A COMPARATIVE ANALYSIS OF CONVOLUTIONAL NEURAL NETWORK ARCHITECTURES FOR THE CLASSIFICATION OF TOMATO (*Solanum lycopersicum* L.) FOLIAR DISEASES

Francis Nathanael De Villena
University of Southeastern Philippines
Bo. Obrero, Iñigo St., Poblacion Dist.
Davao City, Davao Del Sur 8000
(+63)921-9160-753
fnodevillena@usep.edu.ph

Kristian Moreno
University of Southeastern Philippines
Bo. Obrero, Iñigo St., Poblacion Dist.
Davao City, Davao Del Sur 8000
(+63)961-341-9314
kmoreno@usep.edu.ph

Kent Cyril Bordios
University of Southeastern Philippines
Bo. Obrero, Iñigo St., Poblacion Dist.
Davao City, Davao Del Sur 8000
(+63)906-495-5067
kcdbordios@usep.edu.ph

ABSTRACT

The proponents compared the accuracy of LeNet-5 and MobileNetV2 convolutional neural network (CNN) architectures and we propose an improved version of LeNet-5 architecture for classifying ten different tomato foliar diseases. The results shown that the proposed improved LeNet-5 model demonstrated a significant improvement on accuracy in contrast to the original architecture and MobileNetV2.

Keywords

Deep Learning, Convolutional Neural Networks, TensorFlow, LeNetV5, MobileNetV2

1. INTRODUCTION

1.1 Background of the Study

The edible fruit of the cultivated tomato plant (*Solanum Lycopersicum* L.) is the world's most substantially consumed vegetable attributable to its commodity as an essential ingredient in a sizable range of raw, cooked, or processed foods. Today, the said species has a cosmopolitan distribution since it has wide climatic tolerance from temperate to tropical regions around the world due to selective breeding, genetic engineering, and advancements in horticulture [1, 5]. Despite being one of the most valuable crops in the world, tomato cultivation is also highly labor-intensive because not only it requires an ample growing environment to flourish but is also susceptible to numerous microbial pathogens and pests in its entire life cycle [1, 2, 3, 5]. Typically, most of them would not kill the tomato plant, still, the damage to its roots, foliage, and fruit can cause a severe deficit in harvests and degradation of fruit quality [1,2]. Pythopathological diseases and disorders are primarily responsible for substantial yield losses and disruption of the global and local tomato supply. Therefore, the aforementioned problems are some of the reasons that commercial tomato production has shifted from open-field agriculture to protected horticulture [5]. The majority of traditional tomato cultivators rely heavily on chemical pesticides and fungicides to protect crops, yet, excessive and indiscriminate use of which leads to the possibility of resistance, increased production expenses, as well as adverse repercussions to human and environmental health [6].

Not to mention, pesticides and fungicides are futile against infectious microbial and viral diseases. Therefore, disease prevention and management strategies should also be taken into account, as these methods can drastically reduce the probability of infection and may mitigate the impact of an outbreak in case it befalls [7]. For instance, opting to grow cultigen tomatoes that manifest tolerance or resistance to specific pests and diseases can be practical when the circumstances are unmanageable, although the chosen variety should adapt to the farm location and produce a competitively marketable yield [1,2]. Another alternative method is the deployment of biological control agents to suppress the population of pests as well as pathogenic microorganisms and abate their disfigurement to crops, whilst generally harmless to the plant and surrounding environment since biocontrol agents only target their natural enemies. However, deployment and operational costs of biological control are expensive, and agents can never eradicate pest organisms otherwise they obliterate their food source [8]. There are other various proven and reliable treatment approaches intended for particular issues, but bear in mind that the effectiveness of these control measures depends on the accurate identification of diseases and their causal agents. Otherwise, misdiagnoses will squander financial resources and effort, and in the worst case, it can also exacerbate yield losses [9,10].

For this reason, the first crucial step in taking action is to investigate the characteristic symptoms, then identify the disease's causal agents or infesting pests, before deciding on which appropriate and cost-effective strategies to implement [9,10]. Early detection can avert disease outbreaks and quell pests in time as well, even though it involves constant monitoring and inspection. Phytopathologists and specialists from related disciplines typically perform tomato plant disease diagnosis and pest identification through on-site observation or isolating samples by conducting further tests in a laboratory if the information gathered from the immediate examination is insufficient [9, 10]. Considering the scale of the farm and the severity of the case, the whole diagnostic process may be time-consuming and costly [10].

The tremendous growth of computer processing power, drastic improvement of digital camera resolution, and enhancement of machine learning algorithms, these advances have enabled deep learning methods to effectively exploit complex, compositional nonlinear functions, learn distributed and hierarchical feature representations, and make effective use of both labeled and unlabeled data [11,12]. In turn, deep learning and computer vision are extensively applied in plant disease and pest classification as they have achieved efficacious performance in both the scientific community and industry particularly in the field of agriculture. Thus, the augmentation of machine learning concepts with image processing to customary diagnostic techniques will generate accurate results on whether the crop is infected or not at instantaneous speeds [13-30].

1.2 Statement of the Problem

Some small-scale tomato growers, especially in the rural areas of the Philippines, might not afford to hire specialists and have no immediate access to assistance at the onset of disease in their crops; they would rather investigate the symptoms and attempt to treat the crops' ailments by themselves. Besides, if they lack proper knowledge of the entomology and phytopathology of tomatoes, they can only draw subjective judgments based on experience and confront the problem through trial and error, which can be quite cumbersome and ineffective. When the treatment is handled erroneously, the disease could spread further and damage to the crops would worsen. Furthermore, in the case of lucrative profit-oriented tomato production in the country, the massive scale of farms made the continual monitoring of crops laborious, requiring more manpower as well as resources. This could be disadvantageous since early detection of diseases and quick response is crucial for reducing damage to crops. Also, commercial tomato growers place importance on minimizing operational expenses to earn more profit. Hence, the proponents observed the need for rapid and reliable diagnosis of insectile, microbial, and viral diseases manifested in the foliage of tomato plants, through deep learning.

