AI-Enabled Smart Agriculture System

PROJECT SUPERVISOR

Dr. Muhammad Farrukh Shahid

INTERNATIONAL PROJECT CO-SUPERVISOR

Dr. Tariq Jameel Saifullah Khanazada

(King AbdulAziz University, Jeddah, Saudi Arabia)

PROJECT TEAM

Syed Baqar Ali Zaidi	K19-1033
Talal Nasir Mirza	K19-1105
Muhammad Talha Altaf	K19-1041

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

KARACHI CAMPUS

May 2023

Project Supervisor	Dr. Muhammad Farrukh Shahid
Project Team	Syed Baqar Ali Zaidi K19-1033 Talal Nasir Mirza K19-1105 Muhammad Talha Altaf K19-1041
Submission Date	May 2, 2023

Dr. Muhammad Farrukh Shahid _____

Supervisor

Dr. Tariq Jameel Saifullah Khanzada _____

International Co-Supervisor (King AbdulAziz University, Jeddah, Saudi Arabia)

Dr. Zulfiqar Ali Memon _____

Head of Department

FAST SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

KARACHI CAMPUS

Acknowledgement

Our profound gratitude goes out to our supervisor Dr. Farrukh Shahid for his guidance, support, and expertise during this endeavor. His insights, constructive criticism, and support were very helpful in guiding the direction of this research.

We would also like to acknowledge the efforts of our co-supervisor Dr. Tariq Jameel Saifullah Khanzada, who provided us with the opportunity to collect local data for this research.

Furthermore, we extend our gratitude to Dr. Nouman Durrani for providing us with the local dataset that was essential for this research.

We also extend our appreciation to the entire jury for their guidance, encouragement, and support throughout this research namely Dr. Nouman Durrani and Ms. Emaan Shahid.

Lastly but most importantly, to our Parents who dedicated their entire lives for us to reach this milestone in our life, we express our utmost gratitude for their selfless act. Their selfless and unconditional efforts and love for us has shaped us to be who we are today.

Award

The National Idea Bank is a collaborative effort to find and create a repository of creative ideas for support and recognition which was inaugurated on February 4, 2021, by Pakistan's President Dr. Arif Alvi.

The purpose of this event is to bring out ideas of the people to the top by having a competition at the end of which winners get a prize to help their goals. Workshops and investor networking opportunities are also provided to the participants.

We won 1st place in the agriculture panel of NIB at the city level in Karachi for this FYP idea and it has been hosted onto their website for a 1 year duration.

Abstract

Agriculture is essential for economic development and food security. However, plant diseases caused by pests can impede progress. Current diagnosis of plant diseases is dependent on farmer's inspection, which is time-consuming, prone to errors, and costly. Automating detection and identification of diseases using artificial intelligence is necessary. On the PlantVillage dataset a supervised deep learning convolutional neural network model first was trained. PlantVillage is a labeled dataset containing 54,306 images of 14 crop and disease types. GoogleNet, AlexNet, VGG-16 and VGG-19 architectures were used to analyze crop images and identify disease patterns. The VGG16 model has achieved the highest accuracy of *93.59%* which is due to dataset splitting into 80% and 20% for training and testing sets respectively. GoogleNet model is the second most accurate achieving *89.33%* accuracy. GoogleNet is the best and most reliable model due to its lowest loss of *0.35* as compared to the highest loss of the bunch of VGG16 being *5.91*. Later this work extended to train these deep learning models on a local wheat_crop dataset. The highest accuracy achieved on a local wheat_crop dataset is *89.74%* using GoogleNet. This research lays the foundation for future work in developing unsupervised machine learning models for plant disease detection, supporting farmers in protecting their crops and ensuring optimal yields.

Contents

List of Figures

List of Tables

List of Acronyms

Acronym	**Full form**
AMI	Amazon Machine Images
API	Application Programming Interface
AWS	Amazon Web Services
CMD	Command Prompt
EC2	Elastic Compute Cloud
DLCNN	Deep Learning Convolutional Neural Network
HTTPS	HyperText Transfer Protocol Secure
SSH	Secure Shell
DL	Deep Learning
CNN	Convolutional Neural Network
AI	Artificial Intelligence
CV	Computer Vision
IOT	Internet of Things
UAS	Unmanned aircraft system
PA	Precision Agriculture
UAV	Unmanned aerial vehicles
FC	Fully Connected
Conv	Convolutional
ILSVRC	ImageNet Large Scale Visual Recognition Challenge

Introduction

Nowadays, with the rise of technology, Artificial intelligence (AI) is present everywhere. AI excels in optimum solution selection, detection of patterns and making predictions based on significant amounts of data [1]. Several AI technologies have recently been incorporated into a variety of industries such as transport, manufacturing and finance in order to expedite and speed up businesses. AI has not been able to pay much attention to the Agriculture industry which still continues to lack automation [2]. One of the oldest human occupations is considered to be farming. It provides a considerable amount of nutritional resources for both people and animals. Future population growth will cause food consumption to rise by 60% to 70%. [3]. Such an increase in population will inevitably have adverse effects on the whole world, particularly emerging countries [4]. Therefore, the need for a reliable and smart agricultural system is of utmost priority.

One of the major aspects of a smart agriculture system is efficient crop disease detection and prevention techniques, since, if the diagnosis is delayed, it demands substantial pesticide spraying, which has a detrimental impact on healthy plants and lowers their yields [1]. Furthermore, with time and some careless usage, the pathogenic bacteria will undergo a development of a resistive system against the pesticides [5]. This severely degrades the defensive ability of crops against viruses and bacteria. Therefore, to prevent the wastage of food resources, there is a dire need of early and accurate diagnosis of plant diseases. Plant disease diagnosis requires a high level of competence and experience by plant specialists. However, even proficient plant pathologists are sometimes unable to accurately and precisely detect some specific diseases, since the plants might possess symptoms which are invisible to the naked eye. In this case, it might even be too late for action, since the plant might have already undergone severe degradation. Considering these consequences, a fully automated system is of great essence. The system would be specifically designed to analyze, identify and detect plant diseases in the visual spectrum, while simultaneously providing the specialists with more time to focus on the ones which are not visible. These systems can also help by providing thorough analytical information regarding the disease and its possible cures.

With the advancement of AI in technology, modern approaches such as Machine and Deep Learning provide us with an opportunity to apply the effective image processing and analysis techniques in the sector of agriculture, converting it into a smart agricultural system. Deep Learning, a subclass of Machine Learning, contains the fields of Computer Vision (CV) and Deep Neural Networks. In recent times, Deep neural networks can be utilized in multiple fields, including Self Driving Cars [6], Natural Language Processing, Virtual Assistants and many more. An input and an output mapping with a hidden layer connecting the two is called a deep neural network. In terms of plant disease detection, this input could be a picture of a diseased plant and the output a crop-disease pair, which provides us with the classification of the disease present in the given crop. In a neural network, the nodes are quantifiable and accept

mathematical inputs from the inbound edges and make mathematical outputs as the outbound edges. [7]. The end result of these models is the classification and identification of plant diseases with speed and accuracy.

In this research project, we discuss the GoogleNet, AlexNet, VGG-16 and VGG-19 models and their application in detecting and identifying a wide spectrum of diseases in various plants using two datasets: the Plant Village dataset and a local wheat_crop dataset. We have also implemented incremental learning for anomaly detection. Incremental learning is a machine learning technique used to train models on new data while still retaining knowledge from previously learned data [8]. In our case, we use incremental learning to continually update our models with new images, allowing for more accurate and up-to-date disease detection.

Therefore, by using incremental learning in our project, we can ensure that our models are continuously improving and adapting to new data, resulting in disease detection in plants that is higher in effectiveness and accuracy. To sum up, the following are the core features of the proposed models:

1. Classification of plant disease.
2. Accurate prediction of diseases.
3. Detection of diseases at an early stage.
4. Abnormality detection.
5. Providing a roadmap for future autonomous AI-Enabled Agricultural Systems.

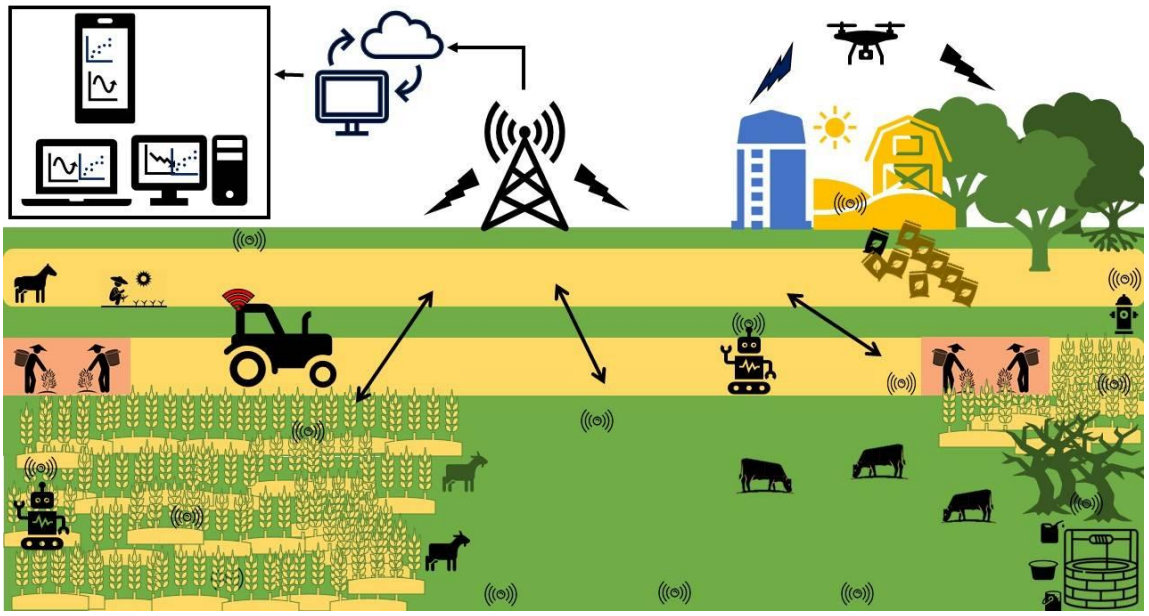

Fig 1: Fully Autonomous Agriculture System

In figure 1, the image at hand portrays our vision of the agriculture industry empowered by Artificial Intelligence (AI) technology. The image's context suggests that AI-enabled agriculture is a promising field that has the potential to revolutionize the

traditional farming industry. At the forefront of this technological revolution, the image depicts a signal tower, which signifies the high-speed wireless networks that enable seamless transmission of data to the cloud. This, in turn, allows machines, connected through mobile phones, to carry out a range of tasks that include disease detection, crop analysis, and prediction.

In addition to the signal tower, the image also features a field with a range of animals and robots. These robots are equipped with advanced surveillance capabilities that enable them to monitor the field for any potential threats or anomalies. The robots work in tandem with a drone that patrols the field, providing an aerial perspective of the area and identifying any issues that might require attention.

The image presents a vivid picture of a world where technology and nature work hand in hand to achieve sustainable and efficient agriculture. The use of AI technology in agriculture holds significant promise in terms of increasing crop yield, reducing resource wastage, and improving overall crop health. The image, therefore, serves as a reminder of the immense potential that AI technology holds in transforming the way we produce and consume food.

In our research, we represent an important step towards realizing the vision of AI-Enabled Agriculture. By focusing on disease detection, we are addressing a critical need in the industry to notify farmers quickly about the health of their crops. Through our work, we aim to empower the farmers with the necessary tools and knowledge to make informed decisions, and ultimately lead to higher crop yields and effective sustainable agriculture.

Literature Review

Deep learning (DL) techniques and modern image processing have recently been progressively adopted in the agriculture industry to detect diseased crops. Many of these solutions classify and detect diseased plants using a CNN and a DL model.

In [7], Mohanty et al. trained the AlexNet and GoogleNet models on 38 classes. This contains 14 healthy crops. It also includes 26 disease variations using the PlantVillage Dataset. The deep learning models used are fairly popular and hold their positions as one of the most efficient and accurate models. On the test set data, they attained a maximum accuracy of 99.35% with GoogleNet. Whereas in [9], Sk. Hassan et al. (2021) use the Plant Village Dataset to train their models, for the identification of plant diseases, four alternative models (Inception Resnet V2, Mobile Net V2, Inception V3, and Efficient Net B0) were used, with the Efficient Net B0 model achieving 99.56%. Moreover, the MobileNetV2 architecture reduces the run time of the model. In [10], Chen et al. implemented the VGG-19 model on the ImageNet dataset for the detection of plant diseases. The VGG-19 model was combined with 2 Inception modules to give an overall accuracy of 92%.

Another implementation of the VGG-19 model has been done by Sukhwinder et al. in [11], where they pre-train the VGG- 19 model on the Plant Village dataset and classify apple leaf images. When trained in 20 epochs, they achieved an overall accuracy of 95.75% and 97.52% respectively for grape and apple leaves.

In [12], Abdullah Alatawi et al., utilize the VGG-16 architecture to classify 19 different classes from the PlantVillage dataset. They achieved an accuracy of 95.2% with the loss being 0.4418.

In [13], Bansal P. et al. also work on apple leaf classification by training an ensemble model on the subset of a dataset created by Cornell University [14]. The ensemble consists of pre-trained models of EfficientNetB7, DenseNet121 and EfficientNet NoisyStudent. The model classifies the apple leaves into mainly 4 categories, namely, Healthy, Scab, Cedar Rust and Other diseases. Their ensemble model achieved an overall accuracy of 96.25% when trained on validation dataset.

Another work done on the same dataset, V. V. Srnidhi et al. in [15], uses the deep learning models of DenseNet and EfficientNetB7 to classify the 4 categories defined above in [13]. The models of DenseNet and EfficientNetB7 gave an accuracy of 99.75% and 99.8% respectively.

In [16], M. Shaoib et al. use the Plant Village Dataset and classified segmented tomato leaf disease images using the recently developed Inception Net Convolutional Neural Network to increase disease detection precision. They utilized three of these techniques namely the InceptionNet1, Incep- tionNet2, and InceptionNet3. The research determines that when employing segmented pictures to categorize diseased and healthy tomato leaves, the InceptionNet1 model is the best performing among the three with an overall accuracy of 99.95%.

In [17], Shrivastava et al. collected images of diseased rice plants. The categorization

of the rice plants has been done in four classes namely the Sheath and Bacterial Leaf Blights, Healthy Leaf, and Rice Blast. There were a total of 619 images taken from the fields. They used the transfer learning concept and used CNN to extract features of rice diseases and a support vector machine to classify the diseases. The overall accuracy achieved by them is 91.37%.

In [18], X. Guan proposed a technique of stacking four models namely the Inception Resnet and Densenet, then trained them on an open source database. The database consisted of 36258 photos which were divided into 61 diseased and healthy classes of 10 plant species. According to the author, a considerable increase in accuracy was achieved by the stacking strategy, with an accuracy rate of 87%.

In [19], Caldeira et al. collected a dataset of cotton crops with over 60,659 images captured in total of diseased and healthy leaves. The deep learning models of Resnet50 and GoogleNet were used with an overall accuracy of 89.2% and 86.6% obtained by both respectively.

In [20], P. V. Raja et al. utilize the PlantVillage dataset to train a Resnet50 deep learning model. The author com- pared the Resnet50 with other CNN models to classify plant diseases. In conclusion, the Resnet50 model gave the highest accuracy in the test dataset, with an accuracy rate of 96.63%.

In [21], A. KP et al. obtain a subset of grape leaf images from the Plant Village data and train multiple models namely, VGG-19, Resnet and Densenet on the healthy and diseased grape leaves. According to the author, the Densenet model deals with complex features better than the others, and performs better in smaller datasets. Model achieved 98.27% accuracy with the Densenet, 97.04% with the Resnet and 95.32% with the VGG-19 model.

In [22], Lakshmanarao et al. applies CNNs on the PlantVil- lage dataset with 3 subdivisions of plant leaves, namely Potato leaves, Pepper leaves and Tomato leaves. Each dataset consists of 15 classifications of the corresponding plant. They achieved an accuracy of 98.3% for Potato leaves, 98.5% for Pepper leaves and 95% for Tomato leaves.

In [23], P. Ferentinos utilized an open database containing 87,848 images which consisted of 58 classes and 25 different plants. The best CNN model achieved 99.53% accuracy. In [24], Geetharamani, Pandian, J.A., and colleagues propose a CNN model with nine layers for classifying diseased plants on an open database. An accuracy of 96.46% is achieved with the proposed model, which, according to the author, is greater as compared to the accuracy of classic machine learning architectures.

