

Real-Time Agriculture Plant Leaf Monitoring and Disease Identification System using Raspberry Pi

T. Nagaraju,

M.Tech Scholar, Department of ECE, QIS college of Engineering and Technology, Ongole,
Andhra Pradesh, India :: nagarajuvasudevan09@gmail.com

B. Malleswari,

M.Tech (Ph.D), Associate Professor, Department of ECE, QIS College of Engineering and
Technology, Ongole, Andhra Pradesh, India :: malleswari.b@gmail.com

Article Info

Page Number: 382 - 399

Publication Issue:

Vol 70 No. 2 (2021)

Abstract

The production of agricultural goods is an essential need for any nation. Monitoring the elements in the environment is an important step, but it is not the only way to increasing agricultural yields. The range of crops that may be grown in India is quite broad. More than five hundred different kinds of crops are cultivated there. If plants are afflicted by diseases, then farmers have a very tough work ahead of them in terms of monitoring plant leaves. As a result, the agricultural productivity of the nation and its economic resources are both negatively impacted. The reasons for this include a decrease in the number of specialized farmers and a shortage of human resources in the field of plant pathology. Using methods such as image analysis and accurately predict and detect pests and diseases and pests at an early stage can reduce the workload of farmers and prevent economic losses in advance. In this project use a novel method by combining the both Azure services and IoT will lead to increase the disease identification. This will give a farmer for better results to increase the yield of crops. Raspberry Pi is the brain of our project to do different tasks, in part cares of irrigation and another will take care of plant leaf images. The system has Raspberry Pi along with the camera module. Camera module capture the leaf image and sends for classification result, azure custom vision will play crucial role for classification the disease. IoT Edge will send the information to cloud for action taken. This system gives better results to produce the yield of crops.

Article History

Article Received: 05 September 2021

Revised: 09 October 2021

Accepted: 22 November 2021

Publication: 26 December 2021

I. Introduction

Agriculture is sometimes referred to be a nation's "backbone" and is a crucial source for the provision of food. It is a reality that this applies to all nations, whether they are developed, developing, or even on their way to being developed. Agriculture is an important factor in ensuring the continued existence of all living beings. It is a source of nutrition not only for humans but also for other living organisms [1]. The rise in the number of people living in poor and developing nations drives up the demand for food in such countries. All of the nations will have difficulties as a result of this. The agricultural industry plays a significant role in the process of a nation's economic growth [2]. The growth of industry and agriculture go hand in hand, like two sides of a coin.

Plants play a vital role in human production, life and even survival. However, due to the continuous expansion of human production and life, excessive hunting and gathering behavior of human beings, pollution of many factories, and the development of large-scale land vegetation [3-4], the natural ecosystem has been greatly changed, making the environmental adaptability of plants reduced, self-regulation disorders, leading to plant diseases.

Plant diseases are mainly caused by biological factors (biological pathogens) and non-biological factors (unsuitable living environment), which make plant growth abnormal and can lead to plant death in a certain period of time [5]. Long-term illness can lead to the extinction of plant species or the extinction of infected other species. This effect can often be intuitively reflected in the leaves and roots of plants, making the leaves and rhizomes of diseased plants sick [6-8]. Targeted treatment and protection of diseased plants is particularly important.

The traditional detection of plant leaf diseases [9] is mainly completed through manual observation and judgment, and in the diagnostic process, due to the similarity of the characteristics of the leaf disease area, the diagnostic process is time-consuming, laborious, expensive, and due to human subjectivity, it may lead to false detection. The sample leaf disease images are given in figure 1



Figure 1: (a) Tomato - Late blight (b) Cucumber - Downy mildew (c)Grape - Black rot

The present invention is intended to solve one of the above technical problems at least to a certain extent. The present invention discloses a plant leaf disease detection method [10-11] based on convolutional neural network, in real-time agricultural scenarios to obtain plant leaf health status and the occurrence of diseases and insect pests image to establish a plant leaf disease detection data set, different diseases of plant leaves for multi-angle [12], different weather, different light under a variety of types of image data acquisition; The present invention uses a convolutional neural network to determine the presence or absence of diseases on the surface of plant leaves [14-15] and the determination of specific types of diseases, which can automatically monitor the growth state of

plants, and the present invention uses a deep learning model based on convolutional neural networks to improve the accuracy of disease detection [16].

For the purposes of this project, the computer vision tools and the localized automatic azure and IoT system will be used experimentally, considering that it is one of the systems with the best results and the least diseases faced by the plants. By combining azure and custom vision and IoT edge, we can reduce the cost of the system and provide good results simultaneously in the both the fields.

II. Literature Survey

Ashqar and Abu-Naser [17] conducted a research that consisted of analysing 9,000 photos of tomato leaves to build a model that could be used on cellphones, with the goal of recognising 5 different illnesses. In order to do this, they produced a model that could be used on smartphones. The model would be based on a deep convolutional network, but it was made up of two parts: the first part of the model (the extraction functions), which was the same for full colour focus and grayscale focus, consisted of 4 layers convolutional with Relu activation function, each accompanied by Max Pooling layer; the second part would contain two dense layers to contain the two approaches, colour and grayscale; and the final part of the model would be composed of two dense layers to contain the two approaches. In the end, they demonstrated that working with colour features yields outcomes that are 99.84% better than working with grayscale, which yielded 95.54% better results.

Barbedo et al., [18] has been working with convolutional neural networks of several different crops to look for different levels of disease in plants. They have been able to classify healthy crops with an accuracy of 89%, crops that are slightly sick with an accuracy of 31%, crops that are moderately sick with an accuracy of 87%, and crops that are severely sick with an accuracy of 94%. He draws the conclusion that the categorization of plant diseases based on digital photographs is very challenging, despite the fact that the findings are good. On the other hand, the constraints of the data set, both in terms of the amount of samples and the diversity of samples, continue to restrict the development of really complete algorithms that are capable of classifying diseases.