1.3 Significance of the Study

This study aims to realistically apply deep learning, particularly the convolutional neural network architectures, in detecting tomato foliar diseases and comparing their performance. The comparison will focus on the accuracy, speed, and efficiency of the algorithms in identifying the disease. The experimental findings are hoped to contribute to the development of practical systems or mobile applications that is focused on the diagnosis of tomato diseases, regardless of tomato cultivar and variety. This will benefit both small-scale and commercial tomato growers in the Philippines to improve their pest and disease management. In addition, the results of the proposed model's performance can also be used as a reference for other studies that are involved in designing a superior neural network for identifying diseases of other economically valuable plants. Lastly, the proponents intended to demonstrate the significance of deep learning to the agricultural sector for Philippine policymakers and agriculturists.

1.4 Scope and Limitations of the Study

The coverage of this study is only exclusive to the diseases that manifest on the foliage of a tomato plant. Diseases and disorders of the tomato's fruit, stem, and root are ruled out. Nevertheless, this study is limited to proposing our neural network model and comparing it to existing pre-trained ConvNet architectures. The external libraries of the Python programming language used for its implementation, such as TensorFlow, might not be available to other languages. Note that in this study there will be no further development of any kind of system or application.

2. REVIEW OF RELATED LITERATURE

2.1 Botany of Tomato

The tomato is a rich source of fiber, potassium, vitamins A, C, and K, and also a dietary source of antioxidants. It contains carotenes, such as lycopene and beta-Carotene that give the fruit predominantly red and orange color, as well as tomatine, an alkaloid with fungicidal properties and its concentration determines the taxonomy of the species. Tomato belongs to the genus Solanum within the large and diverse family Solanaceae, also known as nightshades, which includes several other commercially important species such as potato (Solanum tuberosum L.) and eggplant (Solanum melongena L.). Tomato is a perennial herbaceous plant although it can be biennial under certain conditions and is often cultivated as an annual. It is sensitive to frost and significantly higher temperatures hamper its growth and fruit set. The tomato's growth habit can be indeterminate to determinate, but most commercial varieties focused on large-scale production are determinate while heirloom and greenhouse varieties are indeterminate that can bear fruit during their lifespan. Tomato plants are mostly branched and somewhat trailing when fruiting but some strains are compact and upright. Odorous and hairy glandular trichomes cover the plant's angular stem as well as its leaves that are varied in shape from lobed to compound, with pinnately arranged segments. Varieties with compound leaves consist of petiolate and dentate leaflets. It also has pendent, actinomorphic, five-petaled, yellow flowers blooming in clusters. Its fruits are globular or ovoid depending on varieties, still, they exhibit every characteristic of berries. The bilocular or multilocular cavities of each fruit contain small lens-shaped seeds enveloped by gelatinous pulp. The fruits have thin exocarps and their fleshy tissue comprises the remainder of the pericarp as well as the placenta. In addition, the fruit color is derived from the carotenoid content of the fruits' flesh [1, 2].

2.2 Origin and History of Tomato

The wild ancestors of cultivated tomatoes originated in the western Andes mountain range of South America. They became synanthropic through trade between pre-Columbian civilizations of South America and Mesoamerica. The Aztecs had been domesticating the tomato for food and they called its fruit "tomatl" in their Nahuatl language, which is the etymology of the plant's Hispanicized and Anglicized name. In the 16th century, the Spanish conquistadors introduced the crop's seeds to Spain following their expeditions and conquest of Mexico. As its seeds gradually dispersed from Iberia to several parts of southern Europe, the tomato thrived in the Mediterranean climate. The Crown of Castile started encouraging its production in their lands as well as its distant colonies in the West and East Indies. Meanwhile, the Portuguese maritime explorers introduced the tomato to African and Asian shores through the import of goods from Europe yet the skeptical but curious reception of other parties to the foreign plant rendered it inedible and decorative for a few centuries.

The Spaniards and Italians were the first among the Europeans to adopt the tomato as a staple food and integrate it as an ingredient for their cuisine. The Spaniards and Portuguese called the tomato "tomate", which was derived from the plant's native Nahuatl name, while Italians called the tomato "pomo d'oro" or golden apple, which has given rise to speculation that the first tomatoes known to Europeans were yellow. In the 17th century, the Italian cultivators were the ones who selectively bred the first varieties of tomato with improved yield and rich fruit quality in their gardens. However, the tomato had been propagated solely for ornamental purposes in western and northern Europe and was not regarded as food for several decades. In Great Britain, the fruit was even considered unfit to eat because of the superstitions that the entire plant is poisonous; it was classified as a relative of the deadly nightshade (*Atropa belladonna* L.) and bittersweet nightshade (*Solanum dulcamara* L.).The tomato was then introduced to North America by British colonists as an ornamental plant in the 18th century but its cultivation and selective breeding as a garden vegetable slowly developed several decades later when the Thirteen British Colonies declared their independence from Great Britain. Eventually, the tomato became widely accepted in Europe after the popularity of pizza throughout the continent in 1887. By the late 19th century, the demand for tomatoes in the United States steadily grew after the successful development of commercialized tomato strains and the innovation of lengthening its shelf life via canning. Since then, countless varieties of tomatoes are cultivated extensively in several regions of the world for local consumption and perishable goods exportation; an indirect consequence of colonialism, mercantilism, and westernization begot by the European colonial empires [1, 2].

