MDPI

Article

Real-time Plant Health Assessment via Implementing Cloud-based Scalable Transfer Learning on AWS DeepLens

Asim Khan [1,*], Umair Nawaz [2], Anwaar Ulhaq [1,3] and Randall W. Robinson [4]

[1] The Institute for Sustainable Industries and Liveable Cities (ISILC), Victoria University, Melbourne; asim.khan@vu.edu.au
[2] Namal Institute, Mainwali; umair.nawaz@namal.edu.pk
[3] Machine Vision and Digital Health Research Group,Charles Sturt University, NSW, Australia; aulhaq@csu.edu.au
[4] Applied Ecology Research Group, Victoria University, Melbourne Australia; randall.robinson@vu.edu.au
* Correspondence: asim.khan@vu.edu.au

Version September 12, 2020 submitted to Remote Sens.

Abstract: In the Agriculture sector, control of plant leaf diseases is crucial as it influences the quality and production of plant species with an impact on the economy of any country. Therefore, automated identification and classification of plant leaf disease at an early stage is essential to reduce economic loss and to conserve the specific species. Previously, to detect and classify plant leaf disease, various Machine Learning models have been proposed; however, they lack usability due to hardware incompatibility, limited scalability and inefficiency in practical usage. Our proposed DeepLens Classification and Detection Model (DCDM) approach deal with such limitations by introducing automated detection and classification of the leaf diseases in fruits (apple, grapes, peach and strawberry) and vegetables (potato and tomato) via scalable transfer learning on AWS SageMaker and importing it on AWS DeepLens for real-time practical usability. Cloud integration provides scalability and ubiquitous access to our approach. Our experiments on extensive image data set of healthy and unhealthy leaves of fruits and vegetables showed an accuracy of 98.78% with a real-time diagnosis of plant leaves diseases. We used forty thousand images for the training of deep learning model and then evaluated it on ten thousand images. The process of testing an image for disease diagnosis and classification using AWS DeepLens on average took 0.349s, providing disease information to the user in less than a second.

Keywords: Plant Diseases; Modern Agriculture; Plant Health; AWS DeepLens; SageMaker; Machine Learning; Deep Learning

1. Introduction

Plant disease has a destructive impact on quantitative and qualitative production [27], leading to a striking blow to producers, traders and consumers. In a US-based study conducted by the U.G.A. Centre for Agribusiness and Economic Development [1] discovered a 14.1% relative disease loss across all crops. A summary of losses due to plant disease included in the 2017 Georgia Farm Gate Value Report (AR-18-01) [1] published by University of Georgia Extension.

Traditionally farmers detect and diagnose plant diseases through their observations and rely upon the opinions of local experts and their past experiences. An expert can decide whether a plant is healthy or not [8]. If a plant is found unhealthy, noticeable symptoms on its leaves and fruits are observed and reported. It is hard to correctly diagnose specific diseases even for agronomists and plant pathologists, resulting in incorrect decisions [11]. Practical plant health assessment and an early

diseases diagnosis can improve product quality and prevent production loss. Early detection and classification of crop disease are significant to secure the specific species production [24]. Various research studies have found that early detection of plant diseases is crucial as over the period, diseases start affecting the growth of their species, and their symptoms appear on the leaves [26]. When a plant got infected by a specific disease, then significant symptoms are shown on the leaves, which help in the identification and classification of that particular disease [6]. Thus controlling and assessment of diseases outspread becomes essential [37]. As in peach plant, for instance, the decayed area is small and looks similar in appearance to neighbouring healthy tissue at an early stage; therefore, it is tough to detect diseases [1].

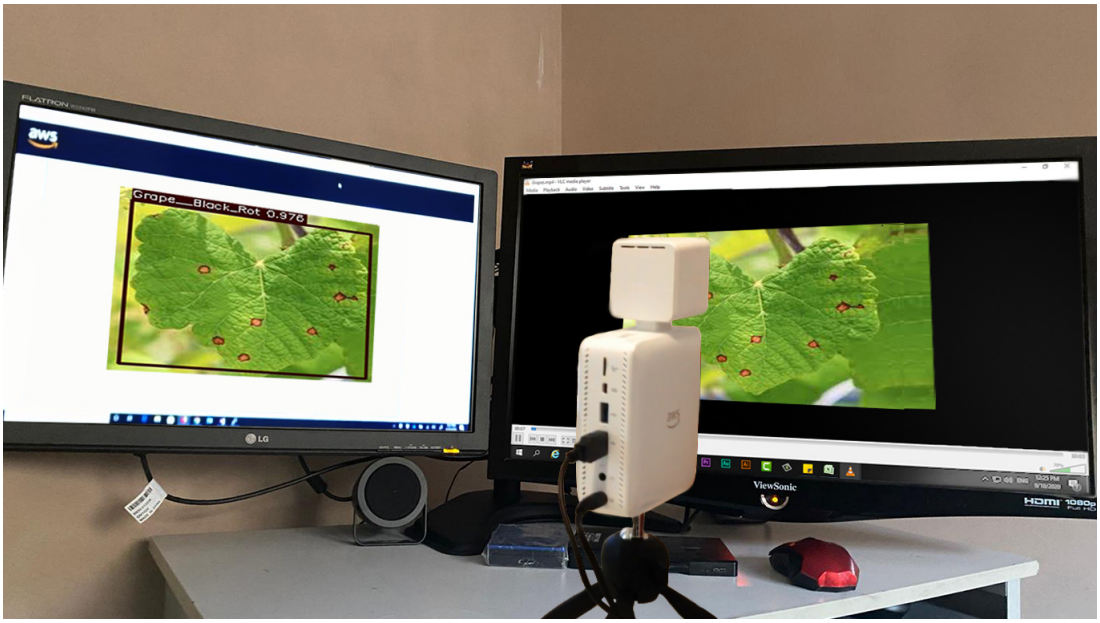


Figure 1. AWS DeepLens Hardware Setup. Input Source (Video) Is In The Right Side Monitor While Output Shown In The Left Side Monitor

Technology is playing a vital role in exploring agriculture sector. Researchers are trying to explore plant disease detection and classification through the use of different machine learning and image processing techniques. Manual detection of plant diseases is difficult, time-consuming and unreliable. As a health assessment of an individual plant in a large plot is cumbersome and time-consuming, explicitly repeating this checking process over time [8]. A single plant may have different diseases having the same pattern of symptoms; moreover, various diseases of the plant show similar signs and symptoms [7], making it challenging to identify the specific disease. Thus technology is helping the agriculture sector, for instance, machine learning (ML) [21] algorithms are serving a lot in the process of classification and identification of plant diseases automation. ML helps in monitoring of health assessment of plant and predicting diseases in the plant at early stages [6]. With the time progression, new ML models evolved, and the researchers used them for their experiments in the field of recognising and classifying images. Some of those are used in automation of Agriculture systems [26].

For the classification and detection of the plant leaves diseases, several classical and modern ML models are used, such as SVM, VGG architectures, R-FCN, Faster R-CNN, SDD and many others. The advancement in deep learning (DL) [9] has provided promising results and solutions in crop disease diagnosis and classification. Islam et al., [14] presented the integration of machine learning and image processing for the detection and classification of leaf disease images. They developed an SVM model for potato disease detection and used potato leaves dataset, consisting of healthy leaves and diseased leaves. For performance, they used performance parameters such as accuracy, sensitivity, recall and F1-score. Dubey et al., [10] came up with an image processing technique by using the K-Means

algorithm for the detection and classification of apple fruit disease and then used multiclass SVM for training and testing images. Al-Amin et al., [6] trained their model for potato disease detection through Deep CNN, and they computed performance for analysing the result using parameters such as recall, precision and F1-score. This model achieved an accuracy of 98.33 % in experiments. According to Sladojevic et al., [29] in order to learn features, CNN must be trained on a large dataset of a large number of images. They developed a CNN model for classification of leaves diseases of apple and tomato plants and the experimental accuracy findings of their research for numerous diseases trial with an accuracy of 96.3%. Miaomiao et al., [16] presented an effective solution for grape diseases detection as they mentioned that two entirely different basic models integrated, it would be more useful to obtain remarkable results and improve the accuracy of detection. Therefore, they proposed a UnitedModel based on the integration of GoogLeNet with ResNet, whereas GoogLeNet raises the total units for all layers of a network and ResNet to raise the total number of layers in a network. Ye Sun et al., [30] developed a model based on structured-illumination reflectance imaging (S.I.R.I.) for identification of peach fungal diseases. CNN and three image classification methods used for processing of ratio images, alternating component (AC) images and direct component (DC) images to detect the diseases and area of peach. As a result, they found that A.C. images performance is better than D.C. images in peach diseases detection and ratio images gave a high accuracy rate. Hyeon Park et al.,[25] developed a CNN network of two convolutional and three fully connected layers, for disease detection in the strawberry plant. They worked on a small dataset of leaves images consisting of healthy leaves and a powdery mildew strawberries disease class. Xiaoyue et al., [34] worked on four typical grapes diseases, and for detection, they proposed a Faster DR-IACNN detector, based on deep learning. They reported that their proposed detector automatically detects the diseased spots on grapes leaves, thus giving an excellent result for the detection of diseases in real-time. In order to detect leaves diseases in vegetables, Zhang et al., [36] come up with a model that is RGB colours based three channels CNN.

