term memory (LSTM) networks and gate recurrent units are comparable, although the former has fewer parameters and is typically easier and faster to train. The GRU is notable for lacking an output gate and only having two gates: an update gate and a reset gate. [3]

IV. Sensor based Plant Health Monitoring

The capacity of plants to develop and yield successfully in the face of environmental stressors, pests, and competition is referred to as plant health. Water stress for instance can affect many physiological and biochemical processes in plants, including photosynthetic capability, reactive oxygen species

(ROS) [12], stomatal response, and metabolic modification, etc. These impacts can have a substantial impact on the health of plants. Also Plants' metabolic processes alter in response to drought stress, which has an impact on both the plants' growth and resilience.

In tomato plant for instance, the root development and nutrient uptake can be impacted by water stress. Additionally, it might cause notable drops in chlorophyll content, which would impair the plant's capacity for effective photosynthesis. This is capable of affecting the overall health of the plant, hence the need for monitoring. These stress when combined together, brings about the unhealthy plant characterised by pest and diseases. But these features differs to a certain degree from plant to plant. For the purpose of a widal scope, we would consider the case of a collection of plants to analyse the processes involved in plant health monitoring using deep learning by classification. These data were made to undergo augumentation, indexing as well as nomalization, before feeding them into the model.

A. Data Collection

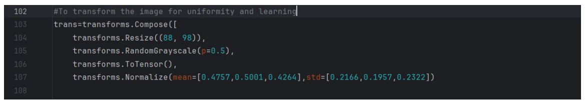
A dataset in deep learning is an organized group of data that is typically connected to a single body of work. It can either be used for training, or testing and validation. Sensor datasets can be obtained from various sources, such as public repositories like Kaggle, Plant Vilage, COCO [7], plantDoc [**?**] etc or by collecting data by the individual using imaging sensors. The dataset is grouped into healthy and unhealthy, with the purpose of training the model to identify and classify an unhealthy from healthy ones.

1) Dataset: In this work, we obtained dataset from Kaggle database which was slightly engineered by Nasser and etal. The data consist of about 87K RGB images of healthy and some diseased plants of different varieties sourced from different places. It is made up of 56236 images. For smooth running on my PC we decided to use ten percent of these images for training using the (reduced-dataset-size) function. We equally used the generally accepted 80:20 ratio for training and validation respectively.

B. Data Preprocessing

It is crucial to pre-process the data to make sure it is in the right format before training a model. These may include such activities as; data cleaning, normalization, augumentation, transformation, splitting. In the figure; 3 shows the preprocessing pipeline. First, the image is changed by scaling the input image to 88 pixels in height and 98 pixels in width. This standardizes the image size so that it will be consistent throughout training.

After that, transformation applies a probability of 0.5 to the image's random conversion to grayscale. It periodically eliminates color information, adding diversity to the training set. Before normalizing the picture elements, the image is further transformed from a PIL Image or NumPy array to a PyTorch tensor because PyTorch is being used as described below.



```
#To transform the image for uniformity and learning
trans=transforms.Compose([
    transforms.Resize((88, 98)),
    transforms.RandomGrayscale(p=0.5),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.4757,0.5001,0.4264],std=[0.2166,0.1957,0.2322])
```

Fig. 3. Data Preprocessing

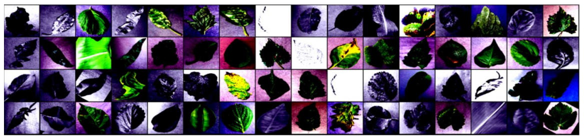


Fig. 4. Normalized images

```
    trans=transforms.Compose([
    transforms.Resize((98, 98)),
    transforms.RandomGrayscale(p=0.5),
    transforms.ToTensor(),
    transforms.Normalize(mean=
    [0.4757,0.5001,0.4264],std=
    [0.2166,0.1957,0.2322])

])
```

Lastly, the image's pixel values were standardized through the process of normalization, which scales the pixel values for each RGB channel by applying the given mean and standard deviation values. In training, this facilitates a faster convergence of the model. These processes also ensures that the model tries so hard to learn, in otherwards avoiding overfitting.