2.3 Deep Learning

Deep learning is a subset of machine learning that learns to represent the world as a nested hierarchy of concepts, with each concept defined in terms of simpler concepts and more abstract representations calculated in terms of less abstract ones. It enables computer models built of several processing layers to learn data representations with varying levels of abstraction. Furthermore, deep learning identifies sophisticated structures in big data sets by employing the backpropagation algorithm to determine how a machine's internal parameters that are used to calculate the representation in each layer from the representation in the previous layer should be changed [11,12]. A deep learning architecture is a multidimensional stack of basic modules, the majority of which can learn and many of which calculate non-linear input-output mappings. Each module in the stack alters its input to improve the representation's selectivity and invariance. A system with several non-linear layers can design exceedingly complicated functions of its inputs that are sensitive to minute details while being insensitive to significant irrelevant fluctuations. Many deep learning applications employ feedforward neural network topologies, which learn to convert a fixed-size input to a fixed-size output. To advance from one layer to the next, a group of units computes a weighted sum of their previous layer's inputs and passes the result via a nonlinear function [11,12].

2.3.1 Convolutional Neural Networks

Convolutional networks, also known as convolutional neural networks (CNNs or ConvNets), are a type of neural network that processes information using a predefined, matrix topology. It originated in 1980 and was coined “neocognitron” by Kunihiko Fukushima. In at least one of their layers, CNNs utilize a mathematical operation called convolution, hence the name, instead of generalized matrix multiplication. Convolution, in its most generic form, is conducted on the input data of a CNN using a filter or kernel, also known as a feature detector, to generate a feature map. CNNs are often applied for visual imagery, assisting a machine to recognize and learn from digital images. It maps the input image using a matrix of pixel values, and then it performs image recognition and classification based on the numbers it reads, recognizes patterns, and learns from them [12,31].

CNNs have three types of layers: convolutional layers, pooling layers, and full-connected layers. The top layer and essential building block of a ConvNet is the convolutional layer, and it is where the majority of computing occurs. Input data, a filter, and a feature map are all crucial parts of this layer. The feature detector is a two-dimensional array of fixed weights that represents a portion of the image, and its width and height define the scale of the receptive field. The filter is then applied to a portion of the input image, and a dot product between the input pixels and the filter is computed. This dot product is then passed into an array of outputs. Following that, the filter shifts by a stride, which is the number of steps the convolution filter moves each time, and the procedure is repeated until the kernel has swept through the whole image. A feature map, activation map, or convolved feature is the ultimate result of a sequence of dot products from the input and the filter [12, 31].

Padding is done to prevent the feature map from shrinking and having the same dimension as the input since the spatial extent of the feature map is always less than that of the input. Some feature detector parameters, such as weight values, are adjusted during training via backpropagation and gradient descent. CNNs have been able to drastically reduce the number of unique model parameters while also greatly increasing network sizes without requiring a comparable increase in training data owing to parameter sharing. However, four hyperparameters affect the volume size of the output, which must be specified before neural network training commences. These variables include kernel size, filter count, number of strides, and padding type. Ultimately, the convolutional layer encodes the image to numerical values, which the neural network may then analyze and extract significant patterns from [12, 31]

Convolutional layers can pass their feature map with other succeeding convolutional layers or pooling layers. If another convolutional layer is added after the first, the structure of the CNN can become hierarchical since the subsequent layers can see the pixels inside the receptive fields of the preceding layers. Adding a pooling layer after the initial or subsequent convolutional layers, on the other hand, allows for dimensionality reduction, reducing the number of parameters in the input to decrease complexity, enhance efficiency, and lower the danger of overfitting. The pooling process, like the convolutional layer, sweeps a filter across the whole input, but this filter does not contain any weights. Instead, the kernel applies an aggregation function on the receptive field values, filling the output array. This layer aids in the reduction of complexity, the enhancement of efficiency, and the mitigation of the possibility of overfitting. It still employs feature mapping as the output, albeit a lot of information is lost when the output is sent to the next layer. The fully connected layer, the last layer in convolutional networks, precisely characterizes itself. Every neuron in the output layer is fully connected to a neuron in the preceding layer in the said layer, which is comparable to the principle of multilayer perceptron neural networks (MLPs). This is where the image is classified based on the characteristics retrieved from the previous layers and their various filters [12, 31].

The CNN becomes more complex with each layer, recognizing more portions of the image. Earlier layers extract basic features like colors and edges. Deeper layers would then integrate these features, and the final layers would recreate the entire image, beginning to identify bigger elements or forms of the object until it eventually recognizes the particular object. Because CNNs outperform other neural networks in image classification and object recognition, they have been applied in the identification of tomato leaf diseases, obtaining very accurate results as demonstrated by multiple studies and might be adapted to identify other plant diseases [13-30].

2.3.2 LeNet-5

LeNet, better known as LeNet-5 is one of the earliest CNN architectures. It was proposed in 1998 by Yann LeCun et al, in their study entitled “Gradient-Based Learning Applied to Document Recognition". It was initially used for recognizing handwritten and machine-printed characters. They proposed that better pattern recognition systems can be implemented by focusing on automatic learning rather than hand-designed heuristics. LeNet-5 is made up of seven layers. It contains 3 convolutional layers, 2 subsampling layers, and 2 fully connected layers. The first layer is the input layer. It is built to take in an image dimension of 32x32. This architecture utilizes two significant types of layer construct namely convolutional layers and subsampling layers. [32]

2.3.4 MobileNetV2

Howard et al. propounded the next generation of the MobileNet model based on a combination of complementary search techniques as well as a novel architecture design, the MobileNetV3. It is a convolutional neural network that is adjusted to smartphone processors through a combination of hardware-aware network architecture search (NAS) complemented by the NetAdapt algorithm and then subsequently improved through complementary search techniques, new efficient versions of nonlinearities practical for the mobile setting, and new efficient network design. These models are then adapted and applied to the tasks of object detection and semantic segmentation [33].