According to the above-discussed studies, CNN always played a significant role and is widely used in the detection and classification of different plant diseases and provided agreeable results. However, there were some limitations such as these approaches lack usability due to hardware incompatibility issues, limited scalability, inefficiency and some real-time inferences in the practical usage in the real world. Konstantinos et al., [11] detected and classified 25 plant diseases by using different CNN based architectures. They trained and tested their model on the open-source dataset named PlantVillage. However, the results obtained in terms of accuracy may differ from using the same dataset for both training and testing purposes. We used a combination of PlantVillage dataset and images collected from the real-cultivation environment and applied different data augmentation techniques on the training data so that we can achieve high accuracy and a robust model.

Moreover, we used a Cloud-based environment for our training and testing as well as implemented the model in AWS DeepLens for real-time results. The recent development in cloud-based services and efficient deep learning has motivated us to devise practical and scalable solutions to agricultural problems, and this paper lies in the similar domain. We proposed a model known as DeepLens Classification and Detection Model (DCDM) to detect and classify various fruits and vegetables leaves diseases, based on Deep Convolutional Neural Network (DCNN) with the integration of IoT Device like AWS DeepLens with an average accuracy rate of 98.78%. In our work, we extracted feature maps [29] of an input image after passing through the CNN model and applied filters to visualise the activations through the CNN layers [22]. We have trained the DCDM based upon PlantVillage dataset [35] and images collected from Tarnab Farm (an agriculture research institute)Pakistan. One limitation we found that most of the images in the PlantVillage dataset are either white or grey background; however, the real-world situation is different and may contain other colours in the background. Thus model trained only uniform background colour may result in low accuracy or wrong prediction. Therefore, in training out DCDM, we included real-world environment field images for training to get maximum accuracy and correct predictions. To make our model scalable

and efficient real-time classification and detection, it is integrated with AWS Cloud and implemented in IoT device, namely AWS DeepLens camera. We tested our system on real-time images of twenty-five classes of plant leaves comprising of a healthy leaves class and twenty-four classes of leaves diseases in apple, peach, grapes, strawberry, tomato and potato. The leaves diseases classes for fruits and vegetables that we used in our dataset listed in Table 1 with both standard and botanical names. Our particular focus is on leaves as they play an essential role in providing energy and producing food supplements for plants. The photosynthesis process produces food in the leaves, and the produced food is supplied to stems and rest of the plant to fulfil its food requirements to keep healthy. Hence, to get more fruit from the plant, their health monitoring is essential. Therefore, our work focuses on early disease detection and classification through health assessment of plant leaves. Another reason for leaf focused disease detection is that leaves remain on the plants for longer times rather than fruits and flowers because they supply energy to the plant.

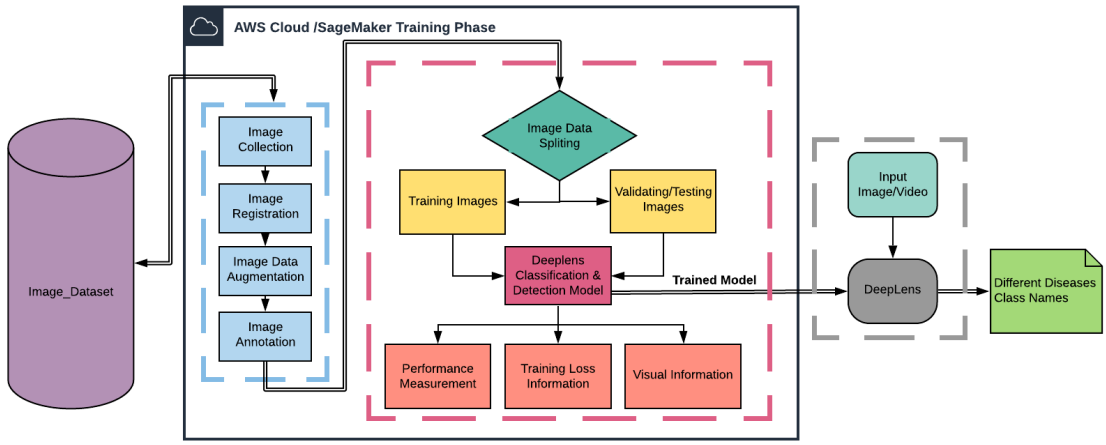


Figure 2. Data Flow Diagram of the DCDM Using AWS Cloud Backend.

2. Materials and Methodology

The development process of DCDM model for plant leaves disease detection, and classification consists of various stages, i.e. starting with data collection along with data pre-processing and preparation, Training model in AWS Cloud (SageMaker Studio) [17] and implementing in AWS DeepLens for inferences purpose, as shown in Figure 1. We used real-time videos and images for testing purpose but in the figure we are testing a recorded video on one monitor and output on the other.
Further details are as below.

2.1. Transfer Learning in AWS Cloud

Transfer learning (TL) is a concept in the ML which simply means that a method learns basic knowledge in solving a particular problem and later reusing that knowledge for other more or less similar problem solution [19]. This technique encourages us to use for solving any relevant problem for which there is not sufficient data available. Thus it relaxed the assumption of training and testing data, should be both distributed identically and independently [31]. It takes a long time and large-sized dataset for training CNN from the very scratch. Hence in certain situations where the dataset is limited, then TL is a helpful method. For our model, we used TL for different architectures and then training our own model from scratch. To address the scalability limitation, we choose Amazon's Cloud platform and AWS DeepLens. Amazon's cloud platform provides the facility of data storage, data transfer and computational capability. Amazon Web Services (AWS) provides multiple services for many different applications. They also provide a platform to build, train and deploy as well as to validate machine learning models. The trained model can be deployed on AWS Services or any

other compatible platforms, for instance, AWS DeepLens. We used SageMaker Studio for DCDM model training in the AWS Cloud Services so that after completion of training, the obtained trained model could be implanted/deployed in AWS DeepLens as both the platforms are compatible. While comparing with another pre-trained model, we used our system for training and testing purposes.

A typical CNN consists of various layers. Each layer consists of multiple nodes with some activation function attached. The first layer is the input layer that takes input data, whereas, the last layer is the output layer that generates output. A random number of layers exists between input and output layer, referred to as hidden layers (i.e. Convolutional or Convo, Pooling, Dense or Fully Connected and SoftMax layer). If CNN contains two or more than two hidden layers, it is known as Deep Convolutional Neural Network (DCNN) [23].

We designed our DCDM using deep learning TensorFlow framework [5] and Keras [13] library. Keras is an open-source deep-learning library used to perform different deep learning applications. We used it for the implementation of DCDM architecture, inspired from architecture of VGG19. This architecture uses filters of the same width and height for all the convolutional layers. That is why the architecture of VGG19 used to be very fast than the other state-of-the-art architectures like ResNet50, DenseNet, InceptionVNet. The VGG19 architecture consists of roughly about 139 million parameters [2] which makes it computationally expensive for training purpose. However, our architecture has same sequential structure as of VGG19 but with some less number of layers. Also, the numbers of parameters are extensively low, which makes it computationally less expensive.