Mengistu et al. [19] used an ANN, a KNN, a Naive Bayes, and a hybrid of self-organizing maps (SOM) and radial basis function (RBF) in order to determine the best classifier to identify coffee rust, wilt diseases (CWD), and the CFD that affects the coffee fruit. All of these methods were used in order to determine the best classifier to identify coffee rust, wilt diseases They indicate in their paper that they achieved an accuracy of 58.16% for KNN, 53.47% for Naive Bayes, 79.04% for ANN, and 90.07% for the combination of RBF and SOM, which demonstrates a significant increase over the prior approach. Despite the fact that they make a point that the latter requires much more time to train,

Tests were carried out by Singh and Misra [20] to see if illnesses or burns that appeared on the leaves could be identified. Bananas, beans, lemons, and roses were the crops that were looked at in this research. After processing the photos, they suggest employing genetic algorithms to do segmentation, and then grouping the results. They decided to employ the colour concurrency approach for the process of feature extraction because they believe that it is superior to use a colour

picture rather than the more conventional grey scale image. They used MDC with K-Mean to make the classification, obtaining 86.54%; they then used MDC with an algorithm that they proposed, obtaining an improvement of 93.63%; finally, they used SVM with the proposed algorithm, acquiring a significant improvement of 95.71%; all three methods were used to obtain the same results. These numbers are representing a general average across all four civilizations that were under investigation.

According to Qin et al. [21], the researchers have developed a workable approach for the detection and identification of four illnesses that affect alfalfa. Through the use of the ReliefF, 1R, and CFS techniques, they were able to extract 129 texture, colour, and form characteristics from the 1,651 photos. They used SVM, KNN, and Random Forest in order to categorise the disorders. discovering that the best predictor was SVM and the RelifF technique for determining the characteristic, because they obtained 97.64% accuracy for such training set and 94.74% accuracy for the test set. obtaining this result led to the discovery that SVM was the best classifier.

Khan et al. [22] suggests using deep convolutional network architectures such as VGGNet and AlexNetOWTBn to automate the process of identifying tomato leaf diseases. Early blight, powdery mildew, and mildew were the diseases that needed to be classified. Preprocessing the gathered photos using various image processing methods, including as noise reduction, regression, and image processing improvement, results in a decrease in both expenses and the amount of time required for computing. After that, he retrieved the properties of the data set by using convolution maps, which are maps in which pictures of healthy and sick leaves are applied to the data that has been entered. Despite the fact that the structure has been shown.

III. Proposed System

Figure 2 shows the proposed block diagram which is used to detect the real time image disease detection. There 4 different sections will place very crucial role.

1. Azure IoT Hub
2. Azure IoT Edge
3. Custom Vision
4. Docker Desktop

Azure provides different services for resolving the real time scenarios. The process flow of the proposed method is starts with camera, it will capture the video frame and feed the data to the camera capture module and every frame will pass to the Image classifies service. Custom vision is a service available in azure, which can provide both the image classifier and image detection algorithms. Created custom vision image classifier service for identifying the leaf disease. Added labels for all the disease and started custom vision training. After completion of the training, the trained model is imported into image classifier service to predict the disease with newly captured image.

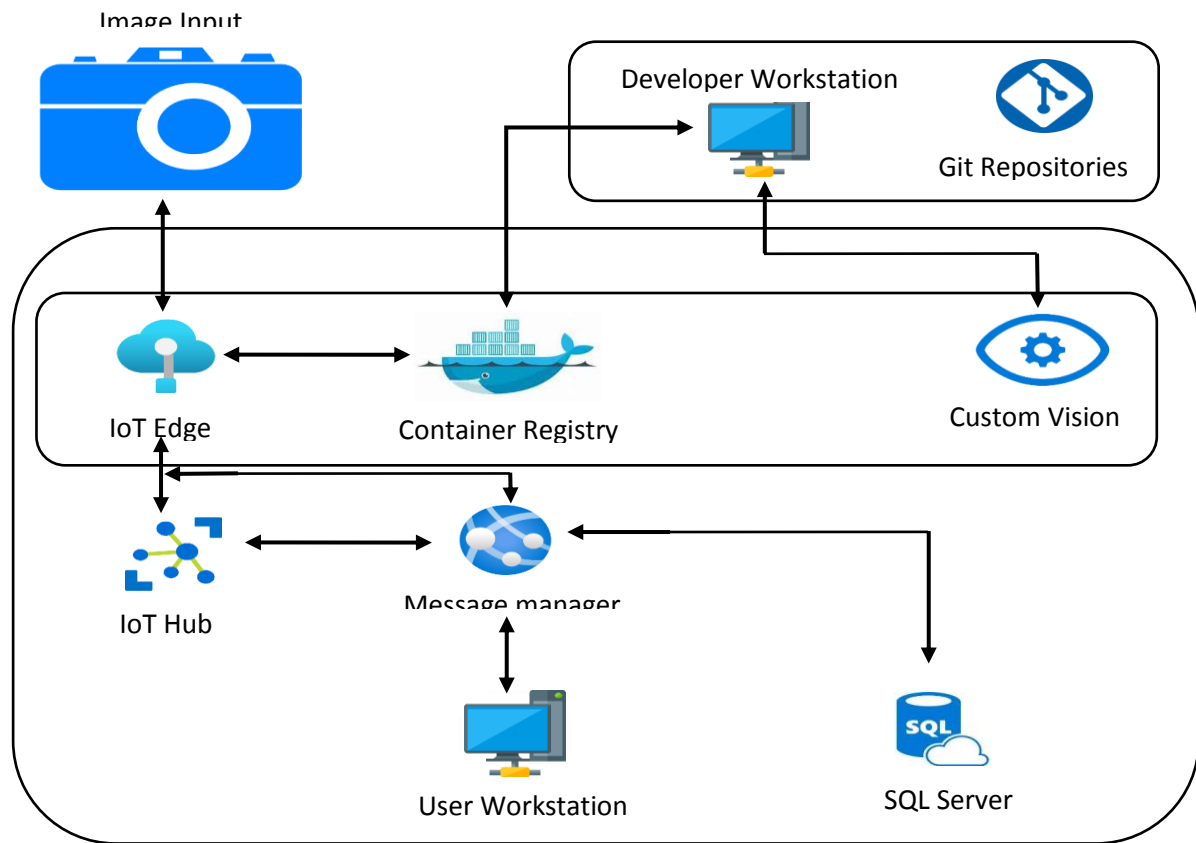


Figure 2: Proposed Block Diagram

Docker images makes these process simple for uploading into IoT Edge device. In this Raspberry Pi4 acts as the IoT edge device, which has 8GB ram and powerful processor to identify the disease quickly.