3. METHODOLOGY

3.1 General Methodological Approach

Action Research design was applied in this study to experiment, analyze and benchmark the performance of the algorithm in identifying the disease ravaging the foliage of the plant, specifically the tomato plant. The proponents utilized the Sequential ConvNet model with the help of Keras and TensorFlow to train, and benchmark the performance of the model in identifying the disease. The proponents will also utilize Jupyter Notebook in implementing the neural network model written mostly with Python. The data processing, modeling, training, and testing procedures were entirely performed in Google Colaboratory to demonstrate our improved convolutional neural network model, namely LeNet-5, and MobileNetV2 architectures. It enables everyone to create and execute arbitrary Python code through the browser and is particularly well suited to machine learning and data analysis. More technically, Colab is a hosted Jupyter notebook service that requires no setup and provides free access to computing resources such as GPUs.

The proponents utilize a very simple approach to Action Research design since it is the best research method to use in this study where we develop, evaluate and analyze a certain method to solve an existing problem. The first step the researchers considered is to identify the problem, every research design starts with identifying and understanding the problem to formulate or create a viable solution. Understanding the problem deeper also helps researchers on creating/formulate a certain specification for the solution. The next step after identifying the problem will be Research Planning, this step involves: formulating theories, collecting literature for reference, collecting data, and exploring viable solutions for the problem. The Action step in the Research design involves building the chosen solution the researchers have deemed viable for the problem. This step involves the training of the model, simulation, evaluation of the model, and benchmarks. After the Action Step, the last step before going to conclusions is the Research Evaluation. The Research Evaluation steps involve checking the results and performance of the model and analyzing the results of if it's a viable solution to the problem. The final step is the conclusions after the model matches the specifications.

3.2 Data Collection

The proponents gathered samples from the PlantVillage dataset [34], retrieved from Sharada Mohanty's GitHub repository [35]. This particular dataset contains a total of 18,162 unique images of tomato leaves, divided into ten categories: two fungal diseases, two bacterial diseases, two Oomycete diseases, two viral diseases, a disease caused by a mite, and a healthy state of a tomato leaf. The identification of the diseases has been determined professionally by an expert plant pathologist who worked with two technicians to do a diagnosis on the said images [35].

Table 1. Overview of PlantVillage Dataset

Class	Category	Samples
0	Bacterial Spot	2027
1	Early Blight	1000
2	Late Blight	1909
3	Leaf Mold	952
4	Septoria Leaf Spot	1771
5	Two-spotted Spider Mites	1676
6	Target Spot	1404
7	Yellow Leaf Curl Virus	5357
8	Mosaic Virus	373
9	Healthy	1591

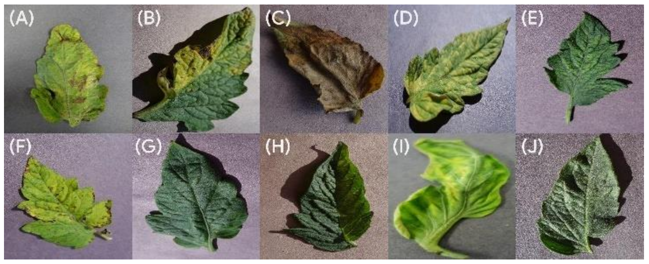


Figure 1. Samples of PlantVillage Dataset

The figure above shows a tomato leaf sample for every category

from top left to bottom right: Bacterial Spot (*Xanthomonas perforans*), Early Blight (*Alternaria solani*), Late Blight (*Phytophthora infestans*), Leaf Mold (*Passalora fulva*), Septoria Leaf Spot (*Septoria lycopersici*), Two-spotted Spider Mites (*Tetranychus urticae*), Target Spot (*Corynespora cassiicola*), Yellow Leaf Curl Virus (*Begomovirus, Geminiviridae.*), Mosaic Virus (*Tobamovirus, Virgaviridae*), and the healthy one.

3.3 Data Processing

Data processing involves the preparation of the dataset for testing and training of the model. For LeNet-5 models, we used a dataset also gathered from PlantVillage but the samples have been already undergone data augmentation techniques [36], so we can only rescale all sample images from 256×256 pixels to 64×64 pixels using the OpenCV library and distribute the samples into new different datasets for training and testing by a 4:1 ratio. The order contents of these datasets are then randomly shuffled. Subsequently, the data are reshaped for the data to be easily handled during the model training. The reshape function from the NumPy library is used in pre-processing the dataset for the model, it works similarly to vectorization, transforming a $m \times n$ matrix into a vector. In mathematics, particularly in linear algebra, vectorization refers to the linear transformation of matrices into a column vector. In the case of reshaping function: instead of transforming the matrix into a column vector, it is instead transformed into a row vector, removing the Transposition superscript. After that, the labels or class of the samples in training and testing datasets are converted to categorical one-hot encoding for model training and prediction.

3.4 Data Analysis

Figures 2 and 3 show the distribution of samples in every category. There are exactly 18,345 samples in the training dataset, while the testing dataset has 4,585 samples.