Author(s)	Dataset	Objective Achieved	Model(s)	Highest Accuracy	Year
Mohanty et al.	PlantVillage	Highly accurate crop disease classifier for smartphones using GoogleNet	AlexNet, GoogleNet	99.35 %	2016
Sk. Hassan et al.	PlantVillage	Efficient and highly accurate DCNN for mobile applications and real-time systems	Inception Resnet V2, MobileNet V2, Inception V3, Efficient Net B0	99.56 %	2021
Chen et al.	ImageNet	Considerably efficient model for rice crop images under complex background conditions	VGG-19	92%	2020
Sukhwinder et al.	PlantVillage	Accurate and efficient model for classifying apple and grape leaves	VGG-19	97.52 %	2022
Bansal P. et al.	Plant pathology	Accurate ensemble model to automate apple leaves disease detection in real-time systems	EfficientNetB7 DenseNet121, EfficientNet NoisyStudent.	96.25 %	2021
V. V. Srnidhi et al.	Plant pathology	Highly accurate models to classify leaf diseases for apple trees using EfficientNet	DenseNet, EfficientNetB7	99.8%	2021
M. Shaoib et al.	PlantVillage	Outperforming and highly accurate InceptionNet model to classify tomato leaf diseases	InceptionNet1, InceptionNet2, InceptionNet3	99.95 %	2022
Shrivastava et al.	Private	Accurate and efficient model for a real-time rice disease classification system	CNN, SVM	91.37 %	2019
X. Guan	Open source	Accurate disease classifier using stacking method with combined CNNs	Inception, Resnet, Inception Resnet and Densenet	87%	2021
Caldeira et al.	Private	Identification of cotton crop lesions with a considerably high accuracy using CNN	Resnet50, GoogleNet	89.2%	2021

P. V. Raja et al.	PlantVillage	Assert higher accuracy of disease detection using Resnet50 as compared to other CNN models	Resnet50	96.63 %	2022
A. KP et al.	PlantVillage	Highly accurate grape leaf disease classification using DenseNet	VGG-19, Resnet, Densenet	98.27 %	2021
Lakshmanarao et al.	PlantVillage	Classifier for potato, pepper and tomato leaves with a considerably high accuracy	CNN	98.5%	2021
P. Ferentinos	Open source	Highly accurate model for an advisory and warning tool using CNN	CNN	99.53 %	2019
Geetharamani et al.	Open source	A nine layered deep CNN model for accurate and efficient disease detection of plants	DCNN	96.46 %	2019
Pandian, J.A et al.	Private	A highly accurate disease classifier using a 14 layered deep CNN model	DCNN	99.965 %	2022
Jadhav et al.	Open source	Soybean leaves classification using AlexNet for a highly accurate disease detection system	AlexNet, GoogleNet	98.75 %	2020
Abdullah Alatawi et al.	Open source	A highly accurate plant disease detection system using VGG-19	VGG-19	95.2 %	2022

Table 1: Summarized Literature Review

In another research [25], Pandian, J.A et al. propose a CNN model with 14 layers for the purpose of plant disease classification. The model was implemented with a newly created dataset which consisted of 58 classifications of different diseased and healthy plants with a total of 147,500 images. The 14-layered model achieved overall 99.9655% accuracy when applied on 8850 test images.

In [26], Jadhav et al. employed 649 and 550 picture samples of damaged and healthy soybean, respectively, to train AlexNet and GoogleNet models , obtaining 98.75% and 96.25% accuracy for both the models.

Looking from the birds eye view, UAVs for smart agri- culture surveillance have also been proposed. In [27], D. Boursianis et al. propose the use of IoT and UAV

technologies to provide a smart agricultural system. With the potential for automated crop disease management, the use of UAV technology in agriculture is deemed critical. UAVs equipped with advanced cameras can capture aerial images of the field, and also predict crop diseases with the help of DL models. In [28], Hunt et al., propose an approach of monitoring remotely using unmanned aircraft systems (UAS) for precision agriculture (PA) applications. The UAS can acquire crop images from low altitude flights with a significantly low pixel size. According to Subeesh, A. Subeesh et al., and C.R. Mehta in [29], AI and IoT can be used to automate and digitalize agriculture. They discuss the implementation of autonomous tractors, intelligent irrigation systems, weed and pest classification using deep learning models, the use of UAVs for pesticide spraying and land surveillance, and the evolution of the 5G technology to provide lightning quick sensor responses.

Research has shown that incremental learning can be particularly useful in domains where new data becomes available frequently, such as in plant disease detection. For example, in

[30] Salima Ouadfel et al. propose an incremental learning method which retains and preserves the previous data and adapts to the incoming data. They tested it on the PlantVillage Dataset to prove its effectiveness.

Motivated by the advancements in the deep learning field, we present an approach based on the deep learning models of VGG16, VGG19, AlexNet and GoogleNet to detect and classify plant diseases, firstly on the PlantVillage dataset, and then on a local private wheat_crop dataset. The idea is to leverage these models and possibly make an application with the hope of combining it with IoT technologies such as, UAV-enabled crop management with field surveillance to deliver an automated and smart farming solution. A potential for a smart agricultural ecosystem exists in the future with the help of these technologies and the resulting ecology will be advantageous to both the owner and the farmer. The goal is to make an automated system which not only performs classification, but it is also self-learning.

Methodology

The methodology section of this work describes the approach taken to develop an accurate and efficient plant disease detection system using DL techniques. To achieve this goal, two datasets were used, the PlantVillage dataset and a local wheat_crop dataset, which contain images of healthy and diseased leaves of various crops. DL models such as VGG-16, VGG-19, AlexNet, and GoogleNet. The best model is saved after each iteration, resulting in a system that can accurately identify diseases in plants by examining key elements in the images.

The following subsections provide a detailed account of each step in the methodology used to develop this plant disease detection system.

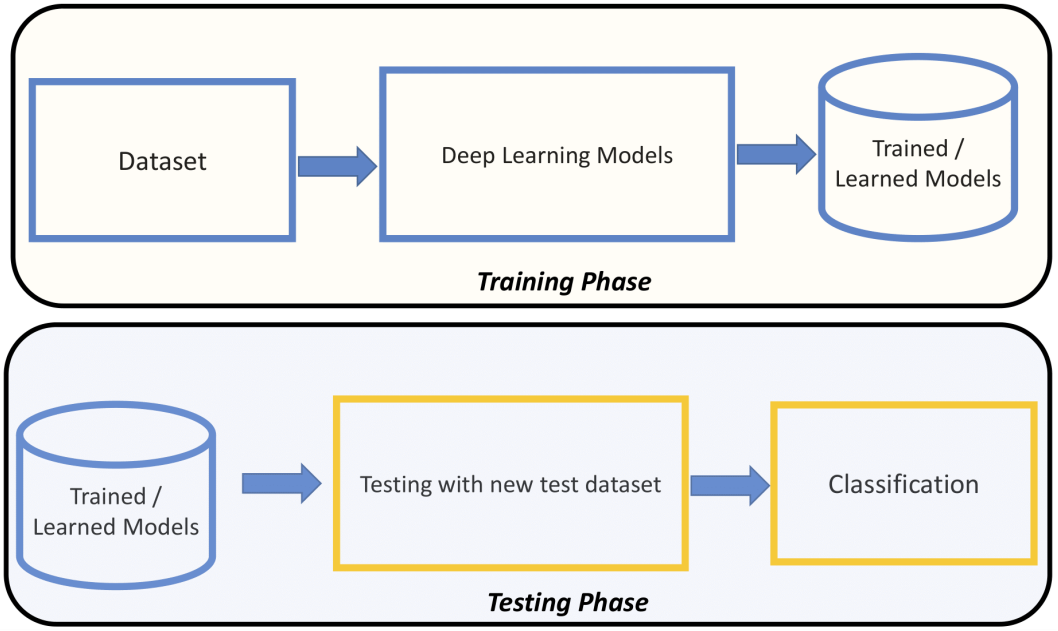

Fig 2: Methodology describing the training and testing phases.

A. Dataset

In this work, there has been use of two datasets. One has been obtained from Kaggle and the other has been obtained for some local wheat crop diseases in plants of Pakistan from a private source. The Kaggle dataset, called "PlantVillage Dataset" [31], is used to extract 54,306 pictures of 14 distinct types of crops with 12 healthy and 26 diseased leaves.

The PlantVillage dataset images have been divided into multiple formats such as colored, gray-scale and segmented. In our work, the training of the data has been done using the raw colored images of healthy and diseased leaves. Some variations of these diseases have been shown in figures 3, 4 and 5.

Fig.3: Pepper Bell Bacterial Spot.

Fig.4: Potato Light Blight.

Fig.5: Tomato Mosaic Virus

The local wheat_crop dataset images are in a raw colored format. This dataset contains a total of 2,489 images that are divided into 5 different wheat diseases, namely, the *Leaf Rust, Common Bunt, Fusarium Head Blight, Common Root Rot and Powdery Mildew* diseases. Some variations of these diseases have been shown in figures 6, 7 and 8.

Fig 6: Common Root Rot

Fig 7: Leaf Rust

Fig 8: Fusarium Head Blight

B. Preprocessing

In pre-processing, the images are gone through some procedures to enhance the data furthermore. These techniques include noise reduction, resizing, flipping, shear, zoom, orientation and rotation etc. The brightness and rgb scale of the are shaped to optimize the images and highlight the significant parts of the image. Figure 9 shows how an image would look after the pre-processing procedure.

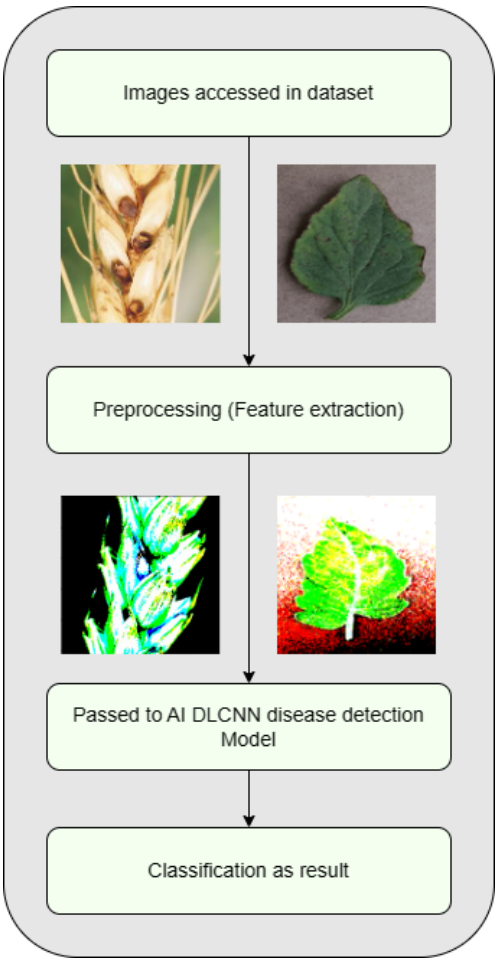

Fig 9: Classification process of images.

C. Deep Learning Models

In our proposed work, the models are trained using two datasets: the Plant Village dataset and a local wheat_crop Pakistani dataset. The PlantVillage dataset consists of 54,306 images. These images have 14 different types of crops. The local wheat_crop dataset contains a total of 2,489 images that are divided into 5 different diseases, namely, the *Leaf Rust, Common Bunt, Fusarium Head Blight, Common Root Rot and Powdery Mildew* diseases. These images are extracted, pre-processed and fed in the CNN model to train the model. CNNs are the basis of the entire DL approach used to implement plant disease detection. The DL models used in our work are fairly popular and hold their positions as one of the most efficient and accurate models, namely VGG-16, VGG-19, AlexNet and GoogleNet. The models are trained on a set amount of epochs, and the best model is saved after each iteration. In order to identify the images, it scans the entire image to extract the key elements. It has the ability to examine images that have been translated, rotated, scaled, and modified. This process has been visually described in Figure 9.

1) VGG-16 and 19:

The VGGNet models proposed by K. Simonyan et al. from Oxford University achieved high accuracy in the ILSVRC 2014 [32]. VGG 19, the first runner-up in the competition, consists of 19 weight, 16 convolutional and 3 FC layers. It uses a 2D

matrix with the size 256x256x3 to handle data and requires pre-processing of RGB images by deducting the mean RGB content value of each pixel in the training set [33], [34]. VGG 16, on the other hand, consists of 16 weight layers with 13 convolutional layers and 3 FC layers. It also uses a 2D matrix with the size 224x224x3 and requires pre-processing of RGB images by deducting the mean RGB value of each pixel in the training set [35]. The first two layers of VGG 16 and VGG 19 are composed of convolutional layers with 3 by 3 filters and 64 filters each, followed by a pooling layer with a stride of 2 and a max pool size of 2x2. Two more convolutional layers with 128 filters each follow, before another pooling layer with the same stride and max pool size. The picture is then reduced to 28x28x256 pixels through the addition of 2 further convolution layers with 256 filters each. Two additional stacks, each with 3 convolution layers, are separated by a 7x7x512 volume max-pool layer that is flattened into a FC layer and then followed by a soft-max layer as an output [33]–[35]. Figure 10 visually describes the architecture of VGG 16 while Table 3 mentions its layers and Figure 11 describes VGG 19 visually while Table 4 mentions its layers.

Layer Name	**Feature Map Size**
Input	224 * 224 * 3
Block 1 Conv 1	224 * 224 * 64
Block 1 Conv 2	224 * 224 * 64
Block 1 Pool	112 * 112 * 64
Block 2 Conv 1	112 * 112 * 128
Block 2 Conv 2	112 * 112 * 128
Block 2 Pool	56 * 56 * 128
Block 3 Conv 1	56 * 56 * 256
Block 3 Conv 2	56 * 56 * 256
Block 3 Conv 3	56 * 56 * 256
Block 3 Pool	28 * 28 * 256
Block 4 Conv 1	28 * 28 * 512
Block 4 Conv 2	28 * 28 * 512
Block 4 Conv 3	28 * 28 * 512
Block 4 Pool	14 * 14 * 512
Block 5 Conv 1	14 * 14 * 512
Block 5 Conv 2	14 * 14 * 512
Block 5 Conv 3	14 * 14 * 512

Block 5 Pool	7 * 7 * 512
Flatten	25088
FC 1	4096
FC 2	4096
Output (Softmax)	1000

Table 2: VGG-16 Model Architecture

Layer Name	**Feature Map Size**
Input	224 * 224 * 3
Convolution 1_1	224 * 224 * 64
Convolution 1_2	224 * 224 * 64
Max Pooling 1	112 * 112 * 64
Convolution 2_1	112 * 112 * 128
Convolution 2_2	112 * 112 * 128
Max Pooling 2	56 * 56 * 128
Convolution 3_1	56 * 56 * 256
Convolution 3_2	56 * 56 * 256
Convolution 3_3	56 * 56 * 256
Convolution 3_4	56 * 56 * 256
Max Pooling 3	28 * 28 * 256
Convolution 4_1	28 * 28 * 512
Convolution 4_2	28 * 28 * 512
Convolution 4_3	28 * 28 * 512
Convolution 4_4	28 * 28 * 512
Max Pooling 4	14 * 14 * 512
Convolution 5_1	14 * 14 * 512
Convolution 5_2	14 * 14 * 512
Convolution 5_3	14 * 14 * 512

Convolution 5_4	14 * 14 * 512
Max Pooling 5	7 * 7 * 512
FC 6	1 * 1 * 4096
FC 7	1 * 1 * 4096
FC 8	1 * 1 * 1000 (output)

Table 3: VGG-19 Model Architecture

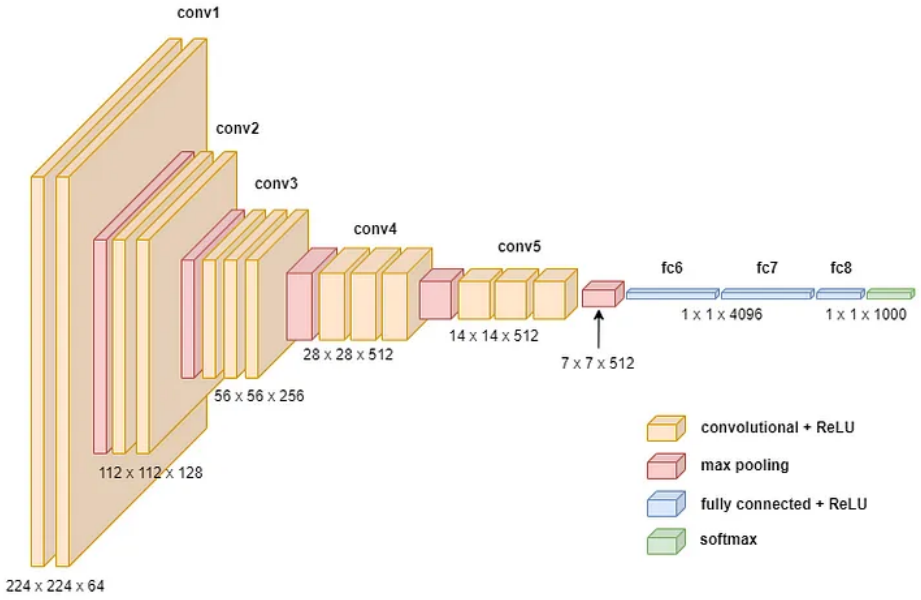

Figure 10: VGG 16 Architecture [44].