There are multiple steps to follow to create all the services. The steps are shown below:

Create Azure Account:

Step 1: Sign up for Azure by going to the site at portal.azure.com.

Step 2: After successful signup, go for sign in and then it will redirect to home page of azure.

IoT Hub Setup:

Step 3: In the search box, type iotHub, and then pick IoT Hub from the list of services.

Step 4: Simply selecting the create button will bring up the IoT Hub's creation window.

Step 5: In the "basics" page, provide the name of the Subscription, Resource Group, and location of the IoT Hub, then click "Next."

Step 6: Select the connection settings in the Networking section, and then click on next: management.

Step 7: Choose the appropriate tier for your needs on the management page by considering the access type and TLS connectivity. IoT Hub may be created by selecting the Review + Create button.

Step 8: Choose IoT Edge from the list on the left. Click the Add IoT Edge Device button.

Step 9: Please give an authentication type, as well as a name for the device, and then save.

Register a device:

Depending on what works best, register a device by using the Azure command line interface (CLI), Visual Studio Code, or the Azure site.

IoT Edge devices are produced and maintained in a manner that is distinct from IoT devices that do not have edge capabilities inside your IoT hub via the Azure portal.

1. After logging in to the Azure portal, go over to Internet of Things hub.
2. From the menu, choose Devices, and then in the left pane, choose the option to Add Device.
3. Provide the relevant information on the page designated for the creation of a device:
 - Create a meaningful device ID. You're going to need to remember this device ID in the future, so jot it down.
 - Tick the box designated for the IoT Edge Device.
 - For the kind of authentication, choose the Symmetric key option.
 - In order to automatically create authentication keys, make use of the default settings, and then connect the new device to a hub.
4. Select Save.

Retrieve the information needed to finish the installation and provisioning of the IoT Edge runtime now this device registered in IoT Hub. This will allow to do so more quickly.

View the registered devices and obtain information about the provisioning process:

In order to successfully finish the installation and provisioning of the IoT Edge runtime, devices that make use of symmetric key authentication need their respective connection strings.

On the Devices page, you will see a listing of all of the edge-enabled devices that connect to a IoT hub. There is an option of sorting the list according to the kind of IoT Edge Device.

When device is ready to configure, need the connection string that connects the actual device to the identity it has in the IoT hub.

The connection strings of symmetric-key authenticating devices are made accessible for copying in the portal so that other devices might use them.

5. Navigate to the Devices tab of the portal and choose the IoT Edge device ID from the list of available options.
2. Select either the Primary Connection String or the Secondary Connection String, then copy the value from one of them.

Container Registry creation:

Step 1: Select **Create a resource** > **Containers** > **Container Registry**.

On the Basics page, need to input the values for the Resource group and the Registry name fields. It is required that the name of the registration be exclusive.

Using Azure, and consist of between 5 and 50 alphabetic and numeric characters. Create a new resource group in the "West US" Location and give it the name "MyResourceGroup." Then, for the SKU, choose "Basic." This will complete the quickstart.

Leave the rest of the parameters at their default levels. Then, pick the Create and Review options. Select Create after finished evaluating the parameters.

Select the container registry on the portal after the notice indicating that the deployment was successful displays.

IoT edge agent (Raspberry Pi):

a. Install IoT Edge:

First, the package repository must be added by executing the instructions listed below, and then list of trusted keys must be updated to include the Microsoft package signing key.

Installing just only a few simple instructions to complete. Launch a terminal and enter the commands in the following list:

```
curl https://packages.microsoft.com/config/debian/11/packages-microsoft-prod.deb >
./packages-microsoft-prod.deb
sudo apt install ./packages-microsoft-prod.deb
```

b. Install a container engine:

An OCI-compatible container runtime is required for Azure IoT Edge to function. We strongly suggest that you make use of the Moby engine for any production-related situations. There is just one container engine that is officially supported by IoT Edge, and that is the Moby engine. The Moby runtime is compatible with container images created using Docker CE and Docker EE.

```
sudo apt-get update; \
sudo apt-get install moby-engine
```

c. Install the engine for the Moby:

Configure the Moby engine to utilise the "local logging driver" as the logging mechanism after the installation of the engine has been completed successfully.

Open or create the configuration file for the Docker daemon located at /etc/docker/daemon.json.

- Follow the instructions in the following example to change the default logging driver to the local logging driver.

```
{  
  "log-driver": "local"  
}
```

- In order for the modifications to take effect, need to restart the container engine.

d. Install the IoT Edge runtime:

On the IoT Edge device, the security standards are provided and maintained by the service provided by the IoT Edge. Every time the device is powered on, the service will begin running, at which point it will bootstrap the device by beginning to execute the remainder of the IoT Edge runtime.

IoT Edge and other device components that need to interface with IoT Hub are able to have their identities provisioned and managed by the IoT identity service, which began working with version 1.2 of the software.

The procedures that are outlined in this section are representative of the standard procedure that must be followed in order to install the most recent version on a device that has access to the internet. Follow the instructions outlined in the section for "Offline or particular version installation.

IoT Edge and the IoT identity service package should both have their most recent updates installed:

```
sudo apt-get update; \  
sudo apt-get install aziot-edge
```

e. Provision the device with its cloud identity:

Now that the container engine and the IoT Edge runtime have been installed on your device, you are ready for the next step, that is to set up the device including its cloud identity and authentication information. This can be done by going to the settings menu on your device and selecting the "IoT Edge" option.