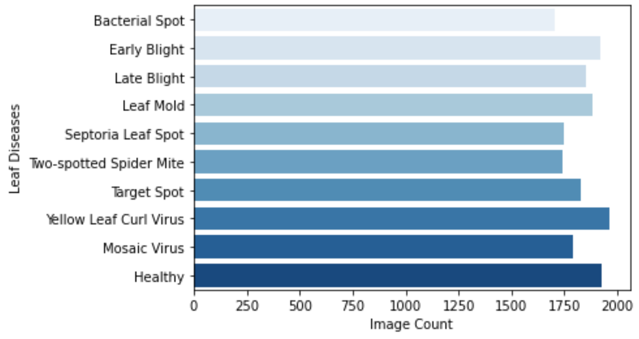


Figure 2. Training Dataset

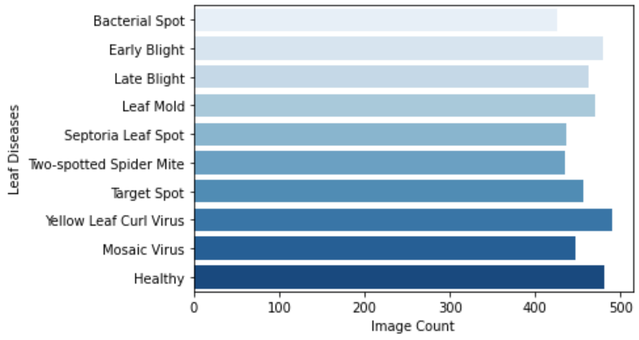


Figure 3. Testing Dataset

3.5 Modeling

3.5.1 LeNet-5

LeNet-5 has comprised of two parts: a convolutional encoder with two convolutional layers and a dense block with three fully connected layers. The architecture is illustrated in the figure below:

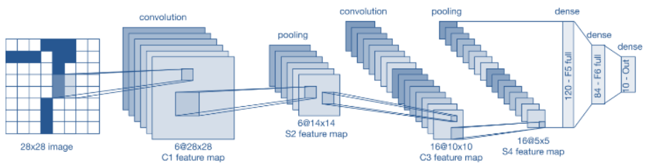


Figure 4. LeNet-5 Architecture

The fundamental components in each convolutional block are a convolutional layer, a sigmoid activation function $S(x) = \frac{1}{1+e^{-x}}$, and a subsequent average pooling operation. To transfer output from the convolution layers to fully connected layers, each sample in the mini-batch is flattened. That is, the four-dimensional input is being transformed into the two-dimensional input that the dense block expects. The desired two-dimensional representation uses the first dimension to index samples in the mini-batch and the second dimension to deliver the flat vector representation of each sample. The dense block from LeNet-5 contains three fully connected layers, each with 120, 84, and 10 outputs. As we deal with the classification problem, the 10-dimensional output layer corresponds to the total classes of tomato foliage and we apply the Softmax function $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ for activation,

The proponents reproduced the original architecture of LeNet-5 according to the published study of LeCun et. al. [32] and implemented it using TensorFlow and Keras. The only difference is 64×64 input shape is specified to the initial layer, which is the first convolutional layer so that the model receives the images corresponding to their resolution and its subsequent layers be able to do automatic shape inference from the initial layer. Table 2 shows the summary of LeNet-5 after compiling the said model with Adam optimization and applying categorical cross-entropy for loss calculation.

Table 2. LeNet-5 Model Summary

Layer	Hyperparameters	Input Shape	Output Shape	# Parameters
Input	-	64×64×3	-	-
Convolution	-	64×64×3	60×60×3	456
Activation	Sigmoid	60×60×3	60×60×3	-
Average Pooling	-	60×60×3	30×30×6	-
Convolution	Sigmoid	30×30×6	26×26×16	2,416
Average Pooling	-	26×26×16	13×13×16	-
Flatten	-	13×13×16	2704	-
Dense		2704	120	324,600
Activation	Sigmoid	120	120	-
Dense		120	84	10,164
Activation	Sigmoid	84	84	-
Dense	Softmax	84	10	850
			Total Parameters	338,486
			Trainable Parameters	338,486
			Non-trainable Parameters	0

3.5.2 Proposed Improved LeNet-5

The proponents devised an improved version of LeNet-5. A third 2-dimensional convolution layer is added and the entire convolution block is defined with the ReLU activation function $f(x) = \max(0, x)$ instead of Sigmoid activation. Each of these convolutional layers has double the filters from the previous one starting from the first layer with 32, 64, and 128, respectively. Succeeding after each convolution layer is an identical 2×2 max-pooling layer, replacing the average pooling layers, to downsample the input along its spatial dimensions; thus, it optimizes the processing on subsequent layers. The dropout layer precedes every max-pooling layer, it randomly selects 20-40% of neurons to ignore during modeling training. This implies that on the forward pass, their contribution to the activation of downstream neurons is eliminated, and any weight changes are not transferred to the neuron on the backward pass. Essentially, dropout prevents overfitting. Next to the last dropout layer is the flattening layer. It transforms the feature map into a long 1-dimensional feature vector that will be then passed into the final phase of the model, the dense layers. We added a fourth dense layer and then specified 64, 128, and 64 units to the first three fully-connected layers. The proponents also replaced the Sigmoid functions with ReLU activation, whereas the final fully-connected layer has a Softmax classifier function $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ for and 10 output shapes.

After defining the layers, the proponents compile the proposed model before training to configure its learning process. Here, we specify some additional properties, namely loss function, and optimizer. For the loss function, we selected categorical cross-entropy since it is suitable for our study that involves multi-class classification and we used one-hot encoded labels for the model. We also chose the Adam algorithm with default values as the optimizer to train the model quickly and efficiently. Then, we added accuracy metrics to report the model's loss and accuracy during training as well as validation. Once the model has been compiled, we trained the model for 30 epochs with 32 batch sizes. The tuples for fitting are the reshaped training dataset and one-hot encoded labels. Moreover, 20% of samples of the training dataset were split for validation since 20% was the split for the test dataset. The following table enumerates the layers of the proposed model.