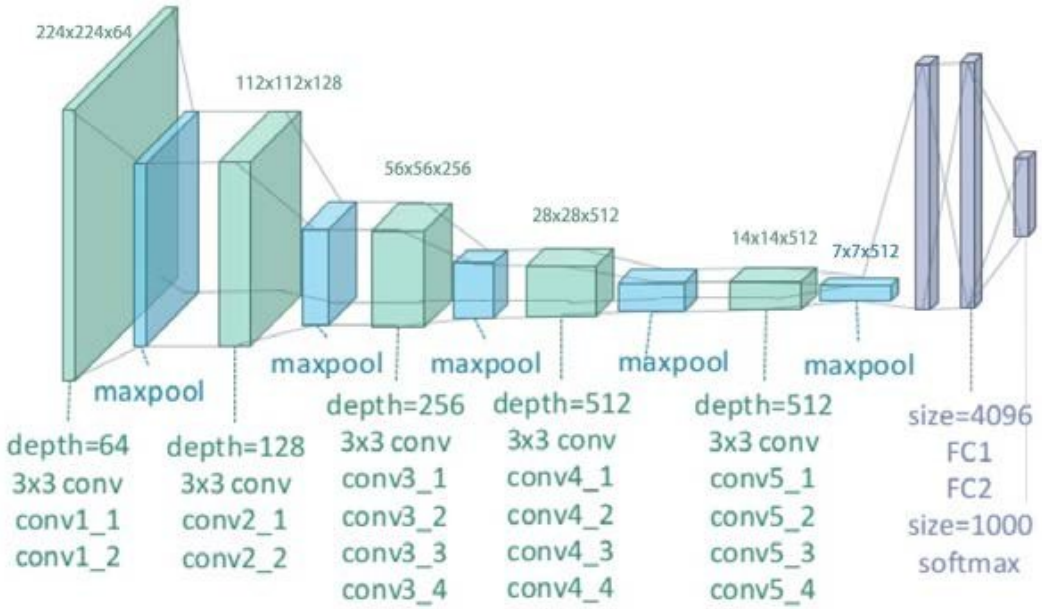

Figure 11: VGG 19 Architecture [45].

2) AlexNet:

The second model we have used in our work is the AlexNet deep neural network. Proposed by Alex Krizhevsky et al. [36] in 2012, AlexNet is also one of the exceptional models such as the VGG-19 and the GoogleNet models. The first ever CNN model to win the ImageNet ILSVRC-2012 competition, coming in at first place was AlexNet[32]. There are five convolutional layers in the AlexNet architecture with 3 max-pooling , 2 normalization , 2 FC, and 1 softmax layer. A nonlinear activation function, ReLU and convolutional filters make up each convolutional layer [37]. To accomplish maximum pool- ing, the pooling layers are employed. Because there are fully linked layers, the input size is fixed. The input size of AlexNet is 227*227*3 and has 60 million parameters [38] [39]. Figure 12 shows the visual representation of the model while Table 5 mentions its layers.

Layer Name	**Feature Map Size**
Input	227 * 227 * 3
Convolution 1	55 * 55 * 96
Max Pooling 1	27 * 27 * 96
Convolution 2	27 * 27 * 256
Max Pooling 2	13 * 13 * 256
Convolution 3	13 * 13 * 384
Convolution 4	13 * 13 * 384
Convolution 5	13 * 13 * 256
Max Pooling 3	6 * 6 * 256
FC 1	4096
FC 2	4096
Output	1000

Table 4: AlexNet Model Architecture

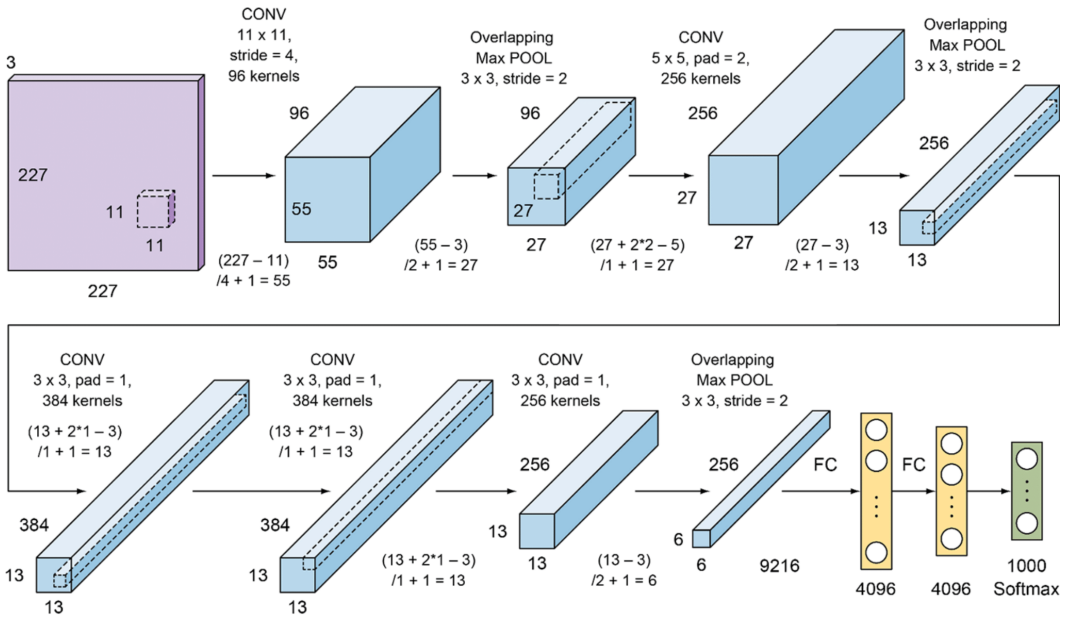

Figure 12: AlexNet Architecture [40].

3) GoogleNet:

GoogLeNet, proposed by Szegedy et. al. [41], is an inception architecture which won the ILSVRC 2014 competition, coming in first place. As compared to the previous winners, AlexNet, which won the ILSVRC 2012 and the runner up of ILSVRC 2014, VGG 19 [32], it generally has a significantly higher accuracy rate than the both of them.

GoogLeNet consists of 22 layers in total, with 9 inception modules. There are 3 convolutional layers in which one of them is to reduce dimensions. There's also 4 max pooling, 2 normalization, one FC and 1 average pooling layer. Finally, for the output, a linear layer is enabled with softmax activation. It has almost 6.8 million parameters [41]. 1 max-pooling layer, 4 convolutional layers for dimension reduction, and two convolutional layers are present in each inception module in turn. The fully-connected layer of GoogleNet also employs dropout regularization, and ReLU activation function is used for all convolutional layers. A visual representation of GoogleNet's architecture is shown in Figure 13 while Table 6 mentions its layers.

Layer Name	**Feature Map Size**
Input	224 * 224 * 3
Convolution 1	56 * 56 * 64
Max Pooling 1	28 * 28 * 64
Convolution 2	28 * 28 * 192
Max Pooling 2	14 * 14 * 192
Inception 3a	14 * 14 * 256

Inception 3b	14 * 14 * 480
Max Pooling 3	7 * 7 * 480
Inception 4a	7 * 7 * 512
Inception 4b	7 * 7 * 512
Inception 4c	7 * 7 * 512
Inception 4d	7 * 7 * 528
Inception 4e	7 * 7 * 832
Max Pooling 4	4 * 4 * 832
Inception 5a	4 * 4 * 832
Inception 5b	4 * 4 * 1024
Average Pooling	1 * 1 * 1024
Dropout	1 * 1 * 1024
Output	1000

Table 5: GoogleNet Model Architecture

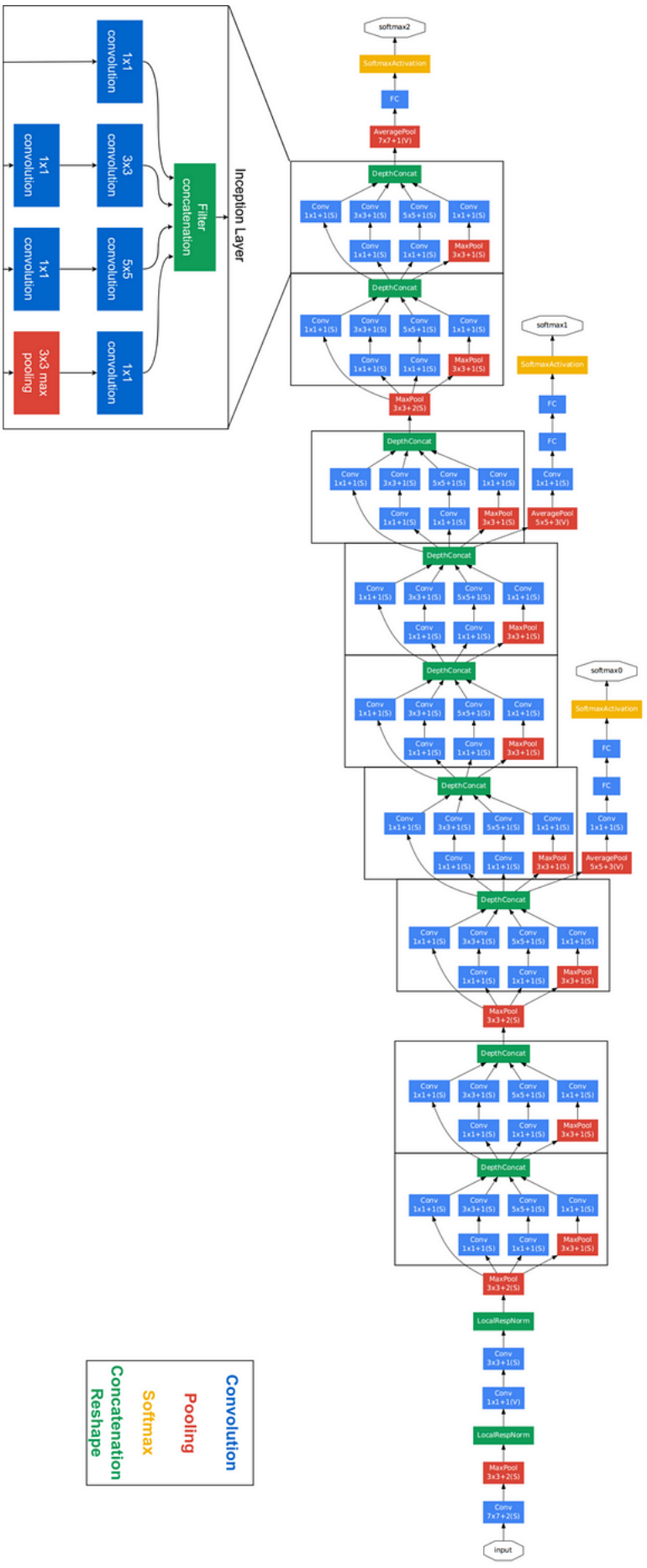

Figure 13: GoogleNet Architecture [41].

D. Anomaly Detection:

Anomaly Detection is a process which identifies outliers in given data which deviate from the norm. According to Chandola et al., anomaly detection can be performed using various techniques, such as statistical modeling, machine learning, and clustering [42]. Its purpose in our project is to identify input of abnormal images (e.g. images of objects besides plants i.e cars, animals etc) to the model in a real-time environment.

The machine learning technique used to group data points together based on their similarity is called clustering. It is commonly used in anomaly detection to find patterns or outliers in data. One popular clustering algorithm is k-means clustering, which is a method for partitioning a dataset into k clusters based on their similarity [43]. In this research we have made use of the k-means algorithm to cluster normal and abnormal data separately.

Experimental Setup, Simulations and Results

In this work, experiments were first conducted on the PlantVillage dataset initially, and then on the local wheat_crop dataset to evaluate effectiveness of the DL models. The experimental setup involved training four deep learning models namely VGG-16, VGG-19, AlexNet, and GoogleNet, on the two datasets. The models were trained using pre-processed images of diseased and healthy leaves for PlantVillage, and only the diseased leaves for the local wheat_crop dataset.

1. Training on PlantVillage Dataset:

No.	Parameter	Value
1	Input Layer size	224*224*3
2	Epochs	6
3	Batch Size	32
4	Image Augmentation	Rotation, width and height shift, flipping, shear, zoom, and preprocessing function
5	Hardware resource	1 GPU

Table 6: Training Parameters for VGG-16

Figure 14: Training and Validation Accuracy Plot for VGG-16

Figure 15: Training and Validation Loss Plot for VGG-16

Results for VGG16 are shown in figures 14 and 15. In figure 14, training and validation accuracies of the VGG16 model are shown at each epoch, in which the last epoch show training and validation accuracies are *95.25%* and *93.73%* respectively. The high values can be attributed to the approach of splitting the dataset into 80% and 20% respectively for training and testing sets. In figure 15, training and validation losses of the VGG16 model are shown at each epoch, in which the last epoch has the following training and validation loss values respectively: *4.07* and *5.91*.

No.	**Parameter**	**Value**
1.	Input Layer size	224 * 224 * 3
2.	Epochs	6
3.	Batch Size	32
4.	Image Augmentation	Rotation, width and height shift, flipping, shear, zoom, and preprocessing function
5.	Hardware resource	1 GPU

Table 7: Training Parameters for VGG-19

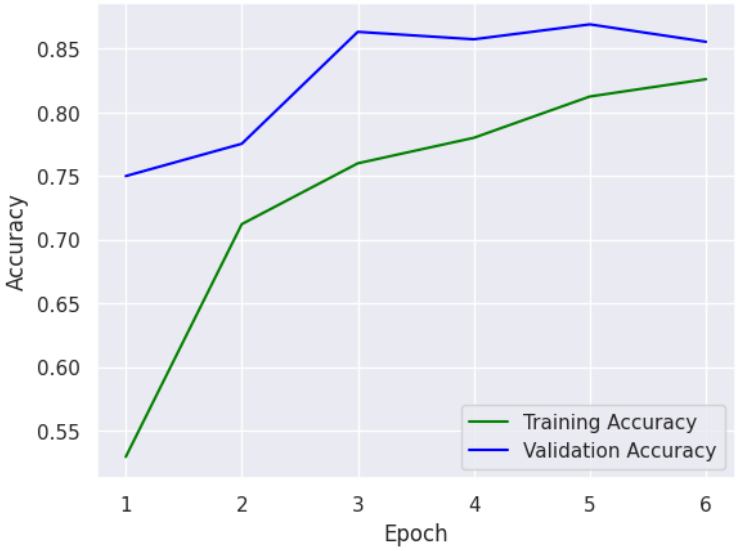

Figure 16: Training and Validation Accuracy Plot for VGG-19

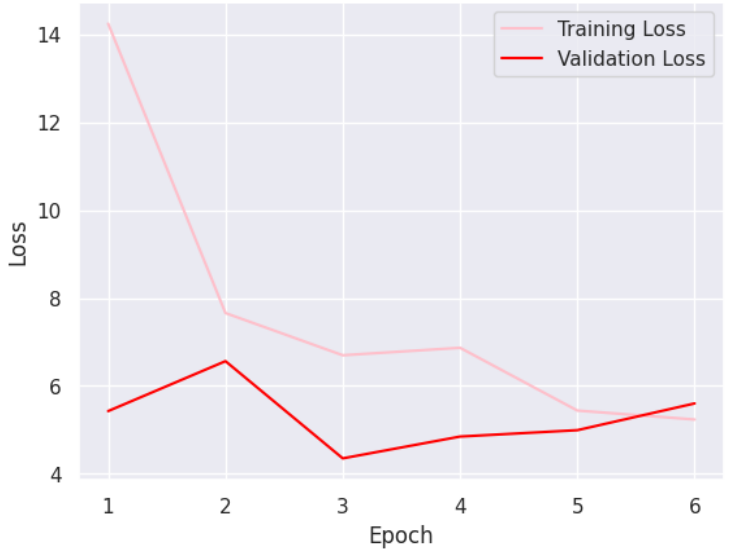

Figure 17: Training and Validation Loss Plot for VGG-19

Results for VGG19 are shown in figures 16 and 17. In figure 16, training and validation accuracies of the VGG19 model are shown at each epoch, in which the last epoch show training and validation accuracies are *82.61%* and *85.55%* respectively. In figure 17, training and validation losses of the VGG19 model are shown at each epoch, in which the last epoch has the following training and validation loss values respectively: *5.24* and *5.60*.