Our architecture comprises of a total of nine layers with six convolutional layers and three fully connected layers. Figure 3 shows visual representation of DCDM Convolutional layers are having non-linearity activation units following by max-pooling layers. The non-linearity activation is often used with convolutional layers. This activation is also known as a ramp function which has a shape of the ramp and transfers the output once it is a positive value; else it results in 0. The last layer, which is also known as a SoftMax layer comprising of 25 nodes is our output layer where each node specifies an individual class of our dataset.

The details of these layers are described as:

Convolutional Layer: The above stated proposed model used six convolutional layers. There are two types of characteristics in each layer, i.e., input and numeral filters. The filter numbers are then convolved on each layer which extracts the useful features and passes it to the next connected layer. For an RGB image, each filter is applied to all three colour channels, and thus, a corresponding matrix is obtained accordingly. We used a filter size of 3 x 3 for all convolutional layers. The number of filters and input of each layer is elaborated below

(a) Convolution 272 x 363 – 64 filter
(b) Convolution 272 x 363 – 64 filter
(c) Convolution 136 x 181 – 128 filte
(d) Convolution 68 x 90 – 256 filter
(e) Convolution 34 x 45 – 512 filter
(f) Convolution 17 x 22 – 512 filter

Pooling Layer: Most commonly, the pooling layer follows each convolutional layer. There are five max-pooling layers in the proposed method. The pooling layers are often used to minimise computational cost as it reduces the size of each convolutional layer output. The max-pooling has an activated filter which slides on the input and based on the size of the filter, and the max value is selected as an output. We used a filter of 2 x 2 for all max-pooling layer. The characteristics for each layer is given below:

(a) Max-pooling 136 x 181 – 64 filter
(b) Max-pooling 68 x 90 – 128 filter
(c) Max-pooling 34 x 45 – 256 filter

(d) Max-pooling 17 x 22 – 512 filter
(e) Max-pooling 8 x 11 – 512 filter

Dense Layer: It is also known as an artificial neural network (ANN) classifier. Our model has three dense or fully connected layers. In fully-connected layers, each node is connected with only one node of another layer. The first two fully-connected layers have ReLu activation while the last layer, which is also known as the output layer, has a softmax activation. The softmax activation works by finding the node with the highest probability value of prediction being made. Hence the node with the higher value is forwarded as an output.

(a) Dense Layer: 1024 units
(b) Dense Layer: 1024 units
(c) Dense Layer: 25 units

Dropout: The overfitting issue is prevented by the addition of a dropout of 0.5. It is added to the dense layers of the model.

Parameters: The total model parameters of our model are 51,161, 305.

The model takes the image data as an input, then processes that input data by extracting features from the image and then classifies it either healthy or a diseased leaf, if it is an infected leaf then it further predicts the disease class name, the most resemble one. The predicted class then results as an output.

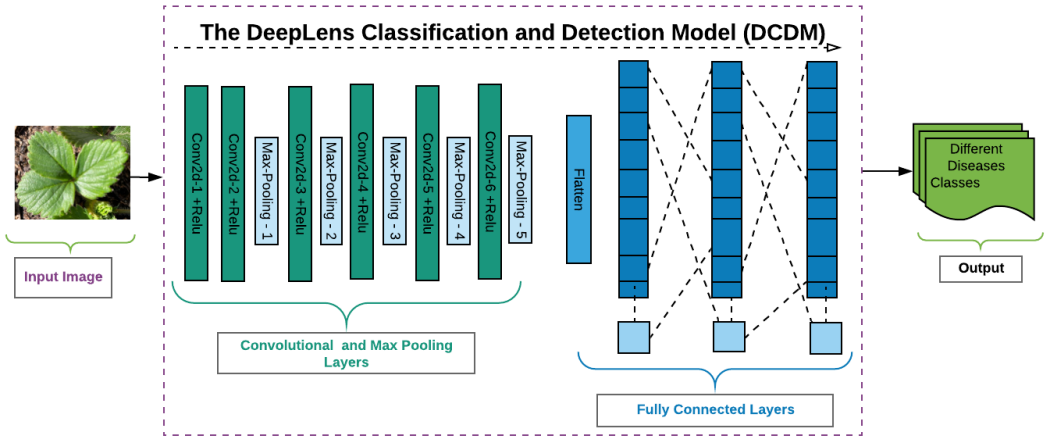


Figure 3. The DCDM Layered Architecture.

2.2. Lambda Function on DeepLens

AWS DeepLens is a deep-learning-based H.D. 4-mega-pixel video camera that is designed specifically for machine learning models developments and implementation. It has a built-in 8GB memory and 16GB storage capacity with 32 GB SD card (extendable). It has more than 100 GFLOPS computing power so it can process machine learning projects independently as well as those integrated with AWS Cloud [15]. It has a straightforward usage process as the user can take picture/ image through DeepLens camera, then store it and process it to use in machine learning projects [12]. There are a large number of pre-trained models, built-in to it, but a customised model can also be used with DeepLens camera. For instance, any custom based model can be trained or imported into in SageMaker and then can be implemented in AWS DeepLens through various deep learning frameworks such as Tensorflow, Caffe [15],[12]. A lambda function is used to establish a successful connection to access the DeepLens on a local computer. The lambda functions are the pre-defined functions executed by DeepLens once the project has been deployed [3]. Lambda function streamlines the development

 doi:10.20944/preprints202009.0142.v3

process by managing the servers necessary to execute code. They serve as the connection between the AWS DeepLens and Amazon Sagemaker for the camera to generate a real-time inference [4]. It controls various resources such as computing capability and power, networking. It has a user-specified function embedded in code, and Lambda function invoke that user code when it is executed. The code returns a message containing data from the event received as input [4]. The visual illustration of the AWS DeepLens work-flow is shown in Figure 4.

After completing the training stage in SageMaker, we implemented the subsequent trained Model in AWS DeepLens camera for inferences of Leaves health assessment. Figure 2. represents the overall flow of the proposed model.

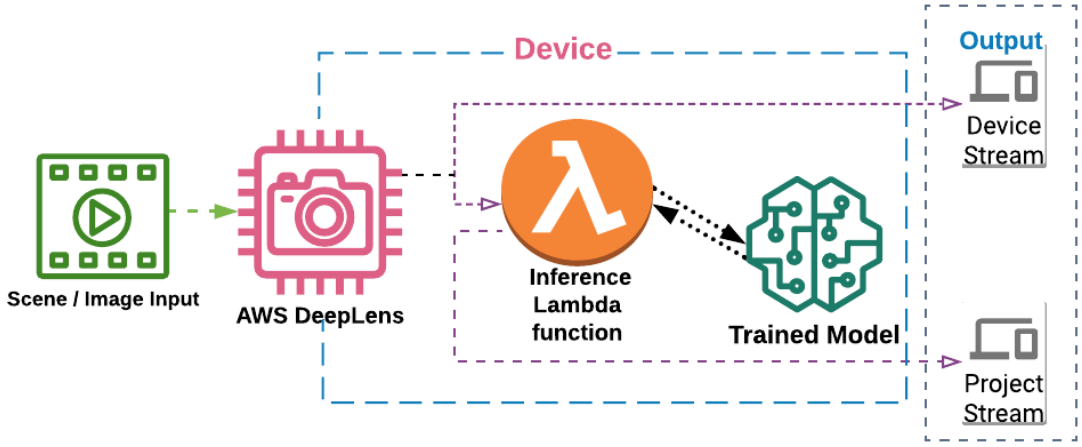


Figure 4. DeepLens [14] Working Framework is Displayed.

3. Dataset Preparation

We used a collection of plant leaves images (including both healthy and infectious leaves images for fruits and vegetables) from local farmlands and publicly available dataset known as PlantVillage [35]. We analysed around 50,000 images of plant leaves, which categorised into 25 classes and assigned labels to all. Each label represents either a plant disease class or its healthy nature class. A sample image for each class shown in Table 1.

				
a.	b.	c.	d.	e.
				
f.	g.	h.	i.	j.