C. Deep Learning techniques and algorithm

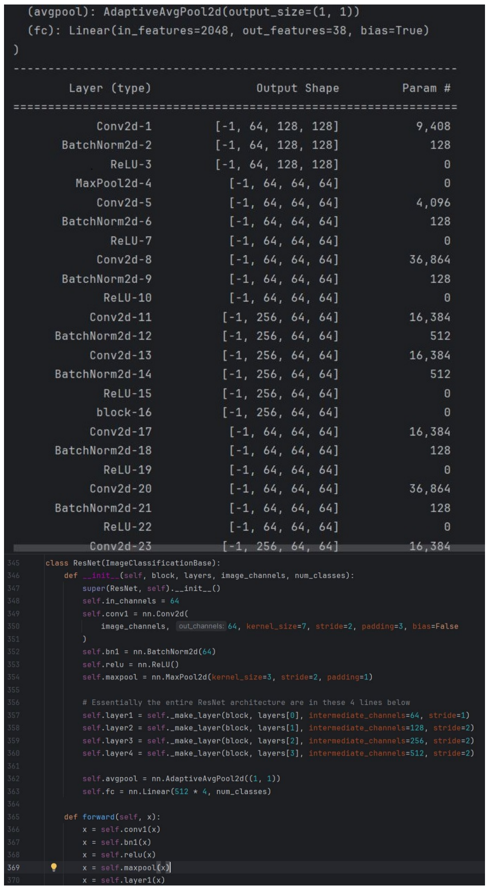
$ResNet50$

ResNet50 is a popular convolutional neural network architecture for image classification applications. Additionally, in computer vision it can be applied in image segmentation and object detection. It has a unique capability of using residual connections by bypassing some layers some times. This informs the name, ResNet(Residual Network). There are other variations of ResNet like ResNet152, ResNet101, ResNet34. The model summary is shown in the figure.5

The choice of ResNet50 for this project is based on proven efficiency in using large-scale image dataset. The major challenge encountered was high memory usage due to its deep architecture.

```
    class ResNet(ImageClassificationBase):
    def __init__(self, block, layers,
    image_channels, num_classes):
        super(ResNet, self).__init__()
        self.in_channels = 64
        # Example convolutional layer

        #self.conv1 = nn.Conv2d
        (in_channels,
```



```
  (avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
  (fc): Linear(in_features=2048, out_features=38, bias=True)
)
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1        [-1, 64, 128, 128]           9,408
       BatchNorm2d-2        [-1, 64, 128, 128]             128
              ReLU-3        [-1, 64, 128, 128]               0
         MaxPool2d-4          [-1, 64, 64, 64]               0
            Conv2d-5          [-1, 64, 64, 64]           4,096
       BatchNorm2d-6          [-1, 64, 64, 64]             128
              ReLU-7          [-1, 64, 64, 64]               0
            Conv2d-8          [-1, 64, 64, 64]          36,864
       BatchNorm2d-9          [-1, 64, 64, 64]             128
             ReLU-10          [-1, 64, 64, 64]               0
           Conv2d-11         [-1, 256, 64, 64]          16,384
      BatchNorm2d-12         [-1, 256, 64, 64]             512
           Conv2d-13         [-1, 256, 64, 64]          16,384
      BatchNorm2d-14         [-1, 256, 64, 64]             512
             ReLU-15         [-1, 256, 64, 64]               0
            block-16         [-1, 256, 64, 64]               0
           Conv2d-17          [-1, 64, 64, 64]          16,384
      BatchNorm2d-18          [-1, 64, 64, 64]             128
             ReLU-19          [-1, 64, 64, 64]               0
           Conv2d-20          [-1, 64, 64, 64]          36,864
      BatchNorm2d-21          [-1, 64, 64, 64]             128
             ReLU-22          [-1, 64, 64, 64]               0
           Conv2d-23         [-1, 256, 64, 64]          16,384
```

```
class ResNet(ImageClassificationBase):
    def __init__(self, block, layers, image_channels, num_classes):
        super(ResNet, self).__init__()
        self.in_channels = 64
        self.conv1 = nn.Conv2d(
            image_channels, out_channels: 64, kernel_size=7, stride=2, padding=3, bias=False
        )
        self.bn1 = nn.BatchNorm2d(64)
        self.relu = nn.ReLU()
        self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)

        # Essentially the entire ResNet architecture are in these 4 lines below
        self.layer1 = self._make_layer(block, layers[0], intermediate_channels=64, stride=1)
        self.layer2 = self._make_layer(block, layers[1], intermediate_channels=128, stride=2)
        self.layer3 = self._make_layer(block, layers[2], intermediate_channels=256, stride=2)
        self.layer4 = self._make_layer(block, layers[3], intermediate_channels=512, stride=2)

        self.avgpool = nn.AdaptiveAvgPool2d((1, 1))
        self.fc = nn.Linear(512 * 4, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.maxpool(x)
        x = self.layer1(x)
```