No.	**Parameter**	**Value**
1.	Input Layer size	227 * 227 * 3
2.	Epochs	10
3.	Batch Size	32

4.	Image Augmentation	Rotation, width and height shift, flipping, shear, zoom, and preprocessing function
5.	Hardware resource	1 GPU

Table 8: Training Parameters for AlexNet

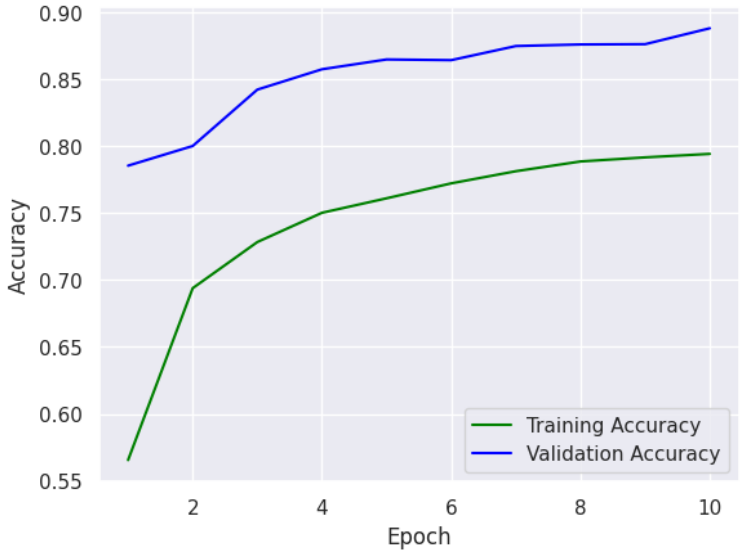

Figure 18: Training and Validation Accuracy Plot for AlexNet

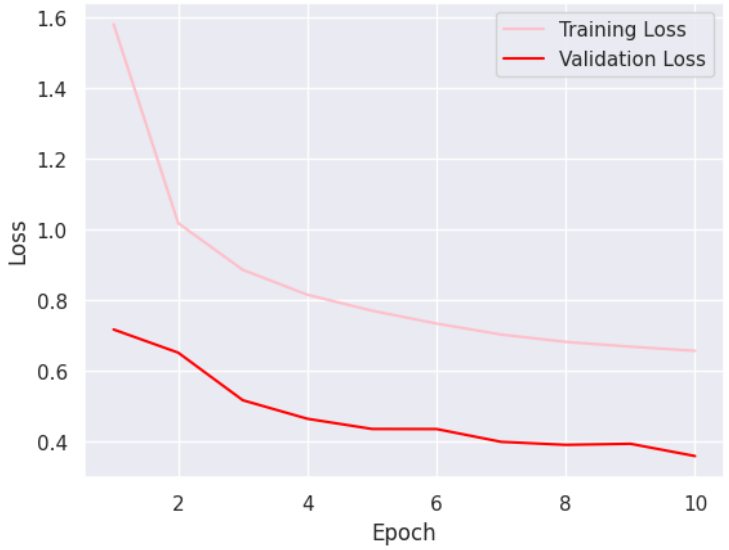

Figure 19: Training and Validation Loss Plot for AlexNet

Results for AlexNet are shown in figures 18 and 19. In figure 18, the training and validation accuracies of the AlexNet model are shown at each epoch, in which the last epoch show training and validation accuracies are *79.42 %* and *88.80%* respectively. In

figure 19, training and validation losses of the AlexNet model are shown at each epoch, in which the last epoch has the following training and validation loss values respectively: *0.66* and *0.36*.

No.	**Parameter**	**Value**
1.	Input Layer size	224 * 224 * 3
2.	Epochs	50
3.	Batch Size	32
4.	Image Augmentation	Rotation, width and height shift, flipping, shear, zoom, and preprocessing function
5.	Hardware resource	1 GPU

Table 9: Training Parameters for GoogleNet

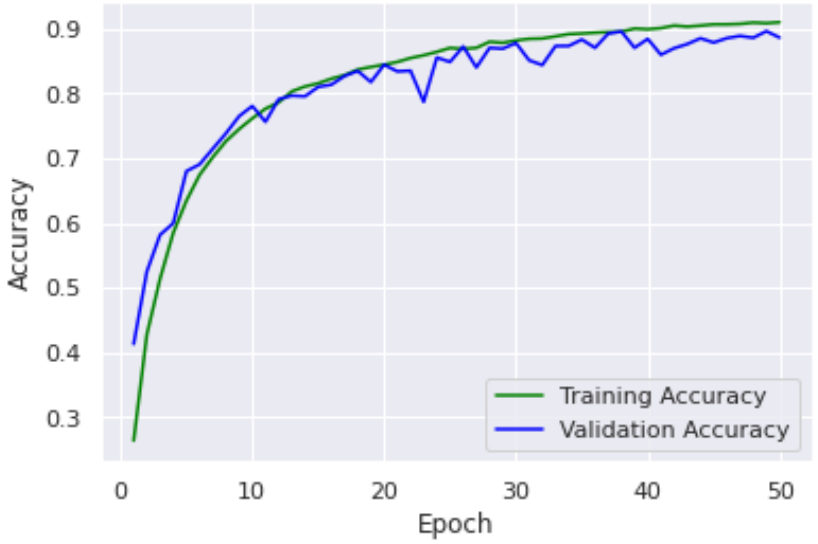

Figure 20: Training and Validation Accuracy Plot for GoogleNet

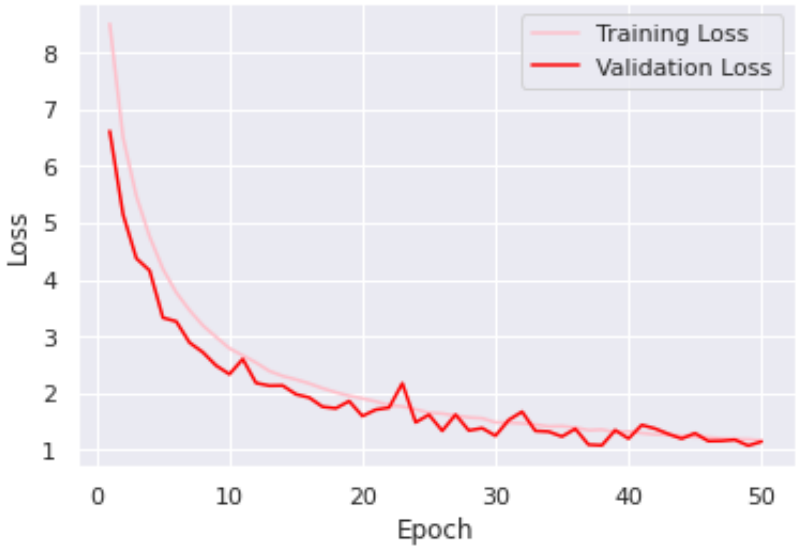

Figure 21: Training and Validation Loss Plot for GoogleNet

Results for GoogLeNet are shown in figures 20 and 21. In figure 20, the training and validation accuracies of the GoogLeNet model are shown at each epoch, in which the last epoch shows training and validation accuracies are *87.76 %* and *89.36%* respectively. In figure 21, training and validation losses of the GoogLeNet model are shown at each epoch, in which the last epoch has the following training and validation loss values respectively: *0.39* and *0.35*.

Results on PlantVillage Dataset:

Model Name	Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
VGG-16	95.25	93.59	4.07	5.91
VGG-19	82.61	85.75	5.24	4.63
AlexNet	79.42	88.80	0.66	0.36
GoogleNet	87.76	89.33	0.39	0.35

Table 10: Summary of Results of Models Trained on PlantVillage Dataset

Observing training and validation accuracies, VGG16 has the highest accuracy out of all the models, which is due to the dataset splitting approach used in which *80%* of the dataset was allocated to training and *20%* to testing the model. However, if we take a look at the loss values GoogLeNet clearly has the lowest training and validation loss values out of all the models, which means that while VGG16 has the highest accuracies, GoogLeNet is the most reliable model.

2. Training on Local Wheat_Crop Dataset:

No.	Parameter	Value
1.	Input Layer size	224 * 224 * 3
2.	Epochs	11
3.	Batch Size	32
4.	Image Augmentation	Rotation, width and height shift, flipping, shear, zoom, and preprocessing function
5.	Hardware resource	1 GPU

Table 11: Training Parameters for VGG-16 on Local Wheat_Crop Dataset

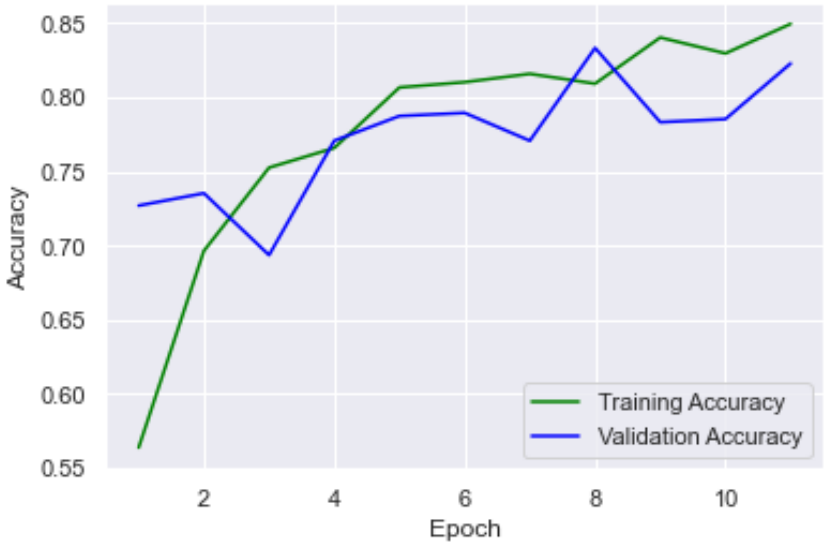

Figure 22: Training and Validation Accuracy Plot for VGG-16

Figure 23: Training and Validation Loss Plot for VGG-16

Results for VGG16 are shown in figures 22 and 23. In figure 22, training and validation accuracies of the VGG16 model are shown at each epoch, in which the last epoch show training and validation accuracies are *84.96 %* and *82.29%* respectively. In figure 23, training and validation losses of the VGG16 model are shown at each epoch, in which the last epoch has the following training and validation loss values respectively: *4.23* and *9.13*.

No.	**Parameter**	**Value**
1.	Input Layer size	224 * 224 * 3
2.	Epochs	12
3.	Batch Size	32
4.	Image Augmentation	Rotation, width and height shift, flipping, shear, zoom, and preprocessing function
5.	Hardware resource	1 GPU

Table 12: Training Parameters for VGG-19 on Local Wheat_Crop Dataset

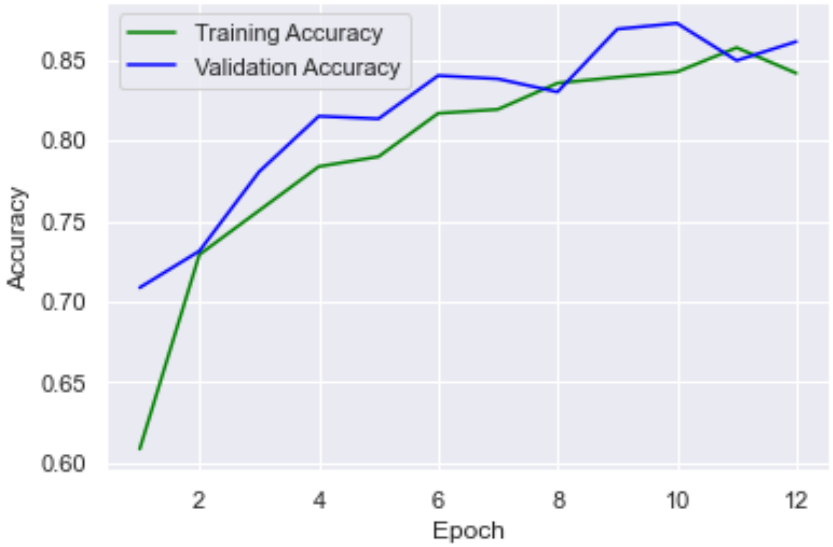

Figure 24: Training and Validation Accuracy Plot for VGG-19

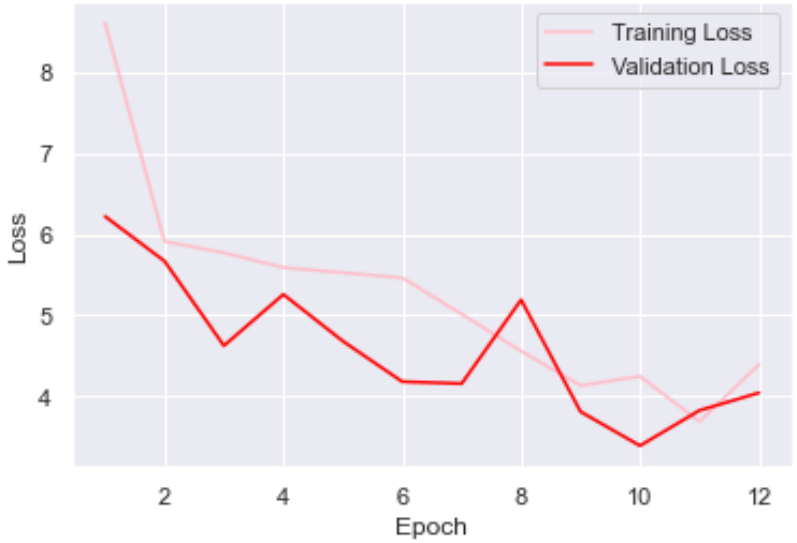

Figure 25: Training and Validation Loss Plot for VGG-19

Results for VGG19 are shown in figures 24 and 25. In figure 24, training and validation accuracies of the VGG19 model are shown at each epoch, in which the last epoch shows training and validation accuracies are *84.15 %* and *86.19%* respectively. In figure 25, training and validation losses of the VGG19 model are shown at each epoch, in which the last epoch has the following training and validation loss values respectively: *4.39* and *4.05*.

No.	**Parameter**	**Value**
1.	Input Layer size	227 * 227 * 3
2.	Epochs	60
3.	Batch Size	32
4.	Image Augmentation	Rotation, width and height shift, flipping, shear, zoom, and preprocessing function
5.	Hardware resource	1 GPU

Table 13: Training Parameters for AlexNet on Local Wheat_Crop Dataset

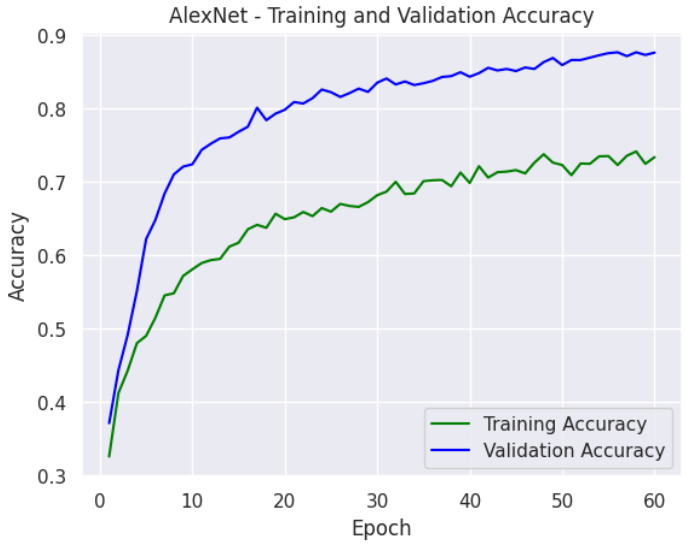

Figure 26: Training and Validation Accuracy Plot for AlexNet

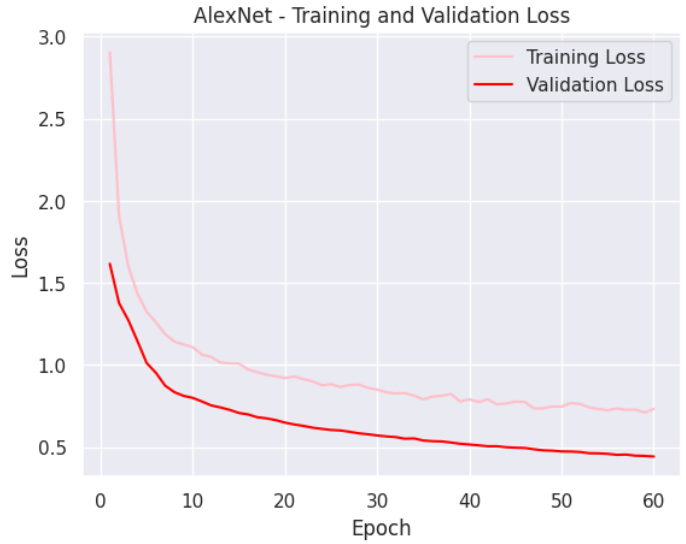

Figure 27: Training and Validation Loss Plot for AlexNet

Results for AlexNet are shown in figures 26 and 27. In figure 26, training and validation accuracies of the AlexNet model are shown at each epoch, in which the last epoch show training and validation accuracies are *73.34 %* and *87.58%* respectively. In figure 27, training and validation losses of the AlexNet model are shown at each epoch, in which the last epoch has the following training and validation loss values respectively: *0.73* and *0.44*.