Table 1 continued from previous page

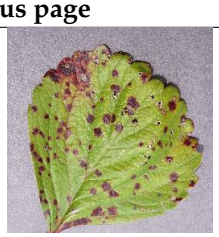
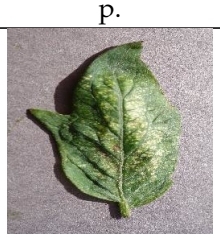
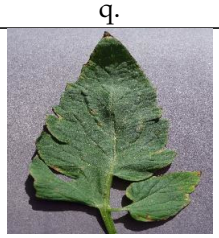
				
k.	l.	m.	n.	o.
				
p.	q.	r.	s.	t.
				
u.	v.	w.	x.	y.

Table 1. Sample Dataset Images: (a). Apple Scab, (b). Black Rot, (c). Cedar Apple Rust, (d). Apple Healthy, (e). Grape Black Rot, (f). Grape Esca, (g). Grape Leaf Blight, (h). Grape Healthy, (i). Peach Bacterial Spot, (j). Peach Healthy, (k). Potato Early Blight, (l). Potato Late Blight, (m). Potato Healthy, (n). Strawberry Leaf Scorch, (o). Strawberry Healthy, (p). Tomato Bacterial Spot, (q). Tomato Early Blight, (r). Tomato Late Blight, (s). Tomato Leaf Mold, (t). Tomato Septoria Leaf Spot, (u). Tomato Spider Mites, (v). Tomato Target Spot, (w). Tomato Leaf Curl Virus, (x). Tomato Mosaic Virus, (y). Tomato Healthy.

3.1. Data Augmentation

For training a DCNN model, a large number of images used for achieving a highly accurate prediction and accuracy. In our case, some of the plants leaves disease classes had fewer images in number; therefore, the process of data augmentation (technique) applied to those limited number of image diseases classes. The process of data augmentation [28] provided us with new images from our existing images. Different augmentation techniques like blurriness, rotation, flipping (horizontal and vertical), shearing (horizontal and vertical), and addition of noise were applied accordingly. An illustration of different augmentation techniques shown in Table 2. By using this technique, the number of images in our dataset increased, which is essential for obtaining more accurate results after the training stage of CNN. The techniques used for augmentation adds new information to the existing images. For instance, the addition of noise, Gaussian noise is added to the image, and thus an image gets noisy. For horizontal flipping, the image is flipped at the centre on x-axis. Thus, the information from the right side is shifted to the left side. For rotation, a certain angle is applied to the original image, and thus a new image is generated. All these augmentation techniques tweak the present information of an image and help to generate new images with some modifications.

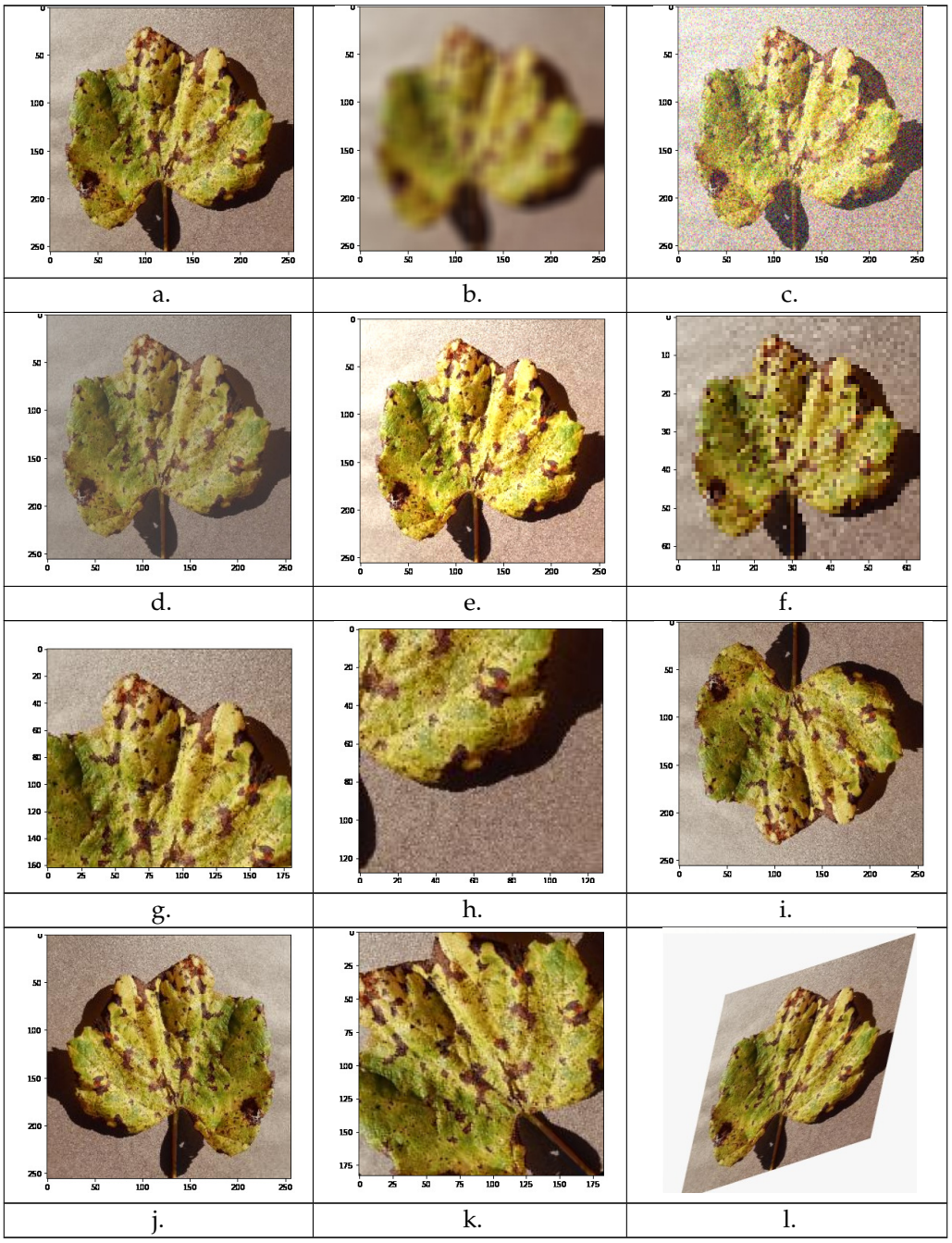


Table 2. Various Data Augmentation Technique Examples: (a). Original Image, (b). Blur, (c) Random Gaussian Noise, (d). Random Contrast, (e). Random Bright, (f). Scale Proportionality, (g). Random Crop, (h). Deterministic Crop, (i). Vertical Flip, (j). Horizontal Flip, (k). Rotate Without Padding, (l). Y-Sheared.

3.2. Image Registration and Classes Annotation

After completion of data augmentation process, we had to re-register the images in same dimensions. As we used two types of dataset having different dimensions. Image registration [38][18] is an essential step in the image processing like quality improvement and formation of geometrically aligning images like others in the same dataset. We resized all the images into 272 x 363 pixels and annotated all the images before putting image as an input to any model/network for pre-training CNN structures. The labelled class names listed in Table 3.