Using the following command, now easily enable a IoT Edge device to use symmetric key authentication.

```
sudo iotedge config mp --connection-string 'PASTE_DEVICE_CONNECTION_STRING_HERE'
```

Using the `iotedge config mp` command will result in the creation of a configuration file on the device as well as the addition of your connection string to the configuration file.

```
sudo iotedge config apply
```


Make the necessary adjustments to the config file.

f. Verify successful configuration:

Check that the runtime was successfully installed and configured on the device that serves as a IoT Edge.

Ensure that the IoT Edge system service is up and functioning before continuing.

```
sudo iotedge system status
```

Okay denotes a successfully completed status answer.

If need to diagnose an issue with the service, acquire the logs associated with the service.

```
sudo iotedge system logs
```

```
sudo iotedge check
```

Make sure that the device's setup and connection status are correct by using the check tool.

```
sudo iotedge list
```

Take a look at all of the modules that are now active on a IoT Edge device. Notice only the edgeAgent module executing the very first time the service starts up once it has been restarted. Any subsequent modules that deploy to a device will automatically be assisted in installing, and once running, the edgeAgent module will continue to operate in the background.

Visual Studio code:

In order to log in with your Azure credentials, go to view > Command Palette > search bar and enter Azure sign in; this will take you to a web page where you can sign in; after you've signed in, shut the web page and verify whether or not vs code is signed in.

Step 1: Edit deployment.template.json file to change the version of CameraCapture and ImageClassifierService

Step 2: Right click on deployment.template.json and select Generate IoT Edge Deployment Manifest.

It will create deployment.json in config folder and verify deployment.json file everything is okay or not.

Step 3: Then right click on deployment.template.json and select Build and Push IoT Edge Solution.

It will build Docker Images and push into Docker Desktop.

Step 4: After successful completion, right click on deployment.json in config folder and select Create Deployment for Single Edge Device.

Copy the string from IoT Hub>Shared access policies>iothubowner>primary connection string and paste it into above.

Step 5: To see the output of the model right click on IoT hub, it is located in Explorer at bottom we can see AZURE IOT HUB.

Will get this string from IoT Hub>Built-in Endpoints, then copy the content from an Event Hub-compatible endpoint and paste it into the space provided above.

Step 6: Then select the device whichever we want then right click on the module. Then select start monitoring built-in event endpoints.

Docker Desktop:

For the first time whenever we build and push, you might get an error unauthorized access to Docker. To overcome this Error, type a below command

```
docker login -u <CONTAINER_REGISTRY_NAME> -p  
<PASSWORD_CONTAINER_REGISTRY><FQDN_CONTAINER_REGISTRY>
```

Figure 3 explains the block diagram of proposed system. The block diagram contains 4 different sections, one is input camera and second one is IoT edge runtime, third one is container registry and finally custom vision service. Each service will perform different tasks.

There 2 different modules to perform whole operation i.e.,

1. CameraCapture
2. ImageClassifierService.

CameraCapture service will responsible for capture live video stream and send it to image processing end point, the image classifier service will get the live feed from the end point and process the image and convert that result into JSON format.

The ImageClassifierService will use the custom vision service, after successful training custom vision service will generate IoT edge ARM file, which had both model and labels file. ImageClassifierService will use these files to generate the classification result.

Communication between the modules:

The figure shows, how the above 2 modules are communicated each other themselves and with the IoT Hub.

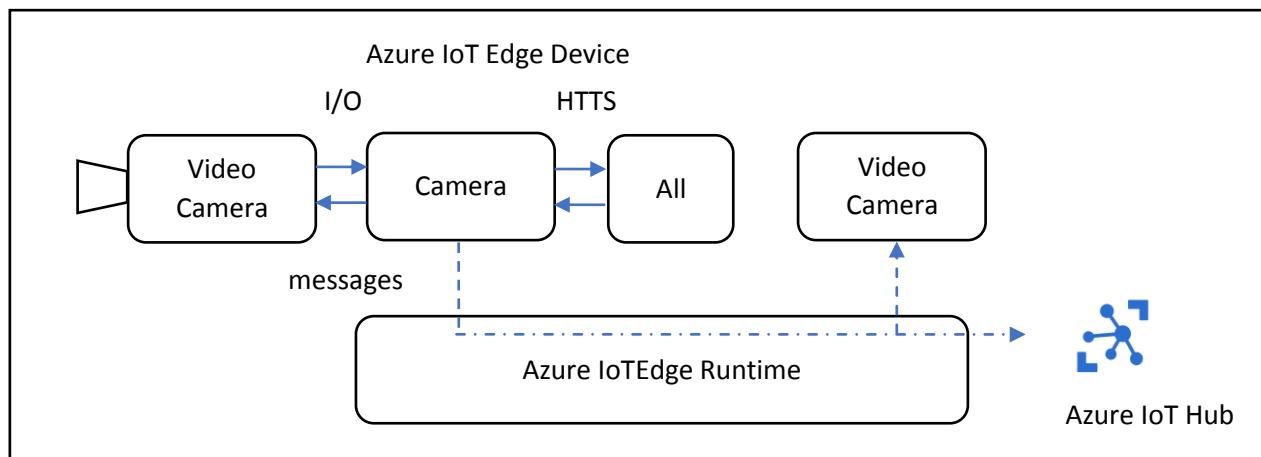


Figure 3: Connection between IoT Edge device and IoT Hub

Follow below steps to run the code:

We need to follow the below steps to deploy the solution on IoT Edge device.

Step 1: create .env file and update container registry credential and make sure that docker engine has access to the docker.

Step 2: Create deployment.template.json file and change the version of each module, when changes made in the modules.

Step 3: Right click on deployment.template.json and select Generate IoT Edge Deployment Manifest, which will create deployment.json file inside the config folder.

Step 4: Then right click on deployment.template.json “Build and put IoT Edge Solution” for building the entire solution.

Step 4: To deploy the solution in IoT Ede device, right click on deployment.json file and choose “Create Deployment for Single device”. Then select the targeted device for deployment.

Step 5: To monitor the IoT Hub result, right click on the device which we uploaded, select “Start Monitoring D2C Message”.