No.	**Parameter**	**Value**
1.	Input Layer size	224 * 224 * 3
2.	Epochs	130
3.	Batch Size	32

4.	Image Augmentation	Rotation, width and height shift, flipping, shear, zoom, and preprocessing function
5.	Hardware resource	1 GPU

Table 14: Training Parameters for GoogLeNet on Local Wheat_Crop Dataset

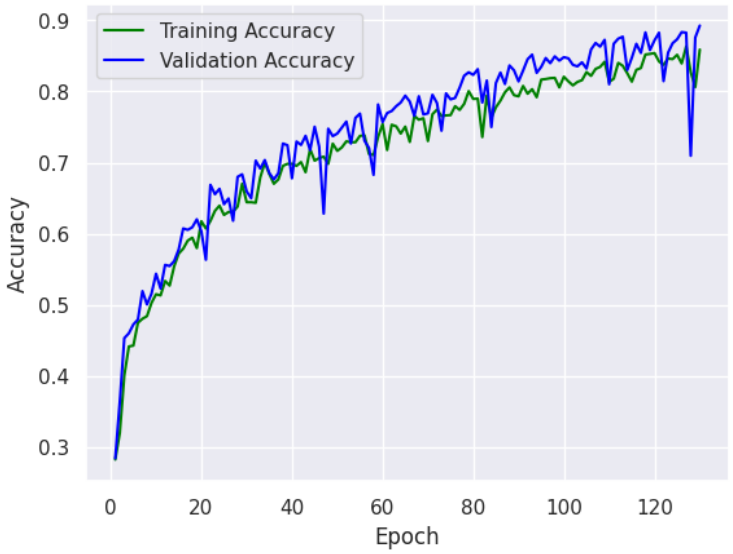

Figure 28: Training and Validation Accuracy Plot for GoogLeNet

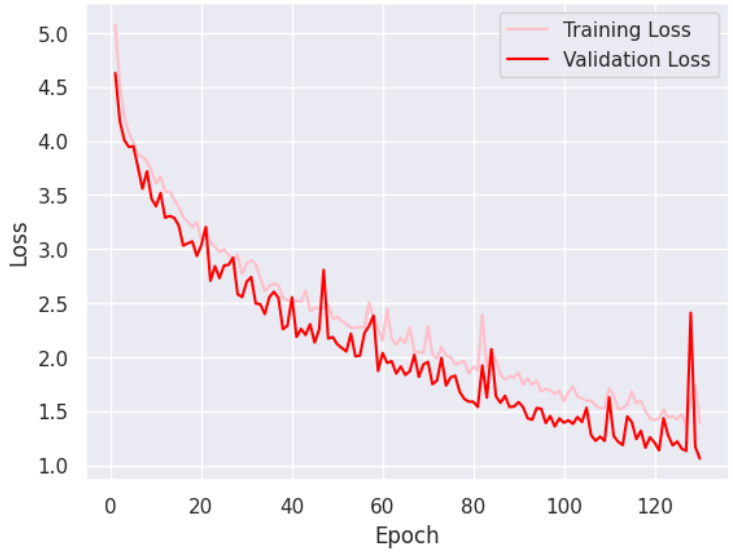

Figure 29: Training and Validation Loss Plot for GoogLeNet

Results for GoogLeNet are shown in figures 28 and 29. In figure 28, training and validation accuracies of the GoogLeNet model are shown at each epoch, in which the last epoch show training and validation accuracies are *85.48%* and *89.74%* respectively. In figure 29, training and validation losses of the GoogLeNet model are shown at each epoch, in which the last epoch has the following training and validation loss values respectively: *0.44* and *0.32*.

Results on Local Wheat_Crop Dataset:

Model Name	Training Accuracy (%)	Validation Accuracy (%)	Training Loss	Validation Loss
VGG-16	84.96	82.29	4.23	9.13
VGG-19	84.15	86.19	4.39	4.05
AlexNet	73.34	87.58	0.73	0.44
GoogleNet	**85.48**	**89.74**	**0.44**	**0.32**

Table 15: Summary of results of models trained on Local Wheat_Crop Dataset

Observing training and validation accuracies, GoogLeNet has the highest accuracy out of all the models. However, if we take a look at the losses GoogLeNet clearly has the lowest training and validation loss values out of all models, which means GoogLeNet is the most reliable model.

3. Anomaly Detection:

The Anomaly k-means cluster algorithm has been run on the private local wheat_crop dataset with 5 disease classes namely *Leaf Rust, Common Bunt, Fusarium Head Blight, Common Root Rot and Powdery Mildew* with a total of 2489 images. 6 clusters have been defined hence all the images from those classes have clustered into 6 parts as can be seen in Figure 30. It is necessary that we give some more information about this test dataset eg: class name etc

On adding abnormal data of 10 images into the test dataset totalling 2499 images, the anomaly detection algorithm was run again and the results in Figure 43 show that the anomalies have been grouped into cluster 0 and the other 5 classes images have clustered into the remaining 5 clusters.

This shows in Figure 31 that the anomaly algorithm has worked properly as it was able to detect anomalies/abnormal data into a separate cluster from the other normal data images in the dataset.

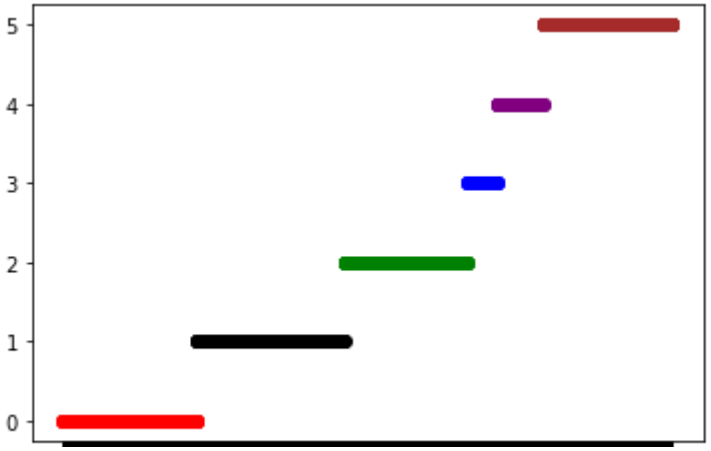

Figure 30: Cluster without anomaly

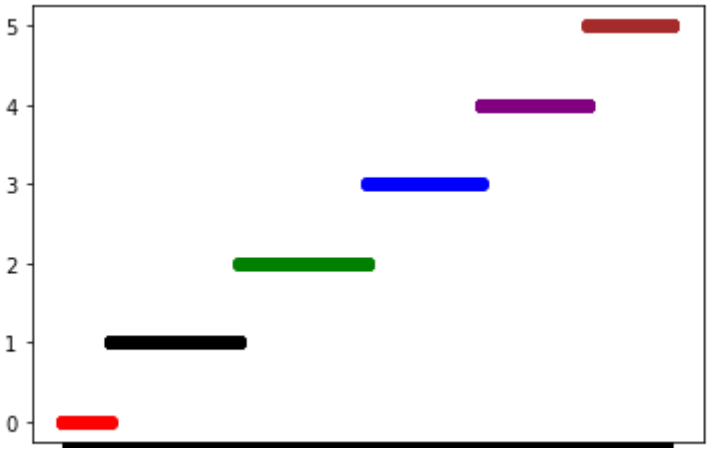

Figure 31: Cluster with anomaly

The datasets used in this second experiment for anomaly detection consists of three different sets of images, each containing five classes of plant diseases: *Common Bunt, Fusarium Head Blight, Common root rot, Leaf rust and Powdery mildew*. The

experiment has been divided using 2 different datasets, a normal dataset and an abnormal dataset.

The normal dataset consists of 3 testsets T1, T2 and T3 all of which contain 25 plant images from each class. The model used here is AlexNet which has already been trained on the full local wheat_crop dataset with an accuracy of 87.65%. The structure of the data in abnormal test datasets T1, T2, T3 is defined below:

T1:

- Common Bunt: 20 Common Bunt images, 5 car images
- Common root rot: 20 Common root rot images, 1 cat image, 4 dog images
- Fusarium Head Blight: 20 Fusarium Head Blight images, 5 cat images
- Leaf rust: 25 Leaf rust images
- Powdery mildew: 20 Powdery mildew images, 5 bicycle images

T2:

- Common Bunt: 15 Common Bunt images, 10 car images
- Common root rot: 15 Common root rot images, 1 cat image, 9 dog images
- Fusarium Head Blight: 15 Fusarium Head Blight images, 10 cat images
- Leaf rust: 25 Leaf rust images
- Powdery mildew: 15 Powdery mildew images, 10 bicycle images

T3:

- Common Bunt: 15 Common Bunt images, 10 car images
- Common root rot: 15 Common root rot images, 1 cat image, 9 dog images
- Fusarium Head Blight: 15 Fusarium Head Blight images, 10 cat images
- Leaf rust: 20 Leaf rust images, 5 bicycle images
- Powdery mildew: 15 Powdery mildew images, 10 bicycle images

The inclusion of unrelated images in the abnormal dataset challenges the machine learning model to identify patterns of the disease in the presence of extraneous information, which is often encountered in real-world scenarios.

Results:

Test Dataset	Normal Observations	Abnormal Observations
T1	95.12%	85.37%
T2	96.80%	76.00%
T3	90.40%	74.62%

Table 16: Normal and Abnormal Dataset trained with Alexnet

Overall, the experimental setup of abnormality detection in plant pathology involves training a machine learning model using a set of healthy and diseased plant images and then evaluating its ability to identify abnormal patterns in new, unseen images. The dataset used in this experiment contains both healthy and diseased plant images, along with a set of unrelated images to challenge the model's ability to distinguish disease patterns from extraneous information. This experimental setup enables researchers to develop accurate and effective methods for detecting plant diseases and mitigating their impact on crop yield and quality.

4. AI Model as a Service AWS Cloud based Simulation:

The need to transition towards cloud-based services in the agricultural sector has become increasingly important in recent years. With the help of high-speed wireless networks and cloud technologies, it is possible to enable a range of tasks such as crop analysis, prediction, and disease detection. With this vision in mind, we designed a service that can classify crop diseases using a deep learning model hosted in the cloud as a service to provide image classification for crop diseases to farmers in Pakistan. The decision to use AWS was based on the fact that it is a highly reliable and scalable cloud platform, with a wide range of services and resources that can be easily integrated to build complex systems. AWS EC2 instances provide a powerful and cost-effective way to host applications and services, while also offering flexible scalability options. Docker was used to create a containerized environment for the deep learning model and the Flask API, which allowed for easy deployment and management of the application. The Flask API is a RESTful web service that exposes the AI model for prediction and classification. Docker's ability to run on multiple platforms and operating systems, and its ability to package and ship applications, made it an ideal choice for creating a portable and scalable application that can be easily deployed and managed on AWS. The Docker container is hosted on an Amazon EC2 instance, which provides scalable computing capacity in the cloud. EC2 allows us to run multiple Docker containers, each handling incoming requests to the AI model. The EC2 instance can be configured to scale automatically to handle increased demand, ensuring that the service remains responsive even during peak usage periods.

To access the service, users can upload images to the service using a web interface or programmatically using any IOT system to the AWS API Gateway. The API Gateway provides a secure and scalable interface for the service that can handle millions of requests per day.

In summary, by using AWS, Docker, and the Flask API Python backend, we have developed a cloud-based machine learning service that can classify crop diseases. The service can be accessed by anyone with an internet connection, providing a valuable tool for crop disease management in Pakistan.

Figure 32 visualizes the holistic architecture of the web service and how a user would interact with it for disease classification by uploading images.

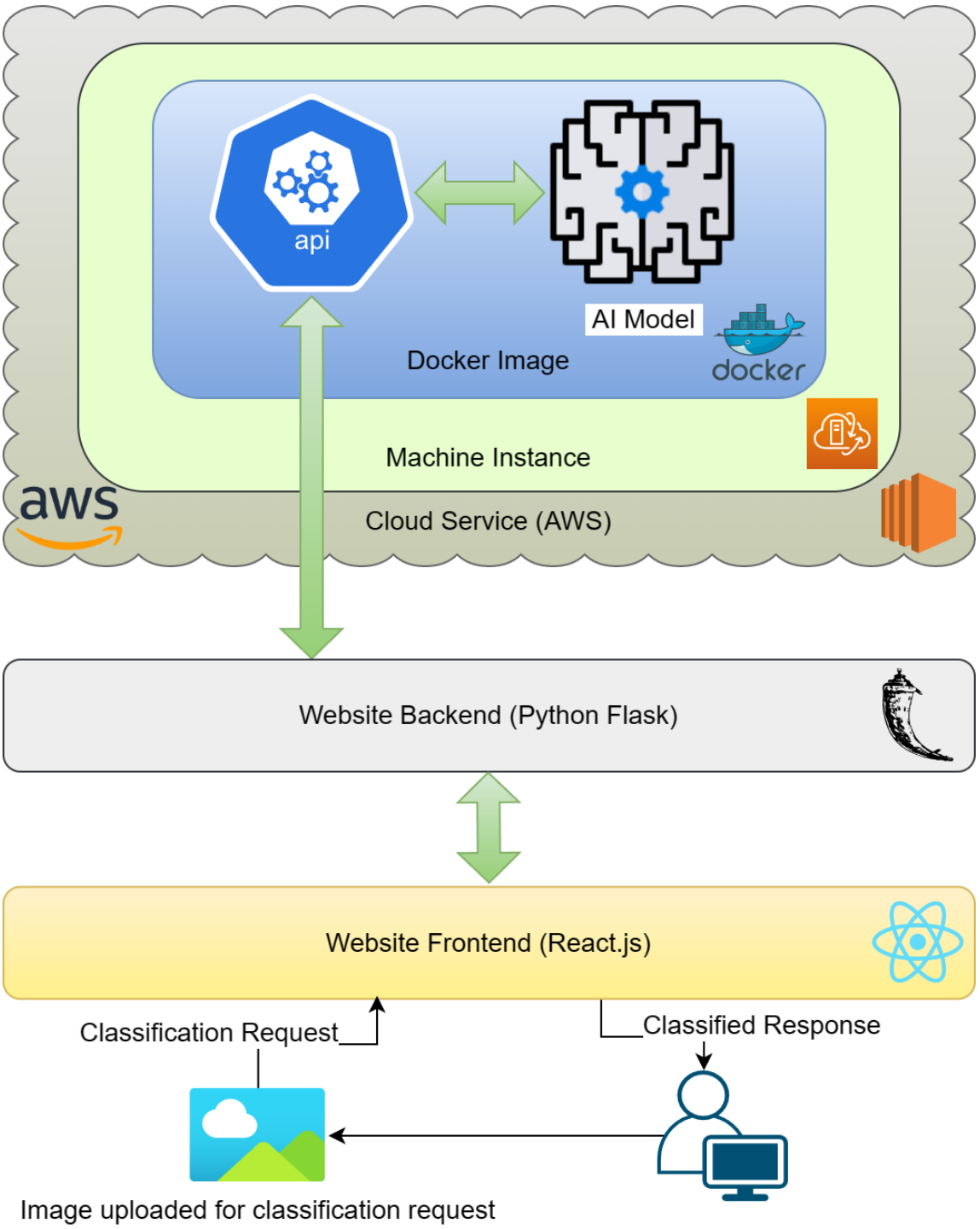

Figure 32: Web Service Architecture in Interaction with User.