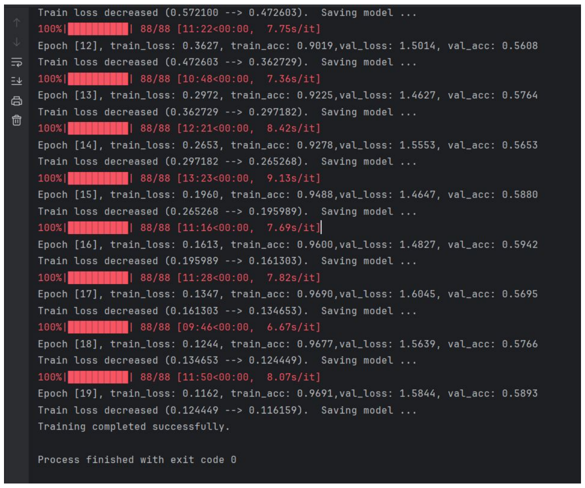
Fig. 5. Model Class and Summary

```
out_channels, kernel_size, stride,
padding)

self.conv1 = nn.Conv2d
(image_channels,
64, kernel_size=7, stride=2,
padding=3, bias=False)

self.bn1 = nn.BatchNorm2d(64)
self.relu = nn.ReLU()
self.maxpool = nn.MaxPool2d
(kernel_size=3, stride=2,
padding=1)
```

```
Predicted: unhealthy, True: unhealthy
Predicted: healthy, True: unhealthy
Predicted: unhealthy, True: unhealthy
Predicted: unhealthy, True: unhealthy
Predicted: healthy, True: unhealthy
Predicted: healthy, True: unhealthy
Predicted: healthy, True: unhealthy
Predicted: unhealthy, True: unhealthy
```



```
Train loss decreased (0.572100 --> 0.472603).  Saving model ...
100%|██████████| 88/88 [11:22<00:00,  7.75s/it]
Epoch [12], train_loss: 0.3627, train_acc: 0.9019,val_loss: 1.5014, val_acc: 0.5608
Train loss decreased (0.472603 --> 0.362729).  Saving model ...
100%|██████████| 88/88 [10:48<00:00,  7.36s/it]
Epoch [13], train_loss: 0.2972, train_acc: 0.9225,val_loss: 1.4627, val_acc: 0.5764
Train loss decreased (0.362729 --> 0.297182).  Saving model ...
100%|██████████| 88/88 [12:21<00:00,  8.42s/it]
Epoch [14], train_loss: 0.2653, train_acc: 0.9278,val_loss: 1.5553, val_acc: 0.5653
Train loss decreased (0.297182 --> 0.265268).  Saving model ...
100%|██████████| 88/88 [13:23<00:00,  9.13s/it]
Epoch [15], train_loss: 0.1960, train_acc: 0.9488,val_loss: 1.4647, val_acc: 0.5880
Train loss decreased (0.265268 --> 0.195989).  Saving model ...
100%|██████████| 88/88 [11:16<00:00,  7.69s/it]
Epoch [16], train_loss: 0.1613, train_acc: 0.9600,val_loss: 1.4827, val_acc: 0.5942
Train loss decreased (0.195989 --> 0.161303).  Saving model ...
100%|██████████| 88/88 [11:28<00:00,  7.82s/it]
Epoch [17], train_loss: 0.1347, train_acc: 0.9690,val_loss: 1.6045, val_acc: 0.5695
Train loss decreased (0.161303 --> 0.134653).  Saving model ...
100%|██████████| 88/88 [09:46<00:00,  6.67s/it]
Epoch [18], train_loss: 0.1244, train_acc: 0.9677,val_loss: 1.5639, val_acc: 0.5766
Train loss decreased (0.134653 --> 0.124449).  Saving model ...
100%|██████████| 88/88 [11:50<00:00,  8.07s/it]
Epoch [19], train_loss: 0.1162, train_acc: 0.9691,val_loss: 1.5844, val_acc: 0.5893
Train loss decreased (0.124449 --> 0.116159).  Saving model ...
Training completed successfully.

Process finished with exit code 0
```

Fig. 6. Metrics

```
Predicted: unhealthy, True: unhealthy
Predicted: healthy, True: unhealthy
```

V. Result and Discussion

After the model it successfully trained for the purpose intended, it is made to undergo validation by the appropriate lines of code. The model is evaluated by the evaluation function cal, which compute the validation losses and accuracy. The ’train-loss’ is computed and stored alongside, in the result diary. Supposing the training stopped earlier, the ’EarlyStopping’ we made an ’Epoch-end’ method to check for that, while printing the respective results. The metrics are reported and printed, including training accuracy and loss, as well as validation accuracy and loss.