Table 3. Proposed Improved LeNet-5 Model Summary

Layer	Hyperparameters	Input Shape	Output Shape	# Parameters
Input	-	64×64×3	-	-
Convolution	-	64×64×3	62×62×32	896
Activation	ReLU	62×62×32	62×62×32	-
Maximum Pooling	-	62×62×32	31×31×32	-
Dropout	0.2	31×31×32	31×31×32	-
Convolution	ReLU	31×31×32	29×29×64	18,496
Maximum Pooling	-	29×29×64	14×14×64	-
Dropout	0.2	14×14×64	14×14×64	
Convolution	ReLU	14×14×64	12×12×128	73,856
Maximum Pooling	-	12×12×128	6×6×128	-
Dropout	0.4	6×6×128	6×6×128	
Flatten	-	6×6×128	4608	-
Dense	-	4608	64	294,976
Activation	ReLU	64	64	-
Dense	-	64	128	8,320
Activation	ReLU	128	128	-
Dense	-	128	64	8,256
Activation	ReLU	64	64	-
Dense	Softmax	64	10	650
			Total Parameters	405,450
			Trainable Parameters	405,450
			Non-trainable Parameters	0

3.5.3 MobileNetV2

The model the proponents used in MobileNetV2 uses something called Transfer Learning where the model utilizes a pre-trained MobileNetV2 model from TensorFlow to help train our model. The first thing to do is to instantiate the base model of the MobileNetV2, this is where we activate the pre-loaded model trained in ImageNet, an online image database organized according to the WorldNet hierarchy. The base model is also where we set the input shape to 64, 64, and 3. After setting up the base model, the next step is to create a feature extractor to which we convert our input shape of 64x64x3 into a 7x7x1280 feature block. After creating the feature extractor, the next step is freezing the Convolutional base by setting the trainable layer to FALSE, this way we can maintain a consistent weight during the first phase of the training.

The MobileNetV2 model has a lot of layers so setting the trainable layers to FALSE will temporarily freeze all of them, we do this to not lose what the model has already learned in the Transfer Learning. The model that we are going to train first has the following parameters:

Table 4. MobileNetV2Model Summary

Total Parameters:	2,257,984
Trainable Parameters:	0
Non-Trainable Parameters	2,257,984

After the temporary freezing of the trainable layer, an average pooling layer with a size of 7x7 will be added using GlobalAveragePooling2D by TensorFlow to convert the feature block to a single 1280-element vector per image. A softmax layer will then be added to generate the predictions needed for the model. After all of this is done, we can finally go ahead and compile the model. In compiling the model, we used the Adam algorithm with default optimizer to train the model. We have also used sparse categorical cross entropy for this specific mobilenetv2 model. Once the model is compiled, we trained the model for 30 epochs with 32 batch sizes during the first phase. After the first phase of the training, it can be noticeable that the final accuracy is low for the said model, we will then perform a fine-tuning to increase the performance of the MobileNetV2. In the model fine-tuning, we will unfreeze the trainable layer, the reason why we unfreeze the top layer now is for the specialized features to work with the new dataset and not overwrite the generic learning. The tuned model that we are going to train for the second time will have the following parameters:

Table 5. Tuned Model Parameter Summary

Total Parameters:	2,270,794
Trainable Parameters:	1,874,250
Non-Trainable Parameters	396,544

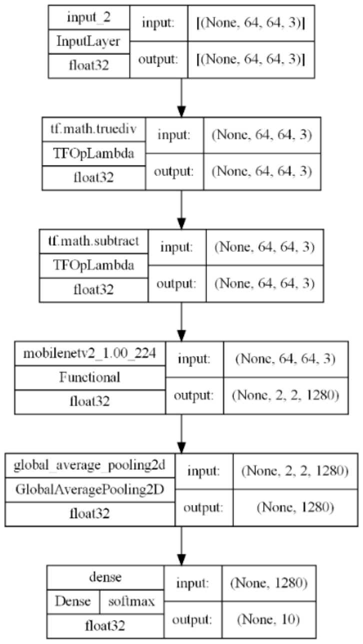

input_2 | InputLayer | float32 | input: [(None, 64, 64, 3)] | output: [(None, 64, 64, 3)]
tf.math.truediv | TFOpLambda | float32 | input: (None, 64, 64, 3) | output: (None, 64, 64, 3)
tf.math.subtract | TFOpLambda | float32 | input: (None, 64, 64, 3) | output: (None, 64, 64, 3)
mobilenetv2_1.00_224 | Functional | float32 | input: (None, 64, 64, 3) | output: (None, 2, 2, 1280)
global_average_pooling2d | GlobalAveragePooling2D | float32 | input: (None, 2, 2, 1280) | output: (None, 1280)
dense | Dense | softmax | float32 | input: (None, 1280) | output: (None, 10)

Figure 5. MobileNetV2 Summary

4. RESULTS AND DISCUSSION

In this chapter, the results of the training process of every model are presented, the proponents assess model performance on both the training and validation dataset and plot the learning curve calculated from the accuracy and loss for each epoch. The shape and dynamics of a learning curve can be used to diagnose the learning behavior and the performance of a ConvNet model, such as fitting and accuracy [37]. Then, the Area Under the Curve (AUC) and Receiver Operating Characteristics (ROC) curves are used by proponents to assess or display the performance of the multi-class classification model. The AUC-ROC curve is a performance metric for classification tasks at various threshold levels. AUC is the degree or measure of separability, whereas ROC is a probability curve. It indicates how well the model can differentiate between classes. The greater the AUC, the better the model predicts a specific class. Moreover, they also implement a confusion matrix, a class-wise distribution of the predictive performance of a classification model—that is, the confusion matrix is an organized way of mapping the predictions to the original classes to which the data belong. The columns represent the original or expected class distribution, and the rows represent the predicted or output distribution by the classifier.