4.1. Web service cloud setup:

4.1.1. Python Flask Server code:

```
import numpy as np
from flask import Flask, request
from tensorflow import keras
from PIL import Image
from keras.utils import img_to_array, load_img
from flask import jsonify
from flask_cors import CORS
import numpy as np
model = None
app = Flask(__name__)
CORS(app)

li = ['Common Bunt', 'Common Root Rot', 'Fusarium Head Blight', 'Leaf Rust',
'Powdery Mildew']

def load_model():
    global model
    model = keras.models.load_model('AlexNet_Model_Local_Dataset_Trained.h5')

@app.route('/')
def home_endpoint():
    return 'Hello World!'


@app.route('/predict', methods=['POST'])
def get_prediction():
    if request.method == 'POST':

        if 'image' not in request.files:
            return jsonify({'error': 'No file part'})
        file = request.files['image']
        img = Image.open(file.stream)
        img = img.resize((256, 256))
        img_arr = np.array(img)
        img_arr = np.expand_dims(img_arr, axis=0)
        img_arr = img_arr/255.0


        prediction = model.predict(img_arr)
        d = prediction.flatten()
        print(d)
        j = d.max()
        print(j)
        for index,item in enumerate(d):
```

```
            if item == j:
                class_name = li[index]
    return jsonify({'prediction': str(class_name)})
if __name__ == '__main__':
    load_model()
    app.run(host='0.0.0.0', port=80)
```

4.1.2. Dockerfile:

```
FROM python:3.6-slim
COPY ./service.py /deploy/
COPY ./AlexNet_Model_Local_Dataset_Trained.h5 /deploy/
WORKDIR /deploy/
RUN pip install numpy
RUN pip install flask
RUN pip install tensorflow==2.5.0
RUN pip install pillow
RUN pip install flask_cors
EXPOSE 80
ENTRYPOINT ["python", "service.py"]
```

4.1.3. Amazon EC2 Instance setup:

4.1.3.1. The first step is to make an AWS account to be able to access EC2 services. Once the account is created we can view the dashboard and access EC2.

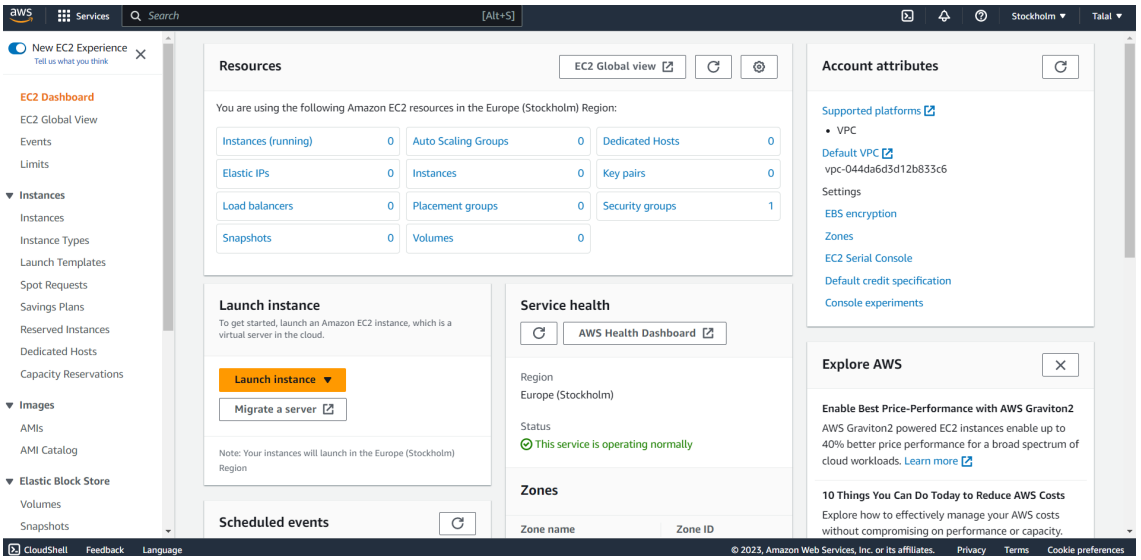

Figure 33: Amazon EC2 Instance Setup Step 1

4.1.3.2. The next step is to create a key pair to be used later. The key chosen is .pem since we plan to SSH into the ec2 instance terminal.