Table 3. The Description of Leaf Disease Dataset

Class No.	Pant Name	Plant Botanical Name	Disease Name	Disease Botanical Name	Training Images	Validation Images
1	Apple	Malus Domestica	Scab	Venturia inaequalis	1504	326
2	Apple	Malus Domestica	Black rot	Botryosphaeria obtusa	1496	325
3	Apple	Malus Domestica	Cedar apple rust	Gymnosporangium juniperivirginianae	1220	455
4	Apple (Healthy)	Malus Domestica			1395	329
5	Grapes	Vitis vinifera	Black rot	Guignardia bidwellii	1944	236
6	Grapes	Vitis vinifera	Esca	Phaeomoniella chlamydospora	1107	276
7	Grapes	Vitis vinifera	Leaf blight	Pseudocercospora vitis	1860	215
8	Grapes (Healthy)	Vitis vinifera			1339	484
9	Peach	Prunus persica	Bacterial spot	Xanthomonas campestris	1838	459
10	Peach (Healthy)	Prunus persica			1288	572
11	Potato	Solanum tuberosum	Early blight	Alternaria solani	1800	200
12	Potato	Solanum tuberosum	Late blight	Phytophthora infestans	1800	200
13	Potato (Healthy)	Solanum tuberosum			1121	531
14	Strawberry	Fragaria spp.	Leaf scorch	Diplocarpon earlianum	1887	350
15	Strawberry (Healthy)	Fragaria spp.			1364	492
16	Tomato	Lycopersicum esculentum	Bacterial spot	Xanthomonas campestris pv. Vesicatoria	1710	425
17	Tomato	Lycopersicum esculentum	Early blight	Alternaria solani	1800	457
18	Tomato	Lycopersicum esculentum	Late blight	Phytophthora infestans	1527	382
19	Tomato	Lycopersicum esculentum	Leaf mold	Fulvia fulva	1761	491
20	Tomato	Lycopersicum esculentum	Septoria leaf spot	Septoria lycopersici	1417	454
21	Tomato	Lycopersicum esculentum	Spider mites	Tetranychus urticae	1340	335
22	Tomato	Lycopersicum esculentum	Target spot	Corynespora cassiicola	1123	481
23	Tomato	Lycopersicum esculentum	Leaf curl virus		3286	571
24	Tomato	Lycopersicum esculentum	Mosaic virus	Tomato mosaic virus	1800	574
25	Tomato (Healthy)	Lycopersicum esculentum			1273	380

3.3. Features Maps Extraction and Filters Visualization in CNN Layers

3.3.1. Extraction of Feature Maps

Feature maps [20] are used to present the local information passing through the CNN Layers. In an ideal feature mapping of CNN, they are sparse and help in the understanding of the classical model. In convolutional layer, to extract feature maps from the source image, several mathematical computations are carried out [32]. In Figure 5, a visual representation for the extraction of feature maps presented for various layers of our model. It also provides information about each layer, i.e. what and how a particular layer of CNN gains information from other layers, such piece of information can help the developer to make proper adjustments in the developing model for best results. From our visualisation images, we found that our model is gaining information in the hierarchical order. It means that the high-level layers present more specific features and vice versa.

Similarly, if the dimensions are higher than feature maps would also classify images more accurately. For instance, in an image, edge corners, some abstract colour features are presented by a deep layer Figure 5, while other corners and edges represented in shallows layers. Moreover, the middle layers are usually responsible for capturing the same textures because these layers are having complex invariance and more layers in number, after extracting higher-level abstract features, the striking posture of the entire image shown by the high-level feature map.

The feature maps extracted in the first layer represents the overall physical appearance of the leaf image. In the middle layers, the patterns of disease are extracted as can be seen in Figure 6. The last layers in Figure 6 often extract the delicate features as they are then used to finalise the predicted class.

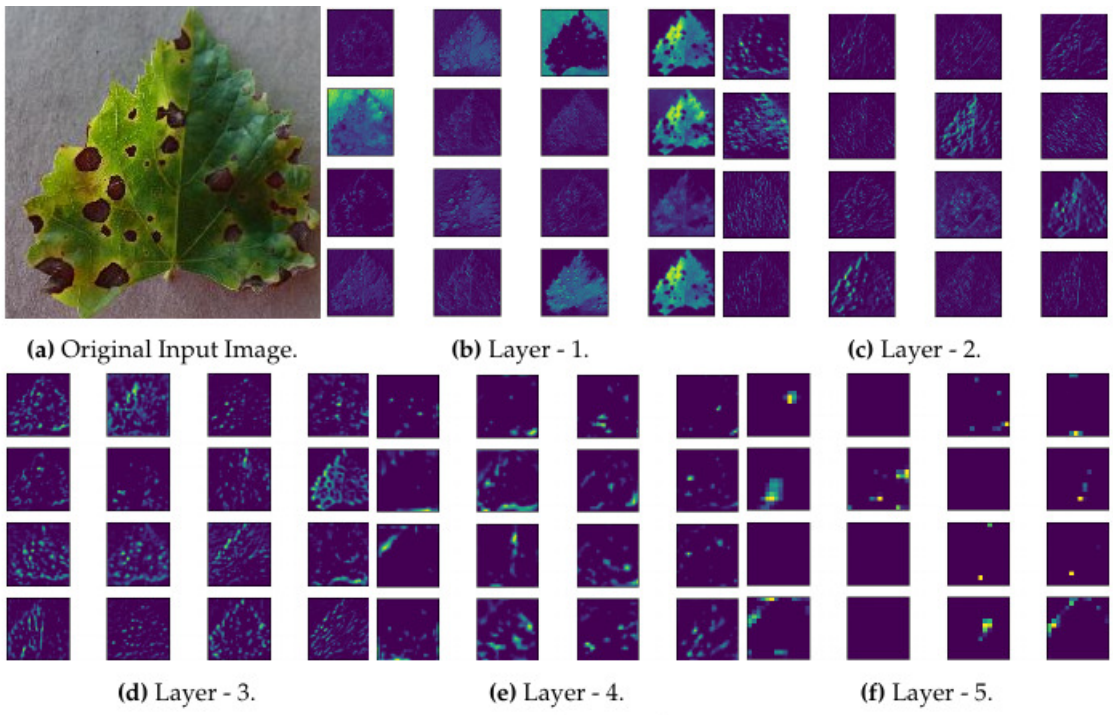


Figure 5. Feature Map Visualisation of DCDM Layers.

3.3.2. Filter Visualization in Model Layers

Generally, filters are used for the detection of unique patterns in an input image. It is done by detecting the change in the intensity values of the image. Thus, each filter has its particular importance for feature extraction [33]. As an example, a high pass filter detects the existence of edges in an image. In our CNN model, various filters are used to extract features like edges, shape, the colour of the leaf, and many more useful features. In Figure 6, a visual representation for a few filters presented

where each filter has its application for extracting leaf features. After detecting the specific feature of the image by a filter, it is then passed to the next layer where other filters extract the additional feature. This process continues until the last layer, and thus integrating all together helps to define the predicted class for an input image.

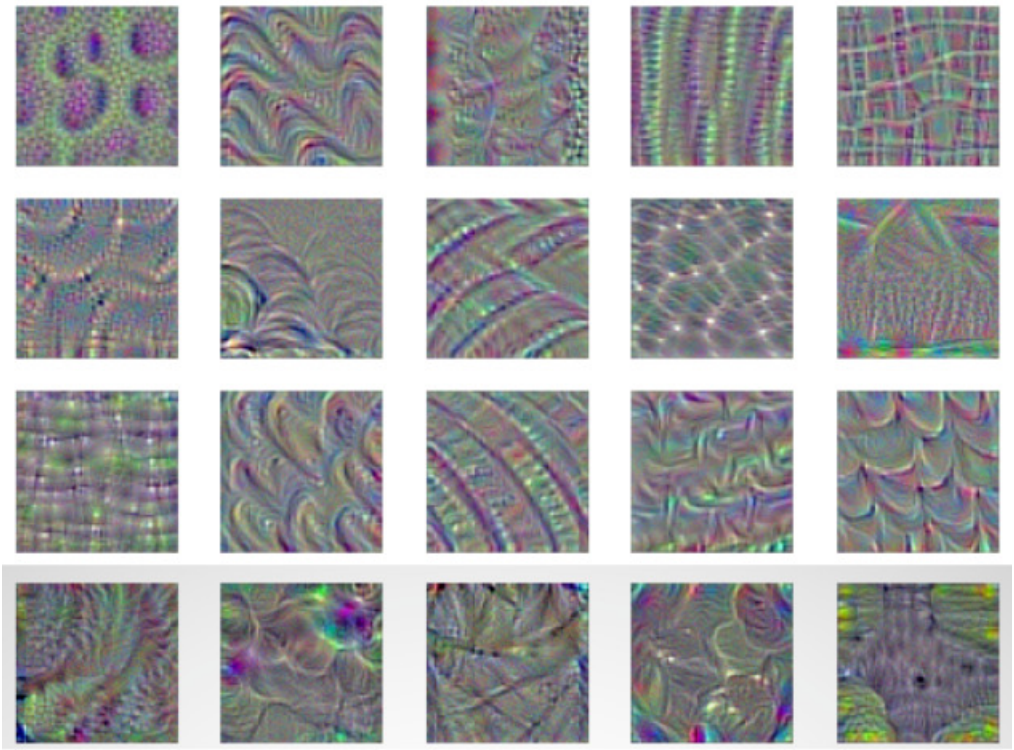


Figure 6. Filter Visualisation.