4.1.1 LeNet-5

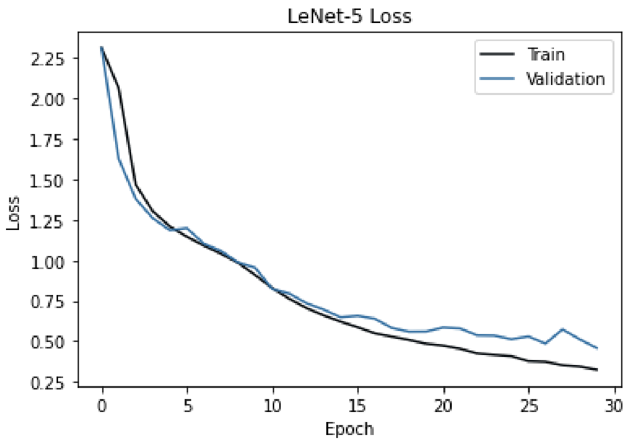


Figure 6. LeNet-5 Optimization Learning Curve

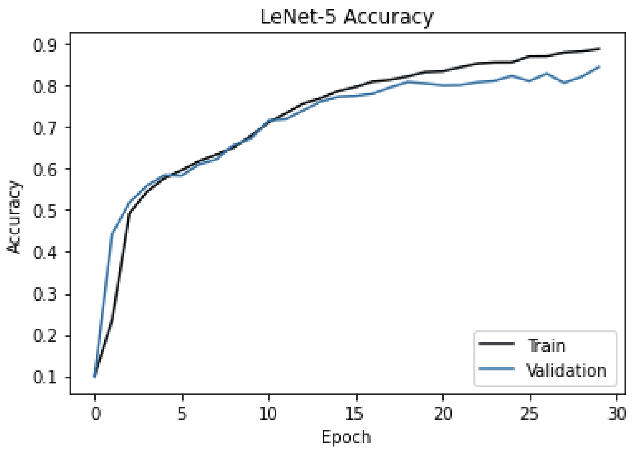


Figure 7. LeNet-5 Performance Learning Curve

As shown in Figure 6, the curve of the training and validation loss has sharply declined to a point of stability and the gap between them is narrow. This implies that the proposed model improves fast as it is training in the beginning but slows down after a few epochs. The validation loss is slightly higher than the training loss as well which means the model has a fairly good fit on the validation dataset. Moreover, the training and validation dataset is suitably representative since there is no significant noise on the learning curve of both training and validation loss. On the other hand, the training and validation accuracy curves rose sharply in the first few epochs and started to plateau later on, as seen in Figure 7.

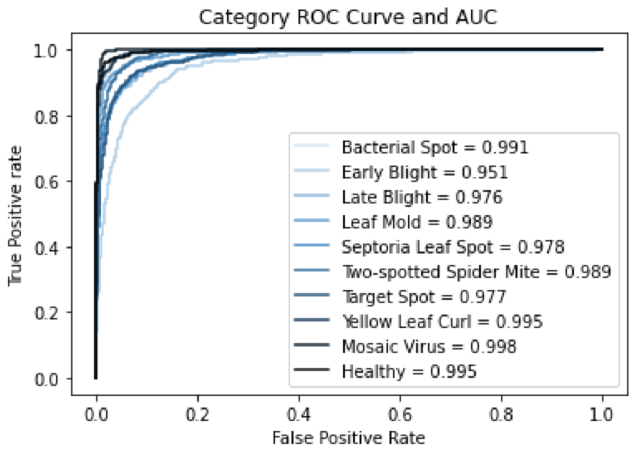
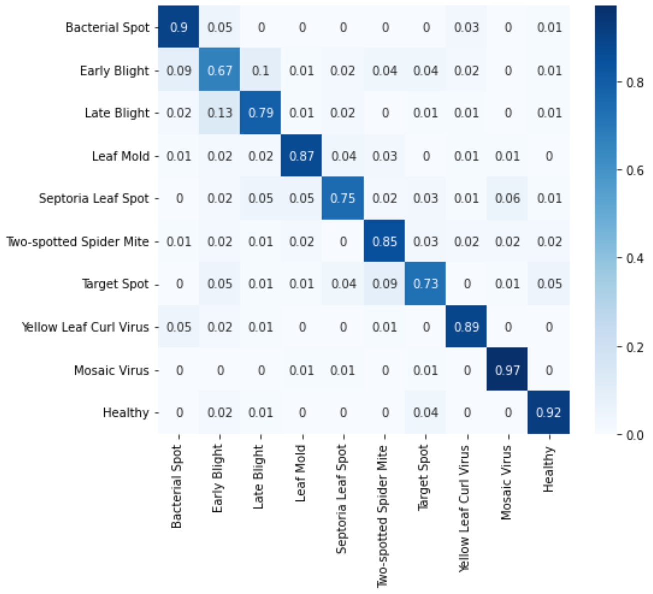