Raspberry Pi:

The Raspberry Pi is a low-cost single-board computer developed by the Raspberry Pi Charitable Trust in England to promote the teaching of programming in schools. Since it was released in 2012, this product has gained a lot of popularity due to its small size, cost, portability, and its high programming and connectivity capabilities. There are currently 9 versions of Raspberry Pi microcomputers with different features. Next, the characteristics of the three most suitable models for this work will be described comparatively.

All of them are the same size (of a credit card) and require a 5V power supply and are equipped with a CSI port for connecting the Pi camera, as well as a DSI port for connecting a touch screen. They also have a port for Micro SD cards to load the operating system and store data. The recommended operating system for normal use of the Raspberry is Raspberry Pi OS (previously called Raspbian) and it is based on Linux. The recommended programming language for working with the Pi is Python.

In addition, they have 40 GPIO (general-purpose input/ output) pins that allow connections with external input and output elements to perform a wide range of diverse functions.

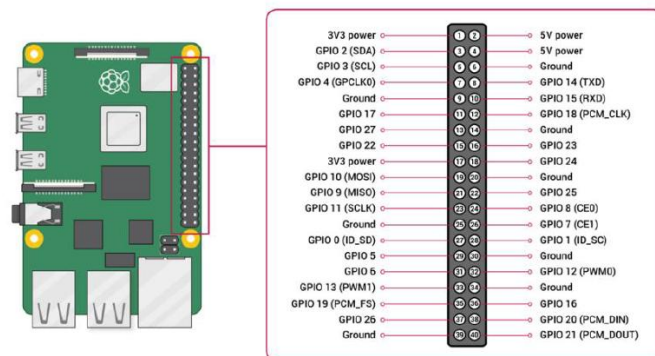


Figure 4: Description of the GPIO pins of the Raspberry Pi

Camera: Webcams that connect through USB often have a lower overall quality compared to camera modules which connect via the CSI interface. They cannot be controlled using `raspistill` and `rasivid` commands in the terminal, nor can they be controlled using the `picamera` recording package in Python. Neither of these options are available. Nevertheless, there could be reasons why would want to connect a USB camera with Raspberry Pi. One of these reasons may be the advantage that it is much simpler to set up several cameras with a single Raspberry Pi. Another reason may be the convenience of the connection.

There are numerous libraries that allow the efficient management of the camera, among which the Python PiCamera library stands out. It is also possible to work with an external camera connected via USB, or even both can be used.

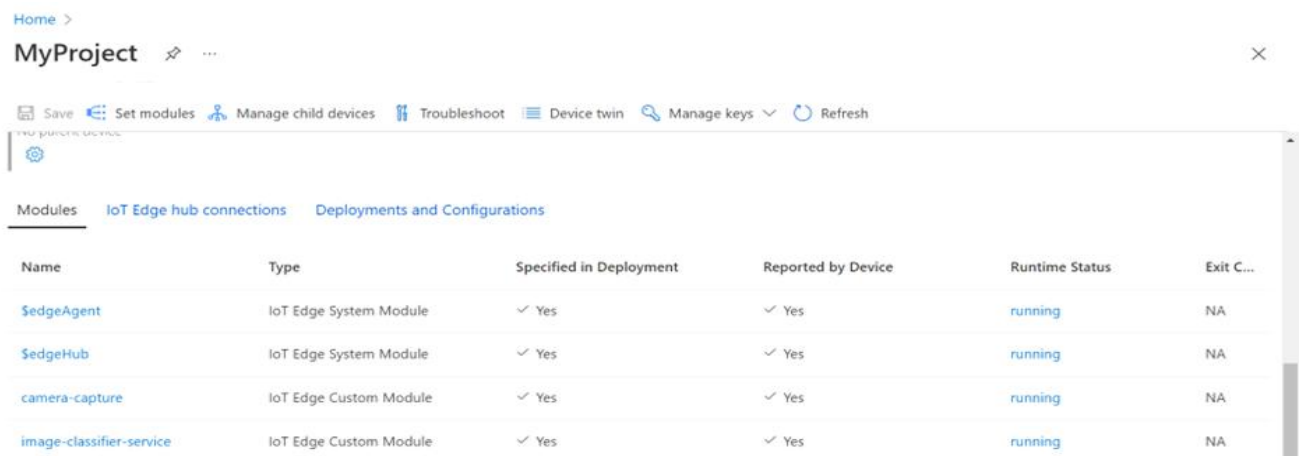


Figure 5: Raspberry Pi with the PiNoIR camera attached.

Results and Discussion:

The original dataset was used as a starting point for the creation of this new dataset, which was developed utilising offline augmentation. Photos of five different crops—apple, maize, grape, potato, and tomato—comprising a total of 31,397 photographs are taken into account. These images include both healthy and sick examples of each crop. Images of six different crop types were analysed and divided into healthy and unhealthy categories. Six of these 12 classes feature photographs of diseases, whereas the other six classes each have images of healthy people.

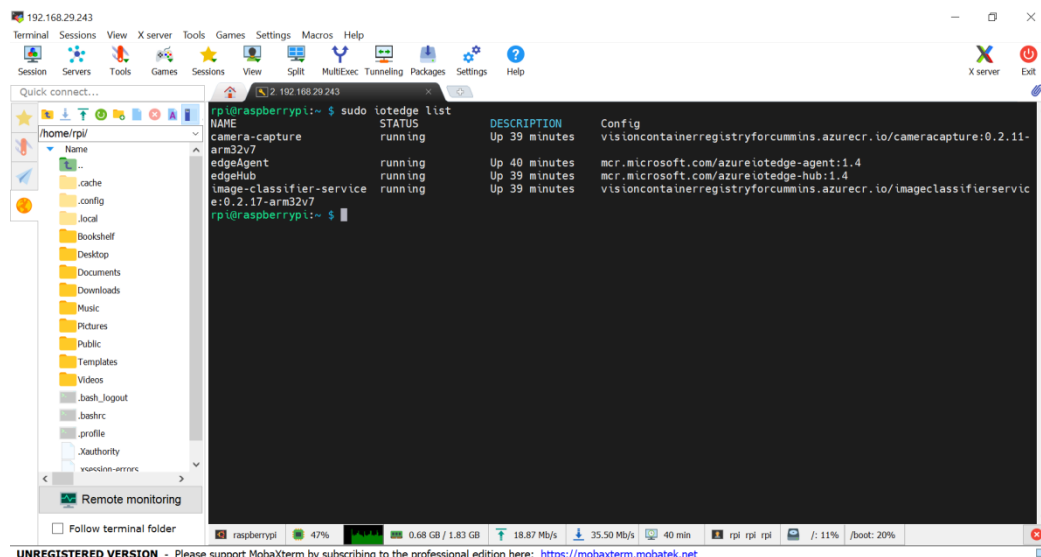
Figure 6 provides an explanation of the Azure IoT edge runtime solution, including the two modules that we developed to execute.