4. Results and Discussion

After implementing the trained Model in AWS DeepLens, Experiments for DCDM model done on both Windows and Ubuntu operating systems workstations. It had Intel 9th Gen i7 CPU, i.e. 9700K, 32 GB ECC RAM and NVIDIA RTX TITAN 24 GB VRAM GPU. Table 4. shows the system specifications.

Table 4. System Specifications for Testing

System Hardware / Software (Operating System)	Specifications
CPU	Intel 9th Gen i7 9700K
RAM	32 GB ECC RAM
GPU	NVIDIA RTX TITAN 24 GB VRAM
Operating System	Windows 10 Professional and Ubuntu 18.04

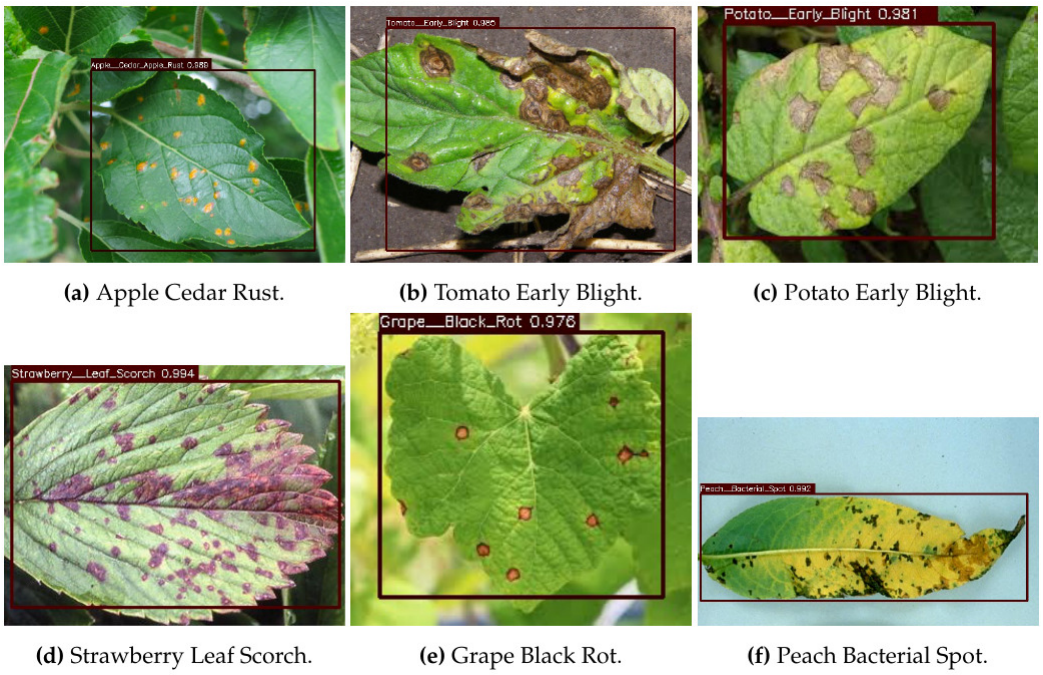
As mentioned above, the deep learning-based framework TensorFlow [5] and Keras Library [13] were the environments used for experiments.

To measure the performance of our approach and to prevent the issue of over-fitting, we also distributed the data into different training-testing sets. First, we split the data into 80-20, in which 80% is used for training whereas 20% for testing purpose. Again, split into 70-30 means 70% of the dataset used for training and 30% for testing purpose and lastly, split into 60-40 training-testing dataset, means 60% of the complete dataset used for training and 40% for testing purpose. The total number of each disease class after the augmentation process and the splitting data ratio is given in detail in Table 5. PlantVillage dataset contains numerous images of the same plant leaves. During the process of splitting data into a training-testing dataset, we kept all data of the same class in one group, i.e. either

in the training set or in the testing set. However, the data split of 80-20 performed very well for DCDM approach with the maximum accuracy of 98.78%. Thus, exceeding the accuracy results for all other CNN architectures. Some of the sample output images with an AWS DeepLens are shown in Figure 7.

Table 5. Dataset Split.

Train Test Data Split %	Training Images	Testing Images
80 - 20	40000	10000
70 - 30	35000	15000
60 - 40	30000	20000



(a) Apple Cedar Rust. **(b)** Tomato Early Blight. **(c)** Potato Early Blight.

(d) Strawberry Leaf Scorch. **(e)** Grape Black Rot. **(f)** Peach Bacterial Spot.

Figure 7. Sample Output Images.

4.1. Performance Measurement

There are many ways for performance measurement that are used to evaluate the performance of neural networks. They include precision, recall, accuracy, and f1-score. The precision tells us about the correct predictions made out of false-positive while recall tells us about the correct predictions made out of false negatives. The accuracy is the number of correct predictions out of both false positives and false negatives. We calculated all the performance measures for our trained model using formulas listed in Eq (1), (2), (3), and (4) from the confusion matrix shown in Figure 9.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{3}$$

$$F1 - Score = 2 * \frac{precision * recall}{precision + recall} \tag{4}$$

Where TP is true positives, TN is true negatives, FP is false positives and FN is false negatives. Here the TP and TN are the correct predictions while the FP and FN are the wrong predictions made by our model. After computing values from the confusion matrix, the results are shown for the 80-20 split in Table 6.

Table 6. Classification / Model Performance Report

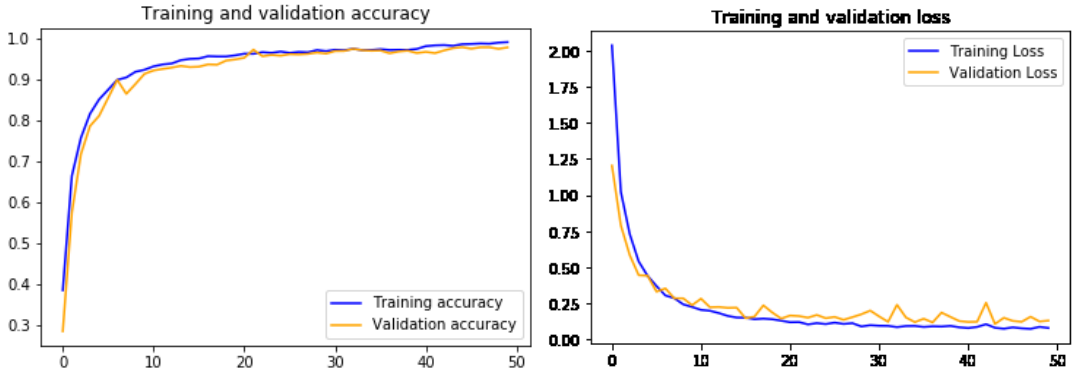
Evaluation Metrics	Value in %
Precision	98.38%
Recall	97.98%
Accuracy	98.78%
F1-Score	98.17%

In Table 7, the performance measure of accuracy for each data split is presented. The values are presented after every ten epochs of training. The bold value for data split of 80-20 and epoch size of 50 represents the highest accuracy.

Table 7. Data Split for Testing / Training & Accuracy Obtained

Dataset (Train/Test) Split in %	Accuracy [%]				
	10 Epochs	20 Epochs	30 Epochs	40 Epochs	50 Epochs
80 – 20	92.31	95.84	96.86	97.39	**98.78**
70 – 30	91.23	94.89	96.15	96.77	97.46
60 – 40	90.70	94.92	95.04	95.98	96.21

In Figure 9, a confusion matrix is presented for a data split of 80 – 20. The confusion matrix shows the prediction of the model on a visual basis. The values on the diagonal are the correct predictions that are made by a model for the testing dataset. While any value other than diagonal represent the wrong prediction of a model. In this matrix, the healthy images of both fruits and vegetables are mostly predicted correctly. For example, a grape leaf, all the healthy images of grape leaf were classified correctly and was not mixed with any other leaf class of fruits or vegetables. Similarly, the disease class of grape leaf blight was also successfully predicted with no wrong predictions.