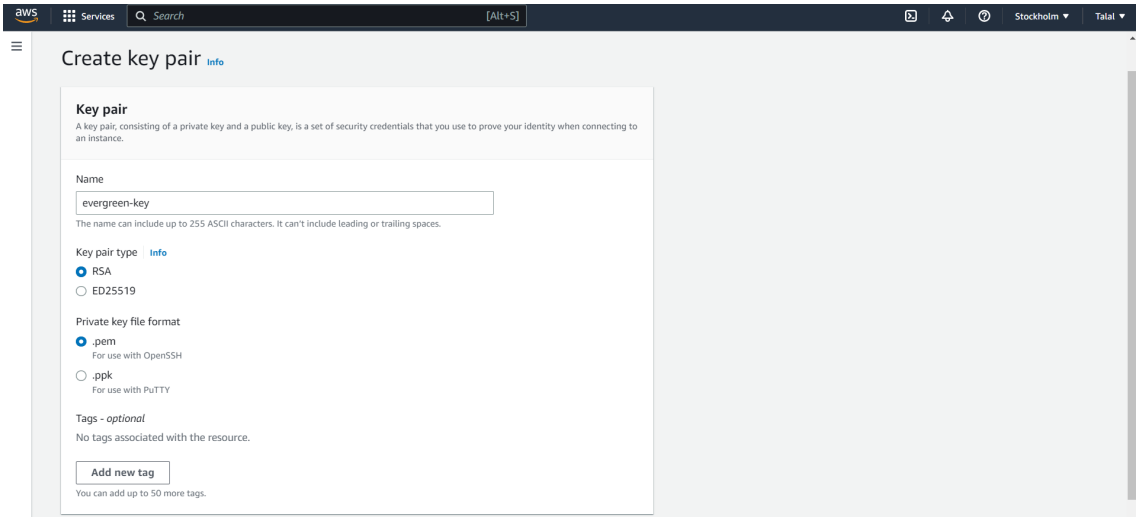

Figure 34: Amazon EC2 Instance Setup Step 2 Part 1

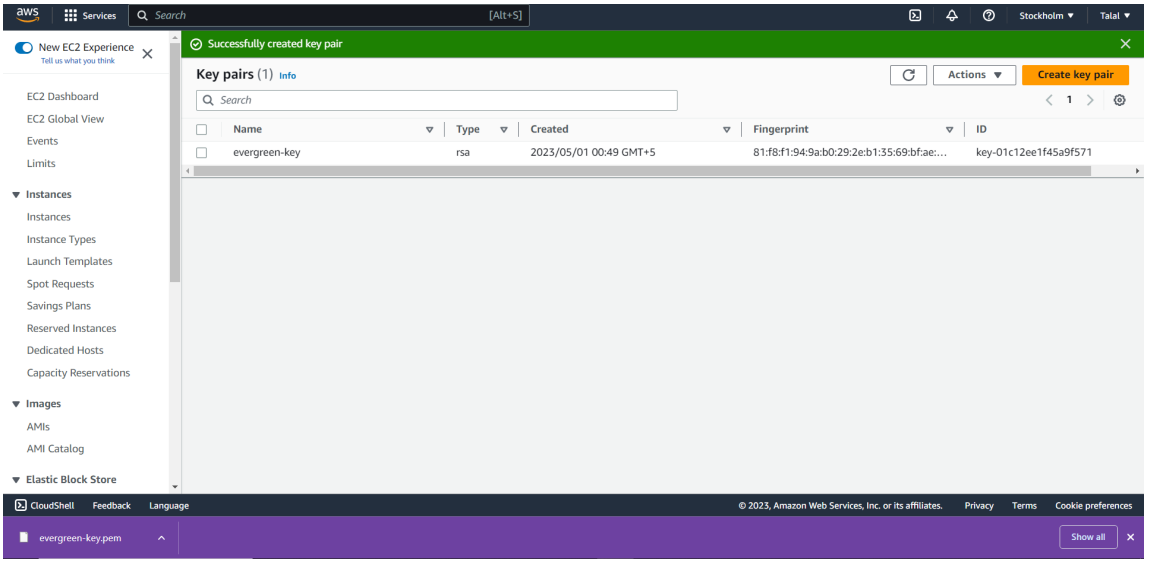

Figure 35: Amazon EC2 Instance Setup Step 2 Part 2

4.1.3.3. Making the key private.

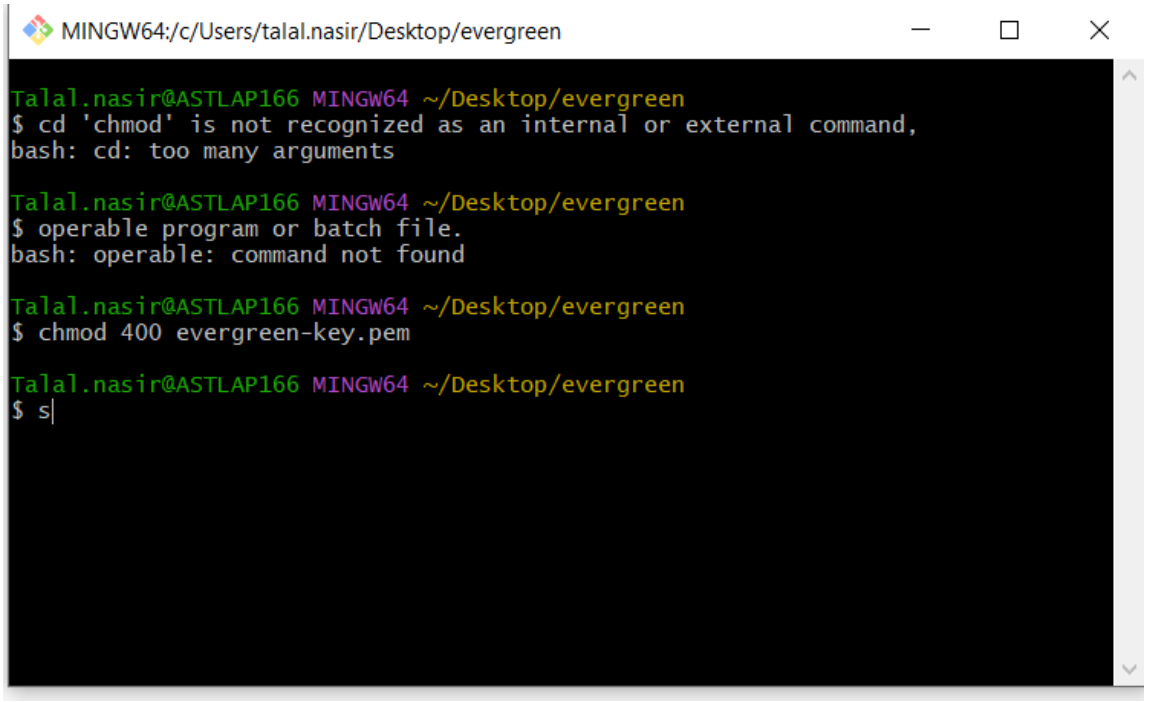

Figure 36: Amazon EC2 Instance Setup Step 3

4.1.3.4. Choosing the AMI instance.

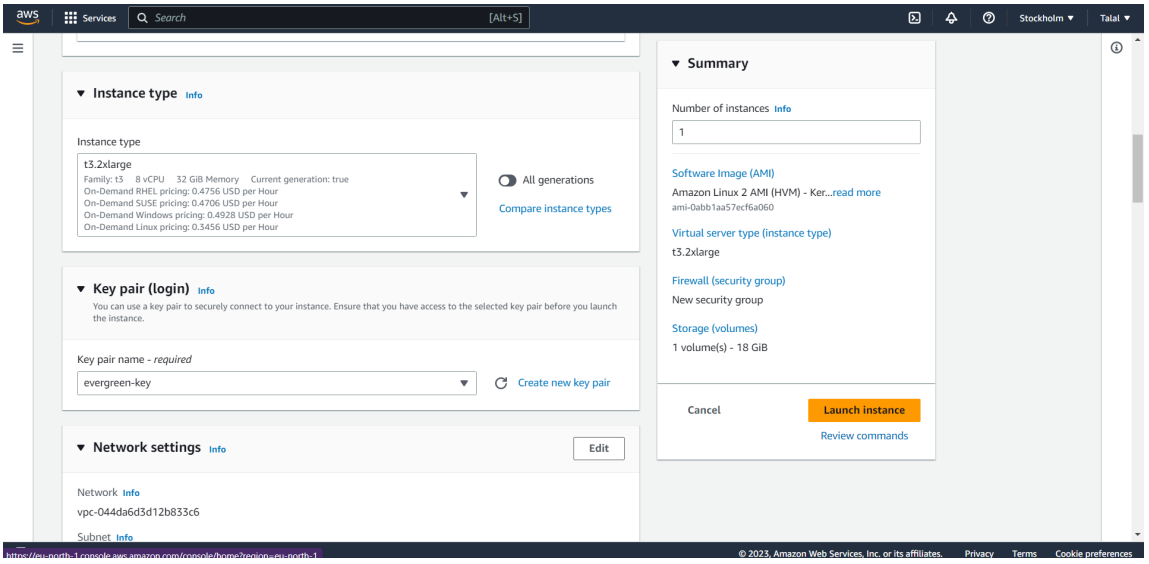

Figure 37: Amazon EC2 Instance Setup Step 4

4.1.3.5. Configuring HTTP and HTTPS requests to be allowed.

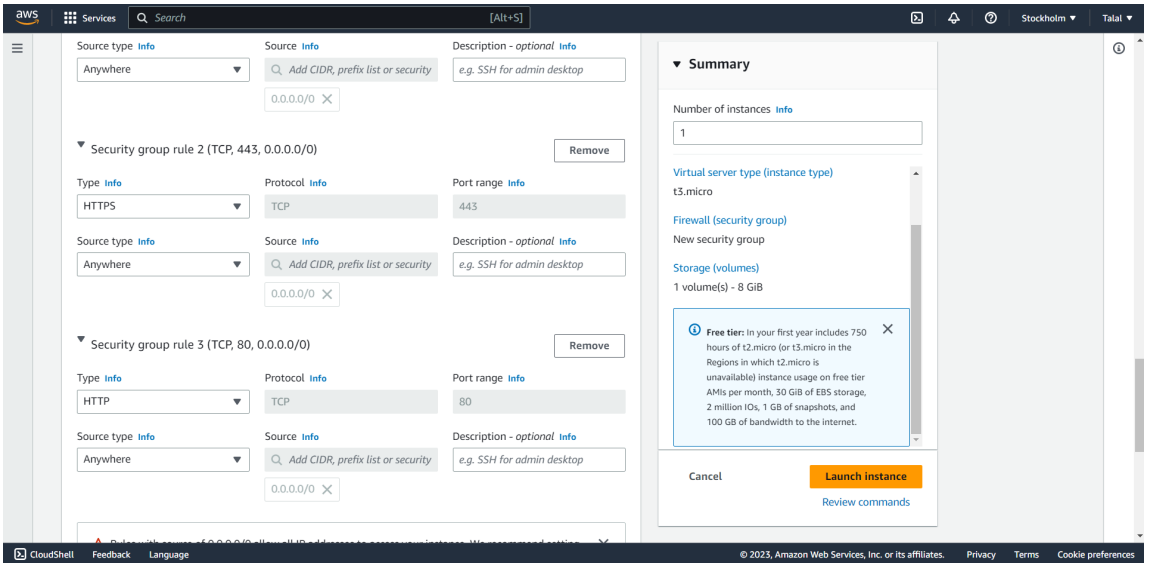

Figure 38: Amazon EC2 Instance Setup Step 5

4.1.3.6. Launch the instance.

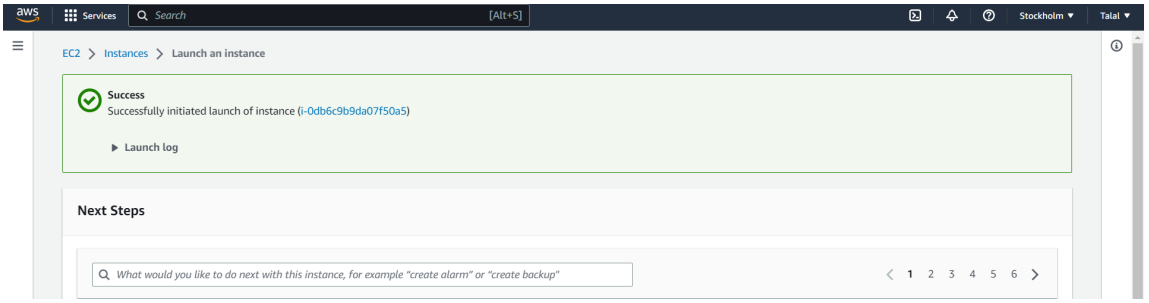

Figure 39: Amazon EC2 Instance Setup Step 6 Part 1

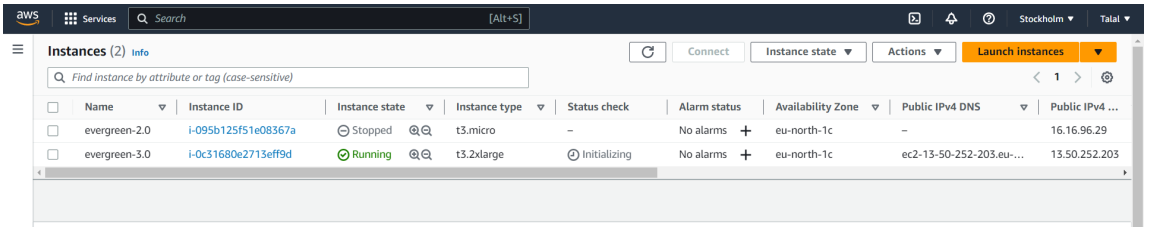

Figure 40: Amazon EC2 Instance Setup Step 6 Part 2

4.1.3.7. Connecting to the instance.

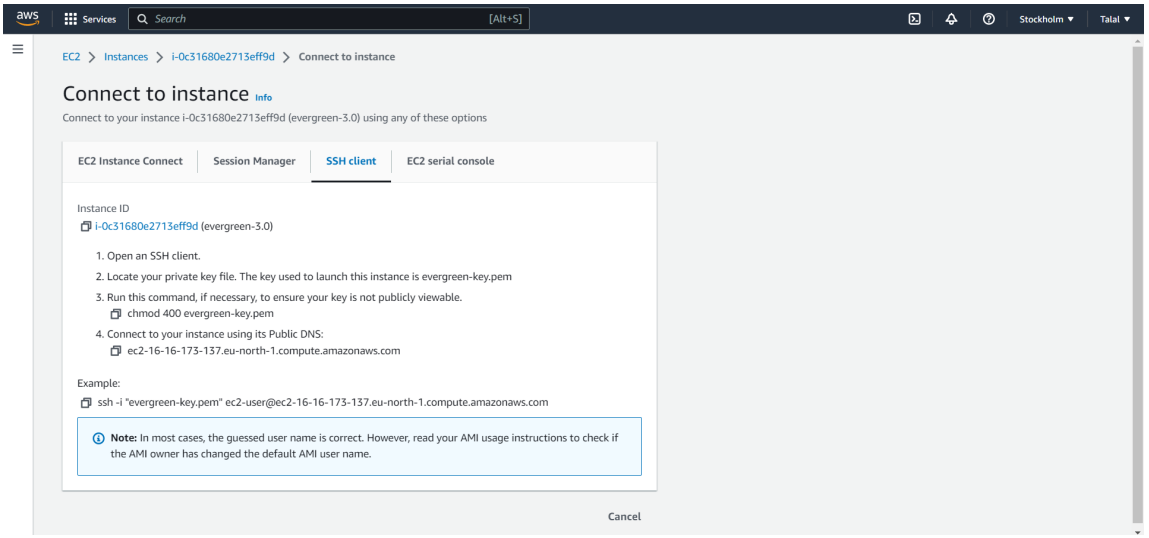

Figure 41: Amazon EC2 Instance Setup Step 7

4.1.3.8. SSH into the instance using CMD.

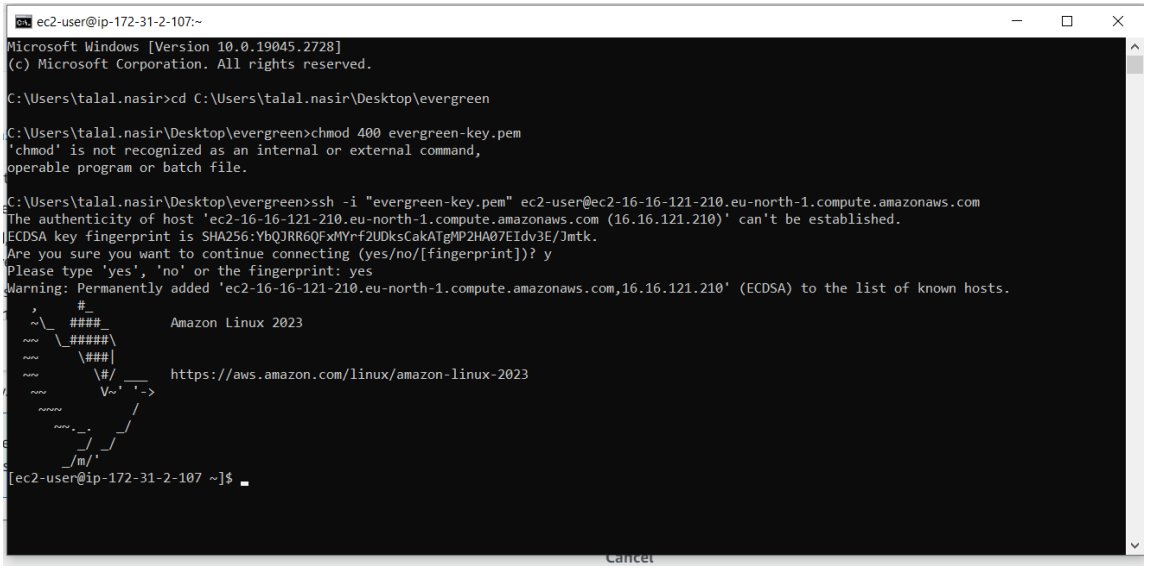

Figure 42: Amazon EC2 Instance Setup Step 8

4.1.3.9. Installing docker.

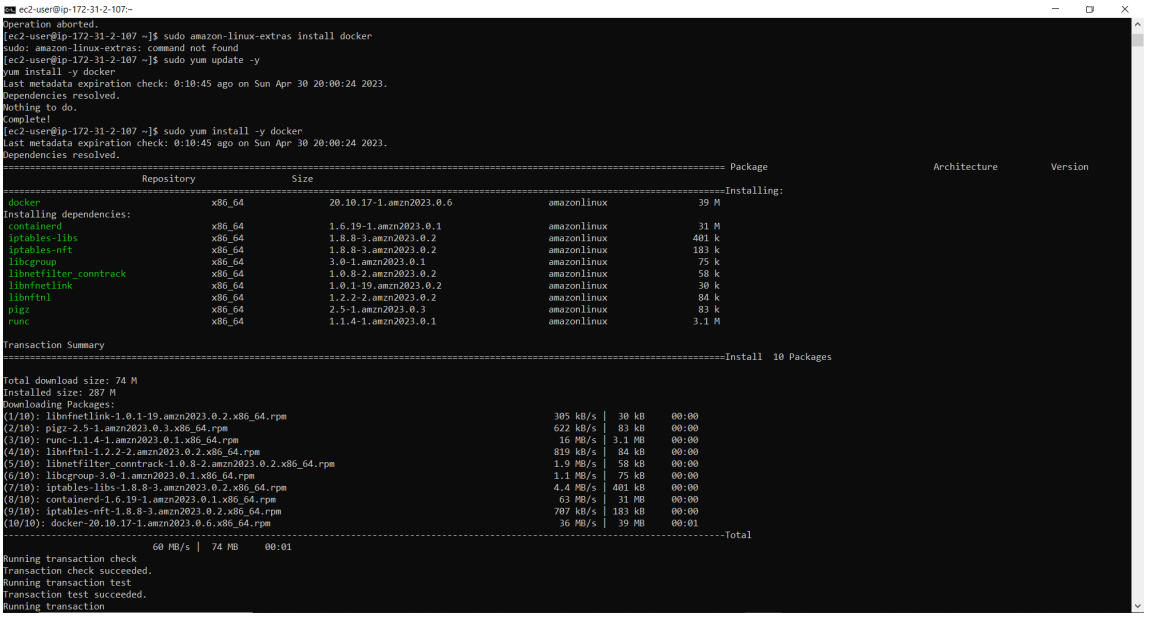

Figure 43: Amazon EC2 Instance Setup Step 9

4.1.3.10. Verifying the docker installation.

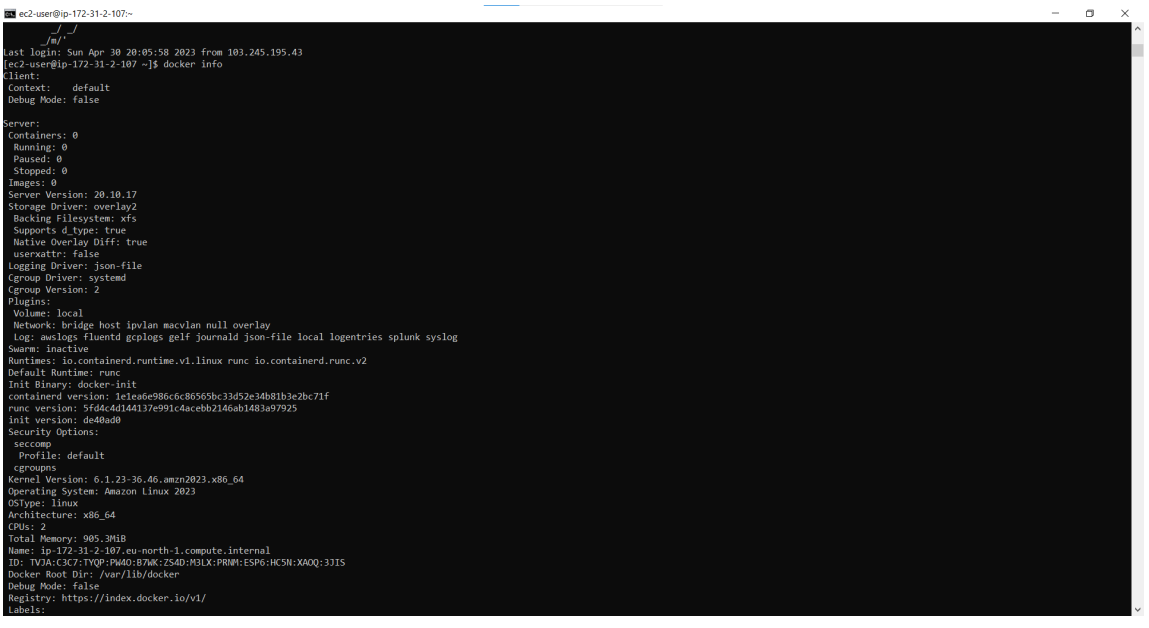

Figure 44: Amazon EC2 Instance Setup Step 10

4.1.3.11. Copying the server file, dockerfile and model to the EC2 instance directory.

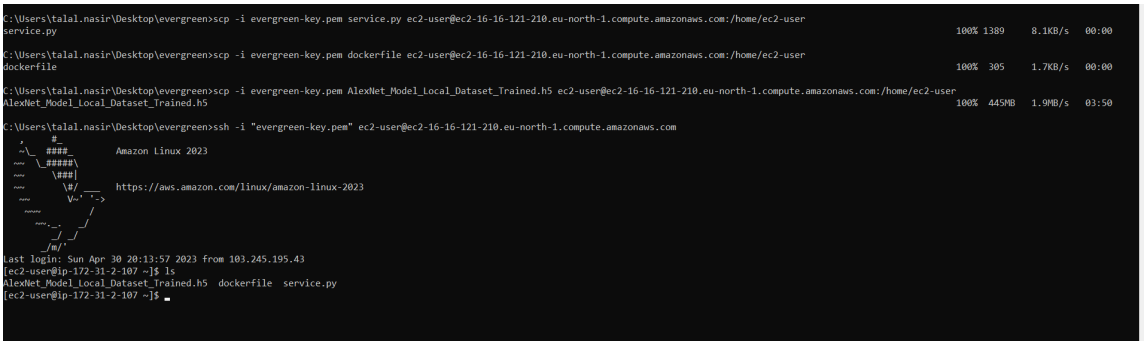

Figure 45: Amazon EC2 Instance Setup Step 11

4.1.3.12. Creating the docker image.

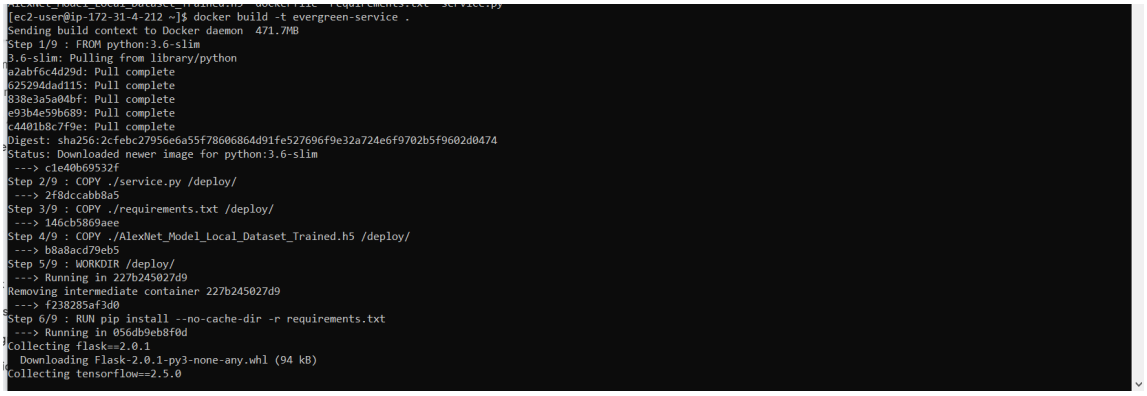

Figure 46: Amazon EC2 Instance Setup Step 12

4.1.3.13. Launching a container with the service.

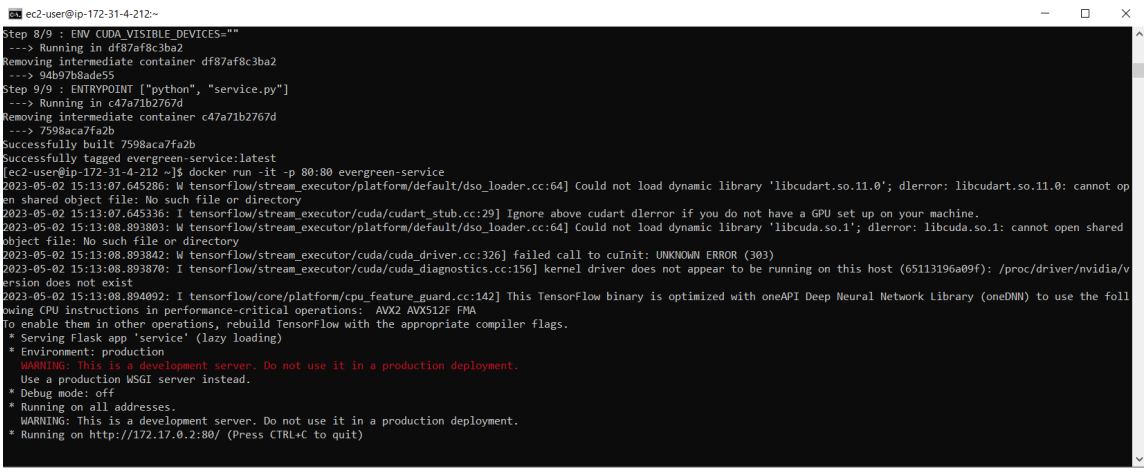

Figure 47: Amazon EC2 Instance Setup Step 13

4.1.4. Uploading an image to predict on a website/client.

4.1.4.1. Uploading an image to predict on a website/client.

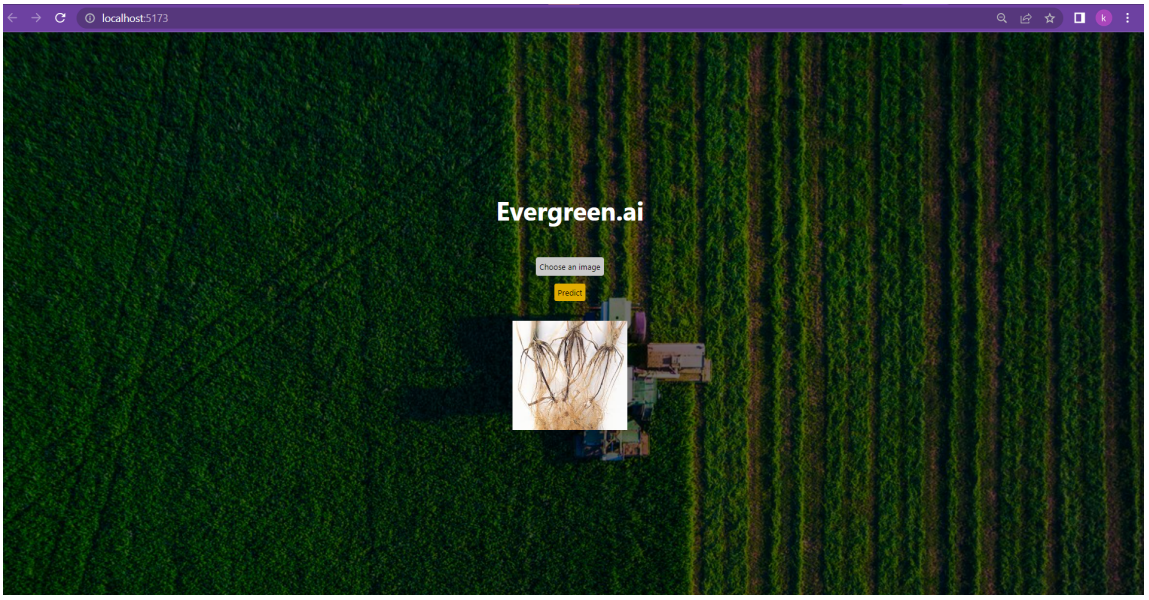

Figure 48: Amazon EC2 Instance Setup Step 14

4.1.4.2. Use the AWS API url.

```
try {
  const response = await fetch('http://ec2-13-50-252-203.eu-north-1.compute.amazonaws.com:80/predict', {
    method: 'POST',
    body: formData,
  });
```