Name	Type	Specified in Deployment	Reported by Device	Runtime Status	Exit C...
\$edgeAgent	IoT Edge System Module	✓ Yes	✓ Yes	running	NA
\$edgeHub	IoT Edge System Module	✓ Yes	✓ Yes	running	NA
camera-capture	IoT Edge Custom Module	✓ Yes	✓ Yes	running	NA
image-classifier-service	IoT Edge Custom Module	✓ Yes	✓ Yes	running	NA

Figure 6: Azure IoT edge runtime

Figure 7 shows the running status of IoT edge devices in IoT hub. Once login to the IoT device, type “sudo iotedget list” will provide the running status of devices. The below screenshot is taken from MobaXterms screen.



NAME	STATUS	DESCRIPTION	Config
camera-capture	running	Up 39 minutes	visioncontainerregistryforcumins.azurecr.io/cameracapture:0.2.11-
edgeAgent	running	Up 40 minutes	mcr.microsoft.com/azureiotedge-agent:1.4
edgeHub	running	Up 39 minutes	mcr.microsoft.com/azureiotedge-hub:1.4
image-classifier-service	running	Up 39 minutes	visioncontainerregistryforcumins.azurecr.io/imageclassifierservice

Figure 7: Running status of IoT edge devices in IoT hub

Figure 8 shows how modules are connected each other in vs code.

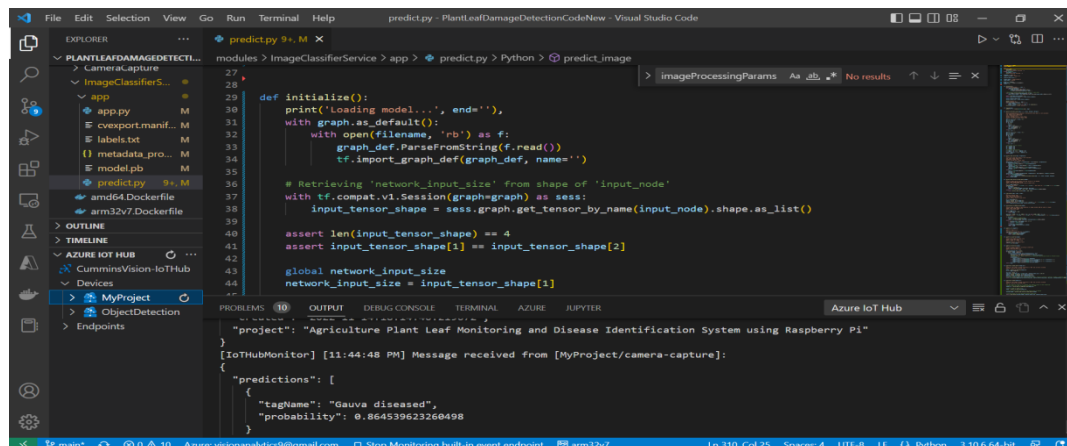

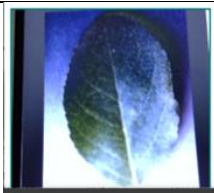

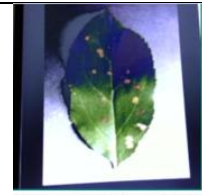

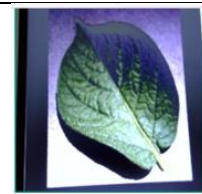

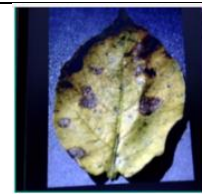


Figure 8: Modules connection VS code

Figure 9 shows example of the classification result of each leaf image. The below table divided into 4 columns. The first column explains the type of leaf image, second column shows the original image, third column shows the camera captured image which runs under IoT edge device, and finally last column shows the result of IoT Hub.

Type of Leaf	Original Leaf Image	Image captured by IoT edge Runtime	Results from IoT Hub Monitor
Apple Healthy			<pre>[IoT Hub Monitor] [11:17:17 PM] Message received from [MyProject/camera-capture]: { "predictions": [{ "tagName": "Apple healthy", "probability": 0.8438912638081177 }], "created": "2022-11-14T17:47:16.876578", "project": "Agriculture Plant Leaf Monitoring and Disease Identification System using Raspberry Pi" }</pre>
Apple Diseased			<pre>[IoT Hub Monitor] [11:23:21 PM] Message received from [MyProject/camera-capture]: { "predictions": [{ "tagName": "Apple diseased", "probability": 0.8720479369163513 }], "created": "2022-11-14T17:53:21.310811", "project": "Agriculture Plant Leaf Monitoring and Disease Identification System using Raspberry Pi" }</pre>
Potato Healthy			<pre>[IoT Hub Monitor] [11:26:00 PM] Message received from [MyProject/camera-capture]: { "predictions": [{ "tagName": "Potato healthy", "probability": 0.8692078113555908 }], "created": "2022-11-14T17:56:00.668437", "project": "Agriculture Plant Leaf Monitoring and Disease Identification System using Raspberry Pi" }</pre>
Potato Diseased			<pre>[IoT Hub Monitor] [11:28:33 PM] Message received from [MyProject/camera-capture]: { "predictions": [{ "tagName": "Potato diseased", "probability": 0.9244812832641602 }], "created": "2022-11-14T17:58:33.530281", "project": "Agriculture Plant Leaf Monitoring and Disease Identification System using Raspberry Pi" }</pre>