With a batch size of 64, 20 epoch sample, we recorded a satisfactory precision result as written in the figure 6. The model achieved a high training accuracy of 96.91 percent with a validation accuracy of 58.93 percent. Observing the metrics in the figure below, we can observe an ever increasing training accuracy with a slightly decreasing training loss over the epoches. The Validation loss decreased from 3.39 percent in the first epoch to 1.58 percent in the last epoch. The training loss is consistently decreasing, demonstrating improved model performance. The overall metric indicates that while the model is still getting better, it can even get better with more epoche and modifications. This would also help to check for overfitting. After series of test from various datasets, we obtained a binary prediction stated above. Hence, the plants could be classified as healthy and unhealthy.

VI. Conclusion

Plant phenotyping and monitoring is an extremely important field of study that has the potential to make a big impact on both agriculture and human health in general. This study has demonstrated that plant diseases might be detected and predicted with deep learning using the right model, such as ResNet50. It can be used in the field of mixed cropping with satisfactory results. Generally speaking, when done correctly, effective plant health monitoring could lead to economic growth. The development of trustworthy sensor-based datasets that can monitor changes in the soil over time, and potentially indicate plant stressors requires particular attention. This would make farming more profitable and sustainable by making it easier to identify diseases while they haven't yet done considerable damage to the plant.

REFERENCES

[1] Sefali Acharya. Plant health monitoring using nanosensor system. In *Nanosensors for Smart Agriculture*, pages 479–492. Elsevier.

[2] Aanis Ahmad, Dharmendra Saraswat, and Aly El Gamal. A survey on using deep learning techniques for plant disease diagnosis and recommendations for development of appropriate tools. 3:100083.

[3] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation.

[4] Zongmei Gao, Zhongwei Luo, Wen Zhang, Zhenzhen Lv, and Yanlei Xu. Deep learning application in plant stress imaging: A review. 2:430–446.

[5] Peng Jiang, Yuehan Chen, Liu Bin, Dongjian He, and Chunquan Liang. Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks. PP:1–1.

[6] Andreas Kamilaris and Francesc X. Prenafeta-Boldú. Deep learning in agriculture: A survey. 147:70–90.

[7] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 740–755. Springer International Publishing.

[8] Jun Liu and Xuewei Wang. Plant diseases and pests detection based on deep learning: a review. 17(1):22.

[9] Kareem A. Mosa, Ahmed Ismail, and Mohamed Helmy. Introduction to plant stresses. In Kareem A. Mosa, Ahmed Ismail, and Mohamed Helmy, editors, *Plant Stress Tolerance: An Integrated Omics Approach*, SpringerBriefs in Systems Biology, pages 1–19. Springer International Publishing.

[10] Dr M. Lakshmi Naga Nandini*. DETECTION OF PLANT DISEASES AND PESTS USING DEEP LEARNING MODELS: A RECENT RESEARCH. 18(2):474–501. Number: 2.

[11] Sapna Nigam and Rajni Jain. Plant disease identification using deep learning: A review. 90:249–57.

[12] Yuriko Osakabe, Keishi Osakabe, Kazuo Shinozaki, and Lam-Son P Tran. Response of plants to water stress. *Frontiers in plant science*, 5:86, 2014.

[13] Kumari Shibani, KS Sendhil Kumar, and G Siva Shanmugam. An effective approach for plant monitoring, classification and prediction using iot and machine learning. In *Artificial Intelligence and Evolutionary Computations in Engineering Systems*, pages 143–154. Springer, 2020.

[14] Vimal K. Shrivastava and Monoj K. Pradhan. Rice plant disease classification using color features: a machine learning paradigm. 103(1):17–26.

[15] Matthew D. Zeiler and Rob Fergus. Stochastic pooling for regularization of deep convolutional neural networks.

[16] Liyuan Zhang, Huihui Zhang, Yaxiao Niu, and Wenting Han. Mapping maize water stress based on UAV multispectral remote sensing. 11(6):605. Number: 6 Publisher: Multidisciplinary Digital Publishing Institute.

VII. DECLARATION OF ORIGINALITY

I, Eze Nnaemeka Valentine, herewith declare that I have composed the present paper and work by myself and without the use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form have not been submitted to any examination body and have not been published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that my work may be checked by a plagiarism checker.