Figure 49: Amazon EC2 Instance Setup Step 15

4.1.4.3. EC2 instance receives the post request on the docker container and returns the prediction.

```
ec2-user@ip-172-31-4-212:~
Removing intermediate container df87af8c3ba2
 ---> 90b97b8ade55
Step 9/9 : ENTRYPOINT ["python", "service.py"]
 ---> Running in c47a71b2767d
Removing intermediate container c47a71b2767d
 ---> 7598aca7fa2b
Successfully built 7598aca7fa2b
Successfully tagged evergreen-service:latest
[ec2-user@ip-172-31-4-212 ~]$ docker run -it -p 80:80 evergreen-service
2023-05-02 15:13:07.645286: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: libcudart.so.11.0: cannot open shared object file: No such
 file or directory
2023-05-02 15:13:07.645336: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on your machine.
2023-05-02 15:13:08.893803: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object file: No such file or d
irectory
2023-05-02 15:13:08.893842: W tensorflow/stream_executor/cuda/cuda_driver.cc:326] failed call to cuInit: UNKNOWN ERROR (303)
2023-05-02 15:13:08.893870: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear to be running on this host (65113196a09f): /proc/driver/nvidia/version does not exist
2023-05-02 15:13:08.894092: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in perf
ormance-critical operations:  AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app 'service' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://172.17.0.2:80/ (Press CTRL+C to quit)
103.245.195.43 - - [02/May/2023 15:20:05] "POST /predict HTTP/1.1" 200 -
103.245.195.43 - - [02/May/2023 15:20:16] "POST /predict HTTP/1.1" 200 -
2023-05-02 15:21:04.828748: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:176] None of the MLIR Optimization Passes are enabled (registered 2)
2023-05-02 15:21:04.829346: I tensorflow/core/platform/profile_utils/cpu_utils.cc:114] CPU Frequency: 2499995000 Hz
[4.2306736e-02 7.5035751e-01 1.8470806e-01 5.3645985e-04 2.2091355e-02]
0.7503575
103.245.195.43 - - [02/May/2023 15:21:05] "POST /predict HTTP/1.1" 200 -
```

Figure 50: Amazon EC2 Instance Setup Step 16

4.1.4.4. Displaying the disease prediction.

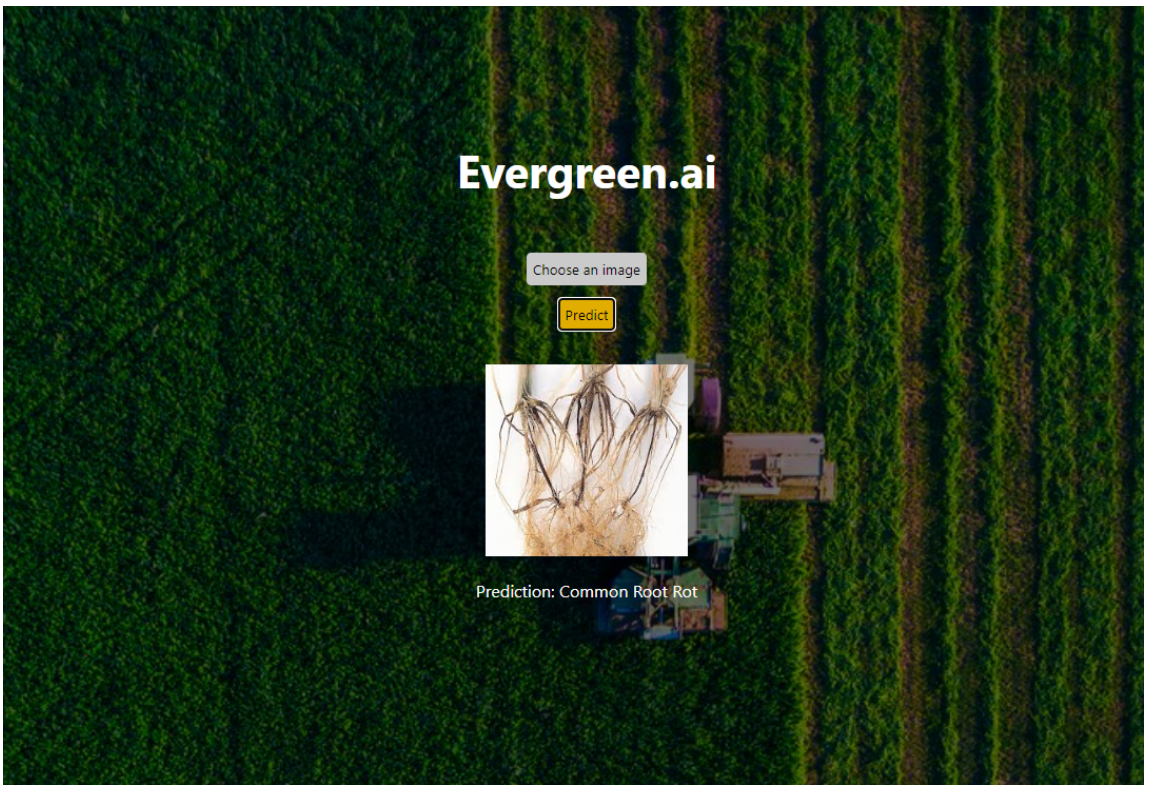

Figure 51: Amazon EC2 Instance Setup Step 17

4.1.4.5. Figure 64 is another example where we have uploaded a different image sending a request for classification and in return we received a response of classification from the cloud hosted AI model using flask API.

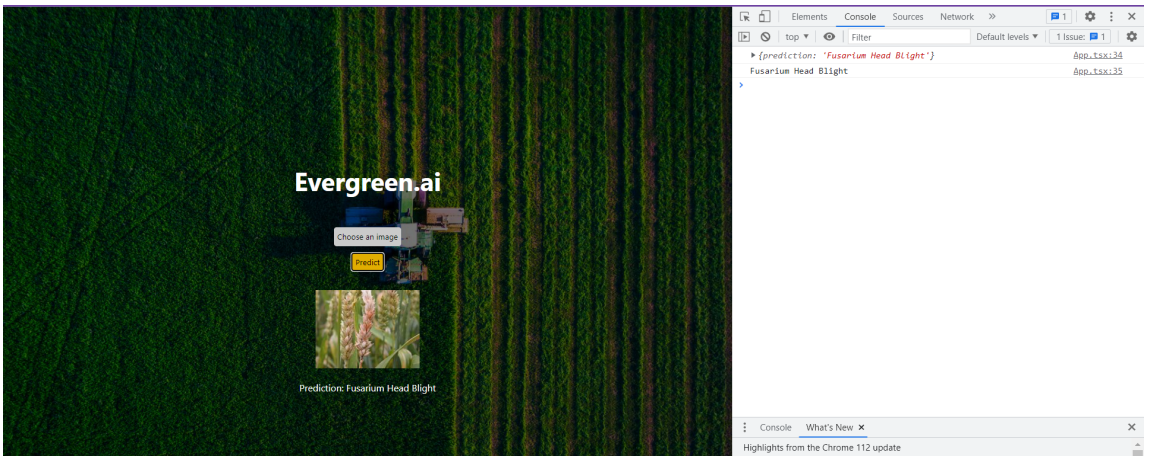

Figure 52: Amazon EC2 Instance Setup Step 18

Conclusion and Future Work

In this research, our main purpose was to provide effective plant disease detection for local pakistani crops. To achieve this goal, we trained deep learning models on both the Plant Village dataset first, as an experiment, and then a local wheat_crop dataset.In the course of our study, we trained four well-known deep learning models: VGG16, VGG19, AlexNet, and GoogleNet on the two datasets to evaluate their performances. Based on our results, GoogLeNet emerged as the best model for the Plant Village dataset due to its low loss, making it more reliable than the other models. Although VGG16 had the highest accuracy, its extremely high loss made it an unreliable choice. AlexNet, on the other hand, had a higher accuracy than VGG19 and a lower loss, making it a better option for the Plant Village dataset. However, GoogLeNet outperformed all the other models in terms of accuracy and loss.

In terms of the results on local wheat_crop dataset accuracy, VGG16 performed slightly better than VGG19, but had a higher loss, making VGG19 the better option. But as we trained on the AlexNet and GoogleNet models, both of these models had higher accuracies and less loss than the VGG Net models, with GoogleNet having the highest accuracy and the lowest loss making it the most suitable option out of the bunch.

Furthermore, we performed anomaly detection using normal and abnormal datasets, which allowed us to identify the negative impact of abnormal images on the performance of our trained models. We observed that when abnormal images were present, the accuracy of the trained model decreased significantly.

Moving forward, we suggest an approach for “Incremental Learning”, which involves continuous improvement through updating models using new data and uses anomaly detection to remove the abnormal images, retraining the models, and saving the best models, we can improve performance of the models significantly and thus enhance the performance of our models. This will help in creating a continuously improving system.

Finally, the use of AI in agriculture has the potential to revolutionize the industry. We envision the evolution of agriculture through the integration of AI and IoT technologies. With the help of drones and sensors, we can collect vast amounts of data on plant growth and crop yields, which can be analyzed using AI algorithms to optimize crop management and maximize agricultural production. Additionally, AI can aid in detection of plant diseases at early stages. This may avert diseases and minimize the loss of crops. Overall, there is immense potential for AI in agriculture, and we can expect to see futuristic visions of AI-driven smart agricultural systems in the near future.

References

[1] Soichiro Torai, Shinichi Chiyoda, and Kenichi Ohara. Application of ai technology to smart agriculture: Detection of plant diseases. In *2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 1514–1519, 2020.

[2] Tejas Khare and Anuradha Phadke. Automated crop field surveillance using computer vision. 12 2020.

[3] Faisal Karim Shaikh, Mohsin Ali Memon, Naeem Ahmed Mahoto, Sherali Zeadally, and Jamel Nebhen. Artificial intelligence best practices in smart agriculture. *IEEE Micro*, 42(1):17–24, 2022.

[4] Sameer Qazi, Bilal A. Khawaja, and Qazi Umar Farooq. Iot-equipped and ai-enabled next generation smart agriculture: A critical review, current challenges and future trends. *IEEE Access*, 10:21219–21235, 2022.

[5] Sankarganesh Egambaram. Pesticide resistance and its management in insect pests. 8:82–83, 11 2019.

[6] Chirag Sharma. Self driving car using deep learning technique. *Inter- national Journal of Engineering Research and*, V9, 06 2020.

[7] Sharada P. Mohanty, David P. Hughes, and Marcel Salathe´. Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, 7, 2016.

[8] Da-Wei Zhou, Qi-Wei Wang, Zhi-Hong Qi, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Deep class-incremental learning: A survey, 2023.

[9] Sk Hassan, Arnab Maji, Michał Jasin´ski, Zbigniew Leonowicz, and Elz˙bieta Jasin´ska. Identification of plant-leaf diseases using cnn and transfer-learning approach. *Electronics*, 10:1388, 06 2021.

[10] Junde Chen, Jinxiu Chen, Defu Zhang, Yuandong Sun, and Y.A. Nanehkaran. Using deep transfer learning for image-based plant disease identification. *Computers and Electronics in Agriculture*, 173:105393, 2020.

[11] Saurabh Sharma Sukhwinder Kaur. Plant disease detection using deep transfer learning. *Journal of Positive School Psychology*, 2022.

[12] Anwar Abdullah Alatawi, Shahd Maadi Alomani, Najd Ibrahim Alhawiti and Muhammad Ayaz, “Plant Disease Detection using AI based VGG-16 Model” International Journal of Advanced Computer Science and Applications(IJACSA), 13(4), 2022. http://dx.doi.org/10.14569/IJACSA.2022.0130484

[13] Prakhar Bansal, Rahul Kumar, and Somesh Kumar. Disease detection in apple leaves using deep convolutional neural network. *Agriculture*, 11(7), 2021.

[14] Plant pathology 2020 - fgvc7.

[15] V V Srinidhi, Apoorva Sahay, and K. Deeba. Plant pathology disease detection in apple leaves using deep convolutional neural networks : Apple leaves disease detection using efficientnet and densenet. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1119–1127, 2021.

[16] Muhammad Shoaib, Tariq Hussain, Babar Shah, Ihsan Ullah, Sayyed Mudassar Shah, Farman Ali, and Sang Hyun Park. Deep learning-based segmentation and classification of leaf images for de- tection of tomato plant disease. *Frontiers in Plant Science*, 13, 2022.

[17] V. K. Shrivastava, M. K. Pradhan, S. Minz, and M. P. Thakur. Rice Plant Disease Classification Using Transfer Learning of Deep Convolution Neural Network. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 423:631–635, July 2019.

[18] Xulang Guan. A novel method of plant leaf disease detection based on deep learning and convolutional neural network. In *2021 6th In- ternational Conference on Intelligent Computing and Signal Processing (ICSP)*, pages 816–819, 2021.

[19] Rafael Faria Caldeira, Wesley Esdras Santiago, and Barbara Teruel. Identification of cotton leaf lesions using deep learning techniques. *Sensors (Basel, Switzerland)*, May 2021.

[20] P. Vishnu Raja, K. Sangeetha, Ninisa B A, Samiksha M, and Sanjutha S S. Convolutional neural networks based classification and detection of plant disease. In *2022 6th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1484–1488, 2022.

[21] Akshai KP and J. Anitha. Plant disease classification using deep learning. In *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, pages 407–411, 2021.

[22] A. Lakshmanarao, M. Raja Babu, and T. Srinivasa Ravi Kiran. Plant disease prediction and classification using deep learning convnets. pages 1–6, 2021.

[23] Konstantinos P. Ferentinos. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture*, 145:311–318, 2018.

[24] Geetharamani G. and Arun Pandian J. Identification of plant leaf diseases using a nine-layer deep convolutional neural network. *Computers Electrical Engineering*, 76:323–338, 06 2019.

[25] J. Arun Pandian, V. Dhilip Kumar, Oana Geman, Mihaela Hnatiuc, Muhammad Arif, and K. Kanchanadevi. Plant disease detection using deep convolutional neural network. *Applied Sciences*, 12(14), 2022.

[26] Sachin Jadhav, Vishwanath Udupi, and Sanjay Patil. Identification of plant diseases using convolutional neural networks. *International Journal of Information Technology*, 13, 02 2020.

[27] Achilles D. Boursianis, Maria S. Papadopoulou, Panagiotis Diaman- toulakis, Aglaia Liopa-Tsakalidi, Pantelis Barouchas, George Salahas, George Karagiannidis, Shaohua Wan, and Sotirios K. Goudos. Internet of things (iot) and agricultural unmanned aerial vehicles (uavs) in smart farming: A comprehensive review. *Internet of Things*, 18:100187, 2022.

[28] E. Raymond Hunt, Craig S. T. Daughtry, Steven B. Mirsky, and W. Dean Hively. Remote sensing with simulated unmanned aircraft imagery for precision agriculture applications. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(11):4566–4571, 2014.

[29] A. Subeesh and C.R. Mehta. Automation and digitization of agriculture using artificial intelligence and internet of things. *Artificial Intelligence in Agriculture*, 5:278–291, 2021.

[30] Salima Ouadfel, Wafa Mousser, Ismail Ghoul, and Abdelmalik Taleb- Ahmed. 9. Incremental deep learning model for plant leaf diseases detection. In *Artificial Neural Networks for Renewable Energy Systems and Real-World Applications*, pages 207–222. Elsevier, September 2022.

[31] https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset.

[32] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[34] Manali Shaha and Meenakshi Pawar. Transfer learning for image classification. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 656–660, 2018.

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

[37] Jerry Wei. Alexnet: The architecture that challenged cnns. 2021.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, may 2017.

[39] Onkar Saxena, Shikha Agrawal, and Sanjay Silakari. Disease detection in plant leaves using deep learning models: Alexnet and googlenet.In *2021 IEEE International Conference on Technology, Research, and Innovation for Betterment of Society (TRIBES)*, pages 1–6, 2021.

[40] Mohamed Elgendy. *Deep Learning for Vision Systems*. Manning Publications Co., 2020.

[41] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. pages 1–9, 2015.

[42] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009.

[43] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., USA, 1988.

[44] Kenneth Leaung. How to easily draw neural network architecture diagrams, Sep 2022.

[45] Breast cancer screening using convolutional neural network and follow-up digital mammography - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/llustration-of-the-network-architecture-of-VGG-19-model-conv-means-convolution-FC-means_fig2_325137356 [accessed 3 May, 2023]