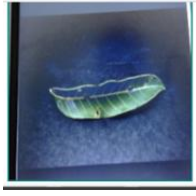




Mango Healthy			<pre>[IoTHubMonitor] [11:35:38 PM] Message received from [MyProject/camera-capture]: { "predictions": [{ "tagName": "Mango healthy", "probability": 0.8274982832641698 }], "created": "2022-11-14T18:05:38.560763", "project": "Agriculture Plant Leaf Monitoring and Disease Identification System using Raspberry Pi" }</pre>
Mango Diseased			<pre>[IoTHubMonitor] [11:42:25 PM] Message received from [MyProject/camera-capture]: { "predictions": [{ "tagName": "Mango diseased", "probability": 0.7562647100646184 }], "created": "2022-11-14T18:12:25.618721", "project": "Agriculture Plant Leaf Monitoring and Disease Identification System using Raspberry Pi" }</pre>
Gauva Healthy			<pre>[IoTHubMonitor] [11:42:30 PM] Message received from [MyProject/camera-capture]: { "predictions": [{ "tagName": "Gauva healthy", "probability": 0.8011188507080078 }], "created": "2022-11-14T18:12:30.420483", "project": "Agriculture Plant Leaf Monitoring and Disease Identification System using Raspberry Pi" }</pre>
Gauva Diseased			<pre>[IoTHubMonitor] [11:44:31 PM] Message received from [MyProject/camera-capture]: { "predictions": [{ "tagName": "Gauva diseased", "probability": 0.9329274296760559 }], "created": "2022-11-14T18:14:31.183802", "project": "Agriculture Plant Leaf Monitoring and Disease Identification System using Raspberry Pi" }</pre>

Figure 9: example of the classification result of each leaf image

Figure 10 show the performance evaluation metrics from custom vision image classifier model. For proposed model precision is 96.7, recall is 100 and mean average precision is 98.9, which are better than existing techniques.

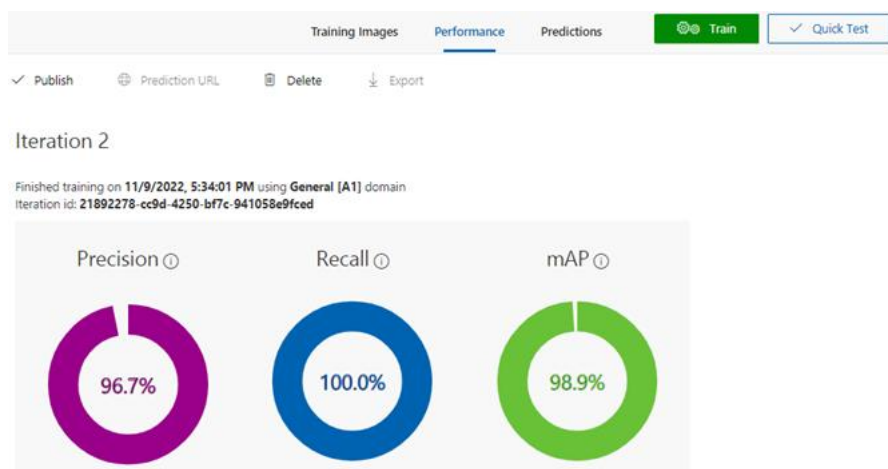


Figure 10: performance evaluation metrics from custom vision

Conclusion:

One of the alternatives that farmers have at their disposal for the early diagnosis of illnesses is disease identification. This particular instance of the issue is not the only one. For the farmer's agricultural area to produce a higher yield, the right kind of fertilizer, insecticides, and crops are required, among other things. This strategy suggests using an AI system that is based on a custom vision analysis, with the addition of IoT for further benefit. The proposed system produced better results when compared to existing methods. The system generated accurate results with less time which increased real time plant disease detection. The farmers can easily use this system to produce the high results.

Future Scope:

Take into consideration all of these criteria, getting the correct results for the case of disease detection. This is due to the fact that several situations, such as illumination and backdrop, are controlling components of the case. In the future, it is probable that illnesses such as foot rot, grain rot, and bacterial sheath may become widespread. It is necessary to identify the plant diseases using a variety of other methods as well. In this day and age of technology, everyone has access to a mobile phone for their own use. Despite this, the user interface for agricultural-related software is insufficient. There is a pressing need for the development of mobile apps that would assist farmers in diagnosing illnesses in plant leaves on their own in order to provide immediate treatments.

References:

1. Fang Y. and Ramasamy R.P., 2015. Current and prospective methods for plant disease detection. *Biosensors*, 5(3), pp.537-561.
2. Balodi R., Bisht S.U.N.A.I.N.A., Ghatak A. and Rao K.H., 2017. Plant disease diagnosis: technological advancements and challenges. *Indian Phytopathology*, 70(3), pp.275-281.
3. Martinelli F., Scalenghe R., Davino S., Panno S., Scuderi G., Ruisi P., Villa P., Stroppiana D., Boschetti M., Goulart L.R. and Davis C.E., 2015. Advanced methods of plant disease detection. A review. *Agronomy for Sustainable Development*, 35(1), pp.1-25.
4. Raina S. and Gupta A., 2021, March. A study on various techniques for plant leaf disease detection using leaf image. In *2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS)* (pp. 900-905). IEEE.
5. Vishnoi V.K., Kumar K. and Kumar B., 2021. Plant disease detection using computational intelligence and image processing. *Journal of Plant Diseases and Protection*, 128(1), pp.19-53.
6. Gupta L., Intwala K., Khetwani K., Hanamshet, T. and Somkunwar R., 2017. Smart irrigation system and plant disease detection. *International Research Journal of Engineering and Technology*, 4(3), pp.2395-56.
7. Chohan M., Khan A., Chohan R., Katpar S.H. and Mahar M.S., 2020. Plant disease detection using deep learning. *International Journal of Recent Technology and Engineering*, 9(1), pp.909-914.
8. Mattihalli C., Gedefaye E., Endalamaw F. and Necho A., 2018, May. Real time automation of agriculture land, by automatically detecting plant leaf diseases and auto medicine. In *2018 32nd*

International Conference on Advanced Information Networking and Applications Workshops (WAINA) (pp. 325-330). IEEE.