(a) Training and Validation Accuracy **(b)** Training and Validation Loss

Figure 8. Training and Validation Graphs

In Figure 8 (a) (b), the accuracy and loss for both training and testing/validating are presented for each epoch. These graphs were generated for the data split of 80 – 20. The accuracy graph visually shows that accuracy for both training and testing increases gradually and then tends to converge on a specific point. It also shows that after 40 epochs, the change in accuracy reduces as the validation accuracy appears to be equivalent to training accuracy. Similarly, the right graph shows how the loss

starts decreasing gradually as the model learns on a given dataset. The loss of validation data becomes stable after 43 epochs and thus tends towards a specific value.

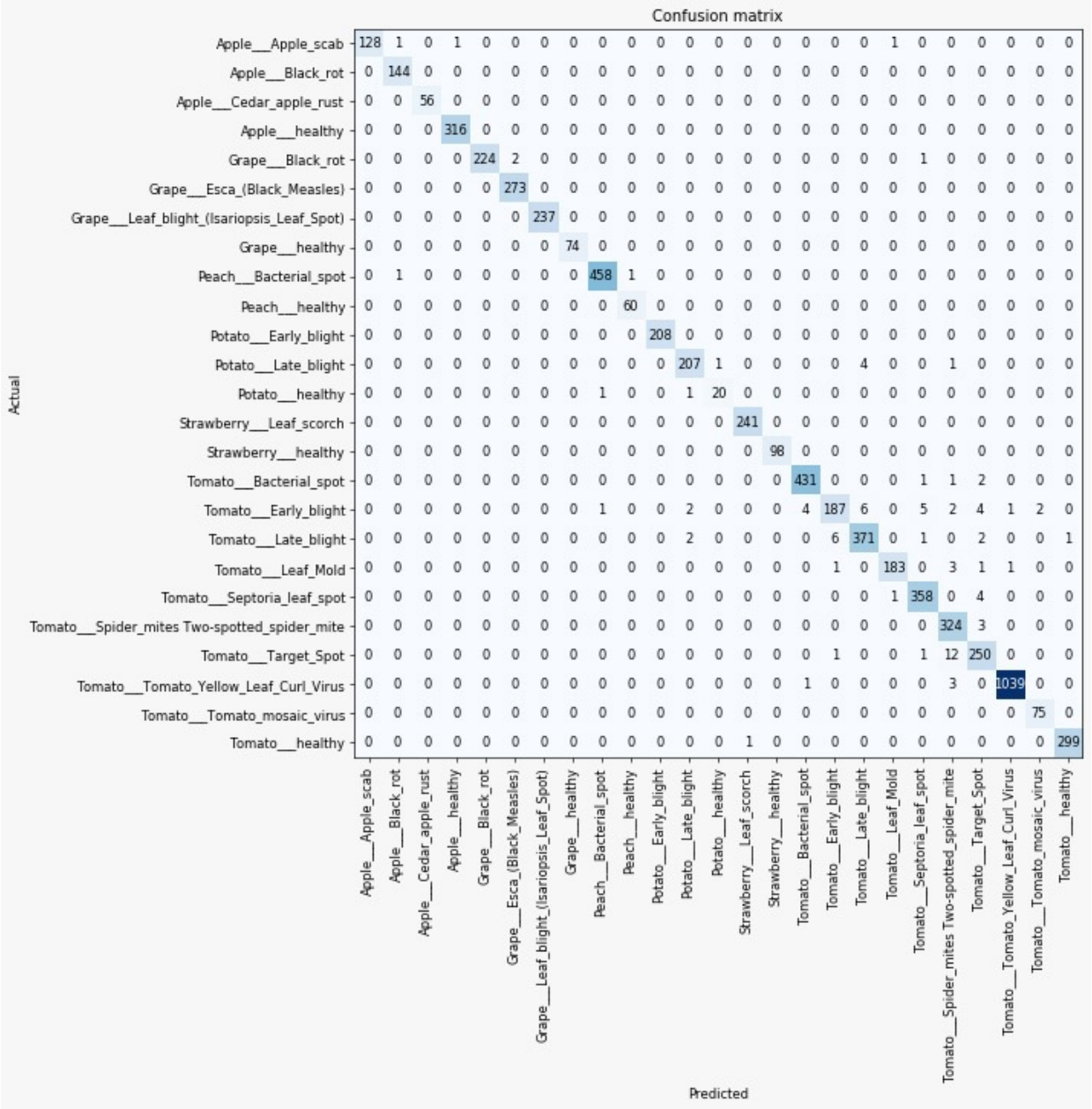


Figure 9. Confusuion Matrix for 80 -20 % Data Split.

4.2. Comparative Analysis

In this section, a visual analysis of different CNN architectures made with the DCDM approach. Training the model on different architecture is a critical approach used to find the best architecture for targeted application. The architectures we used are the best performing architectures for the classification problem. We obtained a performance accuracy of 90+ for each trained architecture. The architecture of AlexNet performs with the lowest accuracy of 92.43%. This architecture is considered as the smallest architecture and most straightforward architecture out of all. However, still providing us with accuracy above 90%. The VGG16 and VGG19 architectures are the same with some minor modifications and a different number of layers. They have a significant record of performing very well

for the classification problems. For our testing dataset, they provide us with an accuracy of 94.05% and 96.89% respectively.

Similarly, the architecture of SqueezeNet and DenseNet also performed with an accuracy of 94.67% and 96.59%. The ResNet50 architecture is well-known for good performance on large datasets. It has a bulk of 50 layers with different inter-connections. Thus, performing with an accuracy of 97.85% and being able to score the position of the third-best architecture in our list. The architecture of DarkNet provides an accuracy close to DCDM approach. It results from an accuracy of 98.21%, scoring the position of second-best architecture while DCDM architecture performed outstanding and stood with the position of best architecture with an accuracy of 98.78%. The results for each architecture is shown in Table 8 and then visually represented in Figure 10. We compared the performance of each architecture for the testing dataset. An evaluation metric of accuracy was used for comparison, based on Equation 3.

The comparison of each architecture with respect to time consumed has also been made which results in the time required for training. The average time for each training epoch is presented in Table 8. The time consumed by our architecture requires less computation time, thus having lowest average time while training. It testifies that our architecture is the most efficient both performance as well as computation wise.

Table 8. Disease Classes Accuracy

Trained CNN Models	Accuracy in %	Average Time Per Epoch(in Minutes)
ResNet - 50	97.85%	2:03
DensNet	96.59%	2:38
VGG-16	94.05%	1:53
VGG-19	96.89 %	1:59
AlexNet	92.43 %	1:44
SqueezeNet	94.67%	2:32
DarkNet	98.21%	2:13
Our Model	**98.78%**	1:26

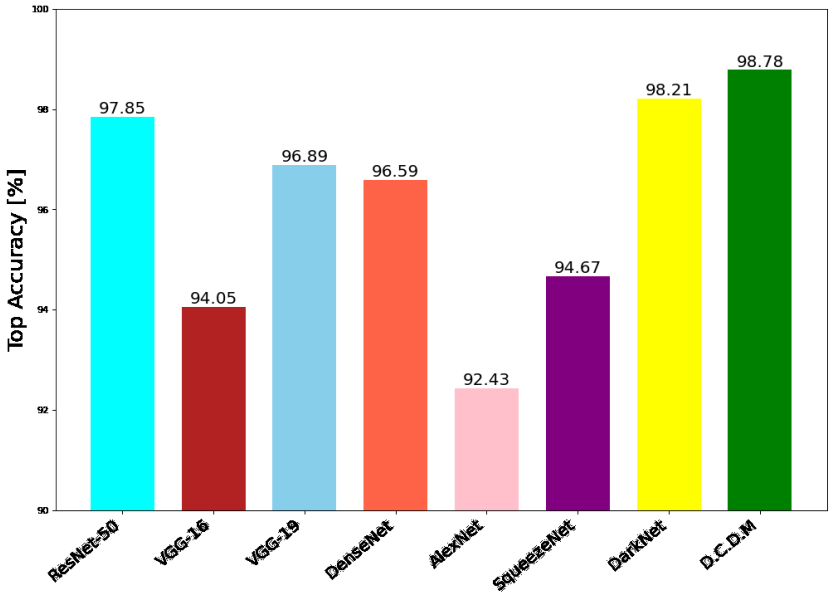


Figure 10. Comparison Graph

5. Conclusion

This proposed deep model implemented on AWS DeepLens can predict 25 different disease classes in Apple, Grape, Peach, Potato, Strawberry and Tomatoes in real-time. Our model obtained 98.78% accuracy in predictions and classifications in real-time on-field experimentations. This practical approach would facilitate the agriculture-related professionals and community by contributing to the Agri-economy enhancement as the grave problem of plant (leaves) diseases would be easily detected and classified instantly. In addition, this approach is scalable, and we can add more classes of other vegetables and fruit leaves. In our future work, we would integrate our models to different mobile platforms such as iOS, Windows and Android-based Applications to increase its usability. Furthermore, new techniques such as Multi-spectral and Hyper-spectral images should also be experimented for detection and classification of plant diseases.

Abbreviations

The following abbreviations are used in this manuscript:

DL	Deep Learning
TL	Transfer Learning
ML	Machine Learning
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
DCDM	DeepLens Convolutional Detection Model

References

1. Ap 102-10_1.pdf. https://secure.caes.uga.edu/extension/publications/files/pdf/AP%20102-10_1.PDF. (Accessed on 07/15/2020).
2. Aws deeplens – deep learning enabled video camera for developers - aws. https://aws.amazon.com/deeplens/. (Accessed on 07/19/2020).
3. Create and publish an aws deeplens inference lambda function - aws deeplens. https://docs.aws.amazon.com/deeplens/latest/dg/deeplens-inference-lambda-create.html. (Accessed on 08/10/2020).
4. Invoke aws lambda functions - amazon connect. https://docs.aws.amazon.com/connect/latest/adminguide/connect-lambda-functions.html. (Accessed on 08/10/2020).
5. Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pp. 265–283.
6. Al-Amin, M., T. A. Bushra, and M. Nazmul Hoq. 2019. Prediction of potato disease from leaves using deep convolution neural network towards a digital agricultural system. In *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1–5.
7. Al-Shawwa, Mohammed and Samy S Abu-Naser. 2019. Knowledge based system for apple problems using clips. *International Journal of Academic Engineering Research (IJAER) 3*(3), 1–11.
8. Boulent, Justine, Samuel Foucher, Jérôme Théau, and Pierre-Luc St-Charles. 2019. Convolutional neural networks for the automatic identification of plant diseases. *Frontiers in plant science 10*.
9. Butt, Charmaine, Jagpal Gill, David Chun, and Benson A Babu. 2020. Deep learning system to screen coronavirus disease 2019 pneumonia. *Applied Intelligence*, 1.
10. Dubey, Shiv Ram and Anand Singh Jalal. 2012. Detection and classification of apple fruit diseases using complete local binary patterns. In *Proceedings of the 3rd international conference on computer and communication technology*, pp. 346–351.
11. Ferentinos, Konstantinos P. 2018. Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture 145*, 311–318.
12. Galkin, Maria, Kashmala Rehman, Benjamin Schornstein, Warren Sunada-Wong, and Harvey Wang. 2019. A hygiene monitoring system.

13. Géron, Aurélien. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.
14. Islam, Monzurul, Anh Dinh, Khan Wahid, and Pankaj Bhowmik. 2017. Detection of potato diseases using image segmentation and multiclass support vector machine. In *2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE)*, pp. 1–4. IEEE.
15. Jaworek-Korjakowska, Joanna, Pawel Kleczek, and Marek Gorgon. 2019. Melanoma thickness prediction based on convolutional neural network with vgg-19 model transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0.
16. Ji, Miaomiao, Lei Zhang, and Qiufeng Wu. 2019. Automatic grape leaf diseases identification via unitedmodel based on multiple convolutional neural networks. *Information Processing in Agriculture*.
17. Joshi, Ameet V. 2020. Amazon's machine learning toolkit: Sagemaker. In *Machine Learning and Artificial Intelligence*, pp. 233–243. Springer.
18. Khan, Asim, Anwaar Ulhaq, and Randall W Robinson. 2019. Multi-temporal registration of environmental imagery using affine invariant convolutional features. In *Pacific-Rim Symposium on Image and Video Technology*, pp. 269–280. Springer.
19. Khan, SanaUllah, Naveed Islam, Zahoor Jan, Ikram Ud Din, and Joel JP C Rodrigues. 2019. A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognition Letters 125*, 1–6.
20. Liu, Tianrui, Jun-Jie Huang, Tianhong Dai, Guangyu Ren, and Tania Stathaki. 2020. Gated multi-layer convolutional feature extraction network for robust pedestrian detection. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3867–3871. IEEE.
21. Michie, Donald, David J Spiegelhalter, CC Taylor, et al. 1994. Machine learning. *Neural and Statistical Classification 13*(1994), 1–298.
22. Mohanty, Sharada P, David P Hughes, and Marcel Salathé. 2016. Using deep learning for image-based plant disease detection. *Frontiers in plant science 7*, 1419.
23. Nielsen, Michael A. 2015. *Neural networks and deep learning*, Volume 2018. Determination press San Francisco, CA.
24. Park, Hyeon, Jee-Sook Eun, and Se-Han Kim. 2017a. Image-based disease diagnosing and predicting of the crops through the deep learning mechanism. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 129–131. IEEE.
25. Park, Hyeon, Jee-Sook Eun, and Se-Han Kim. 2017b. Image-based disease diagnosing and predicting of the crops through the deep learning mechanism. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 129–131. IEEE.
26. Saleem, Muhammad Hammad, Johan Potgieter, and Khalid Mahmood Arif. 2019. Plant disease detection and classification by deep learning. *Plants 8*(11), 468.
27. Sharif, Muhammad, Muhammad Attique Khan, Zahid Iqbal, Muhammad Faisal Azam, M Ikram Ullah Lali, and Muhammad Younus Javed. 2018. Detection and classification of citrus diseases in agriculture based on optimized weighted segmentation and feature selection. *Computers and electronics in agriculture 150*, 220–234.
28. Shorten, Connor and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data 6*(1), 60.
29. Sladojevic, Srdjan, Marko Arsenovic, Andras Anderla, Dubravko Culibrk, and Darko Stefanovic. 2016. Deep neural networks based recognition of plant diseases by leaf image classification. *Computational intelligence and neuroscience 2016*.
30. Sun, Ye, Renfu Lu, Yuzhen Lu, Kang Tu, and Leiqing Pan. 2019. Detection of early decay in peaches by structured-illumination reflectance imaging. *Postharvest Biology and Technology 151*, 68–78.
31. Tan, Chuanqi, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. 2018. A survey on deep transfer learning. In *International conference on artificial neural networks*, pp. 270–279. Springer.
32. Tümen, Vedat, Ömer Faruk Söylemez, and Burhan Ergen. 2017. Facial emotion recognition on a dataset using convolutional neural network. In *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pp. 1–5. IEEE.
33. Xie, Guotian, Kuiyuan Yang, and Jianhuang Lai. 2019. Filter-in-filter: Low cost cnn improvement by sub-filter parameter sharing. *Pattern Recognition 91*, 391–403.

34. Xie, Xiaoyue, Yuan Ma, Bin Liu, Jinrong He, Shuqin Li, and Hongyan Wang. 2020. A deep-learning-based real-time detector for grape leaf diseases using improved convolutional neural networks. *Frontiers in Plant Science 11*.
35. Xu, Hui. 2018, March. PlantVillage Disease Classification Challenge - Color Images. https://gitlab.com/huix/leaf-disease-plant-village, doi:10.5281/zenodo.1204914.
36. Zhang, Shanwen, Wenzhun Huang, and Chuanlei Zhang. 2019. Three-channel convolutional neural networks for vegetable leaf disease recognition. *Cognitive Systems Research 53*, 31–41.
37. Zhao, Jinling, Yan Fang, Guomin Chu, Hao Yan, Lei Hu, and Linsheng Huang. 2020. Identification of leaf-scale wheat powdery mildew (blumeria graminis f. sp. tritici) combining hyperspectral imaging and an svm classifier. *Plants 9*(8), 936.
38. Zitova, Barbara. 2019. Mathematical approaches for medical image registration.