9. Chouhan S.S., Singh U.P. and Jain S., 2021. Automated plant leaf disease detection and classification using fuzzy based function network. *Wireless Personal Communications*, 121(3), pp.1757-1779.
10. De Luna R.G., Dadios E.P. and Bandala A.A., 2018, October. Automated image capturing system for deep learning-based tomato plant leaf disease detection and recognition. In *TENCON 2018-2018 IEEE Region 10 Conference* (pp. 1414-1419). IEEE.
11. Ponnusamy V., Natarajan S., Ramasamy N., Clement J.C., Rajalingam P. and Mitsunori, M., 2021. An IoT-Enabled Augmented Reality Framework for Plant Disease Detection. *Rev. d'Intelligence Artif.*, 35(3), pp.185-192.
12. Mohandas A., Anjali M.S. and Varma U.R., 2021, July. Real-Time Detection and Identification of Plant Leaf Diseases using YOLOv4-tiny. In *2021 12th international conference on computing communication and networking technologies (ICCCNT)* (pp. 1-5). IEEE.
13. Alagumariappan P., Dewan N.J., Muthukrishnan G.N., Raju B.K.B., Bilal R.A.A. and Sankaran, V., 2020. Intelligent plant disease identification system using Machine Learning. *Engineering Proceedings*, 2(1), p.49.
14. Mishra S., Sachan R. and Rajpal D., 2020. Deep convolutional neural network based detection system for real-time corn plant disease recognition. *Procedia Computer Science*, 167, pp.2003-2010.
15. Kumar S.S. and Raghavendra B.K., 2019, March. Diseases detection of various plant leaf using image processing techniques: a review. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)* (pp. 313-316). IEEE.
16. Gajjar R., Gajjar N., Thakor V.J., Patel N.P. and Ruparelia S., 2022. Real-time detection and identification of plant leaf diseases using convolutional neural networks on an embedded platform. *The Visual Computer*, 38(8), pp.2923-2938.
17. Ashqar B.A., Abu-Naser S.S.: Image-Based Tomato Leaves Diseases Detection Using Deep Learning (2019).
18. Barbedo J.G.A., 2019. Plant disease identification from individual lesions and spots using deep learning. *Biosystems Engineering*, 180, pp.96-107.
19. Mengistu A.D., Alemayehu, D.M. and Mengistu, S.G., 2016. Ethiopian coffee plant diseases recognition based on imaging and machine learning techniques. *International Journal of Database Theory and Application*, 9(4), pp.79-88.
20. Singh V. and Misra A.K., 2017. Detection of plant leaf diseases using image segmentation and soft computing techniques. *Information processing in Agriculture*, 4(1), pp.41-49.
21. Qin F., Liu D., Sun B., Ruan L., Ma Z. and Wang H., 2016. Identification of alfalfa leaf diseases using image recognition technology. *PLoS One*, 11(12), p.e0168274.
22. Khan S. and Narvekar M., 2020. Disorder detection in tomato plant using deep learning. In *Advanced computing technologies and applications* (pp. 187-197). Springer, Singapore.
23. P Ramprakash, M Sakthivadivel, N Krishnaraj, J Ramprasath. "Host-based Intrusion Detection System using Sequence of System Calls" International Journal of Engineering and Management Research, Vandana Publications, Volume 4, Issue 2, 241-247, 2014

24. N Krishnaraj, S Smys."A multihoming ACO-MDV routing for maximum power efficiency in an IoT environment" Wireless Personal Communications 109 (1), 243-256, 2019.
25. N Krishnaraj, R Bhuvanesh Kumar, D Rajeshwar, T Sanjay Kumar, Implementation of energy aware modified distance vector routing protocol for energy efficiency in wireless sensor networks, 2020 International Conference on Inventive Computation Technologies (ICICT),201-204
26. Ibrahim, S. Jafar Ali, and M. Thangamani. "Enhanced singular value decomposition for prediction of drugs and diseases with hepatocellular carcinoma based on multi-source bat algorithm based random walk." Measurement 141 (2019): 176-183.
<https://doi.org/10.1016/j.measurement.2019.02.056>
27. Ibrahim, Jafar Ali S., S. Rajasekar, Varsha, M. Karunakaran, K. Kasirajan, Kalyan NS Chakravarthy, V. Kumar, and K. J. Kaur. "Recent advances in performance and effect of Zr doping with ZnO thin film sensor in ammonia vapour sensing." GLOBAL NEST JOURNAL 23, no. 4 (2021): 526-531. <https://doi.org/10.30955/gnj.004020>,
https://journal.gnest.org/publication/gnest_04020
28. N.S. Kalyan Chakravarthy, B. Karthikeyan, K. Alhaf Malik, D.Bujji Babbu,. K. Nithya S.Jafar Ali Ibrahim , Survey of Cooperative Routing Algorithms in Wireless Sensor Networks, Journal of Annals of the Romanian Society for Cell Biology ,5316-5320, 2021
29. Rajmohan, G, Chinnappan, CV, John William, AD, Chandrakrishnan Balakrishnan, S, Anand Muthu, B, Manogaran, G. Revamping land coverage analysis using aerial satellite image mapping. Trans Emerging Tel Tech. 2021; 32:e3927. <https://doi.org/10.1002/ett.3927>
30. Vignesh, C.C., Sivaparthipan, C.B., Daniel, J.A. et al. Adjacent Node based Energetic Association Factor Routing Protocol in Wireless Sensor Networks. Wireless Pers Commun 119, 3255–3270 (2021). <https://doi.org/10.1007/s11277-021-08397-0>.
31. C Chandru Vignesh, S Karthik, Predicting the position of adjacent nodes with QoS in mobile ad hoc networks, Journal of Multimedia Tools and Applications, Springer US,Vol 79, 8445-8457,2020