Figure 8. LeNet-5 ROC-AUC



	Bacterial Spot	Early Blight	Late Blight	Leaf Mold	Septoria Leaf Spot	Two-spotted Spider Mite	Target Spot	Yellow Leaf Curl Virus	Mosaic Virus	Healthy
Bacterial Spot	0.9	0.05	0	0	0	0	0	0.03	0	0.01
Early Blight	0.09	0.67	0.1	0.01	0.02	0.04	0.04	0.02	0	0.01
Late Blight	0.02	0.13	0.79	0.01	0.02	0	0.01	0.01	0	0.01
Leaf Mold	0.01	0.02	0.02	0.87	0.04	0.03	0	0.01	0.01	0
Septoria Leaf Spot	0	0.02	0.05	0.05	0.75	0.02	0.03	0.01	0.06	0.01
Two-spotted Spider Mite	0.01	0.02	0.01	0.02	0	0.85	0.03	0.02	0.02	0.02
Target Spot	0	0.05	0.01	0.01	0.04	0.09	0.73	0	0.01	0.05
Yellow Leaf Curl Virus	0.05	0.02	0.01	0	0	0.01	0	0.89	0	0
Mosaic Virus	0	0	0	0.01	0.01	0	0.01	0	0.97	0
Healthy	0	0.02	0.01	0	0	0	0.04	0	0	0.92

Figure 9. LeNet-5 Confusion Matrix

To further ensure the precision of the model, it performed a prediction of the disease of every sample in the test dataset. The Receiver Operator Characteristic (ROC) plot shown in the Figure above illustrates that four among the ten given classifiers of foliage diseases had an area under the curve (AUC) below 0.98, namely Early Blight, Late Blight, Septoria Leaf Spot, and Target Spot, that is to say, they have poorer performance compared to the rest.

The confusion matrix shown in Figure 9 illustrates the efficiency and effectiveness of the classifier on a dataset for which the true values are given. True negatives occur when the class that was predicted to be negative turns out to be negative, such as not predicting a bacterial spot and the result is that it does not detect a bacterial spot. False negatives occur when the classifier predicts negative cases but turns out to be positive, such as predicting no yellow leaf curve but it turns out to detect that said disease. The numbers shaded in blue are the true positives. False positives fall in the same row as the true positive. False negatives fall in the same column as the true positive, and true negatives do not belong in the same row or column as the true positive. Among the ten different conditions, the Mosaic Virus turned out to be the most accurate, in which the calculated accuracy of the classifier is 0.97%, meaning about 97% of the dataset was correctly predicted. It is followed by healthy tomato leaves with 92%. The least accurate class was the early blight, wherein, it's true positive is 0.67. Target Spot and Septoria Leaf Spot have ~0.75 true positives.

4.1.2 Proposed Improved LeNet-5

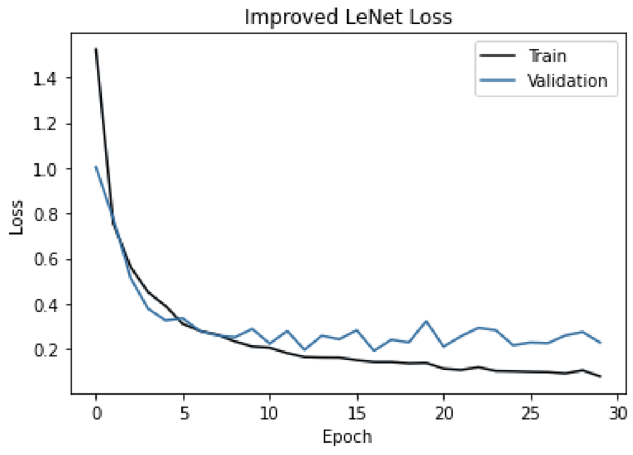


Figure 10. Improved LeNet-5 Optimization Learning Curve

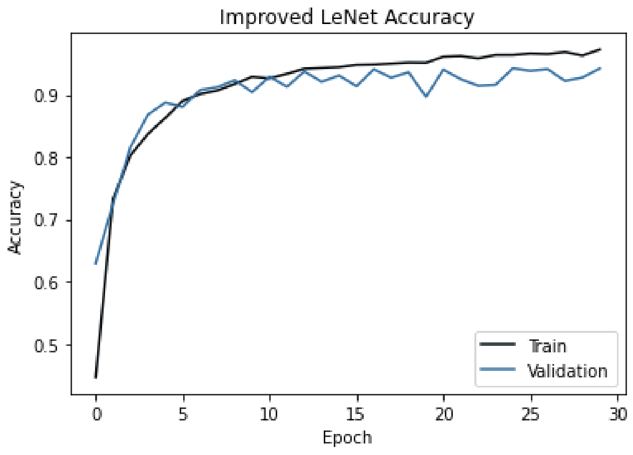


Figure 11. Improved LeNet-5 Performance Learning Curve

As illustrated in Figure 10, the training and validation loss curves declined early on, but the gap between them progressively widened later on. This means that the proposed model improves quickly when training but decelerates after a few epochs. The validation loss is somewhat larger than the training loss, indicating that the model is overfitted to the dataset. Furthermore, the training and validation datasets are sufficiently representative because there is no substantial noise on both the training and validation loss learning curves. The training and validation accuracy curves, on the other hand, climbed dramatically in the first few epochs before plateauing afterward, as seen in Figure 11.

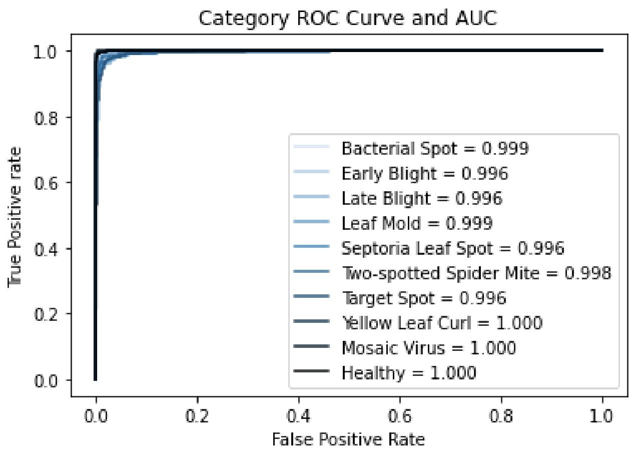
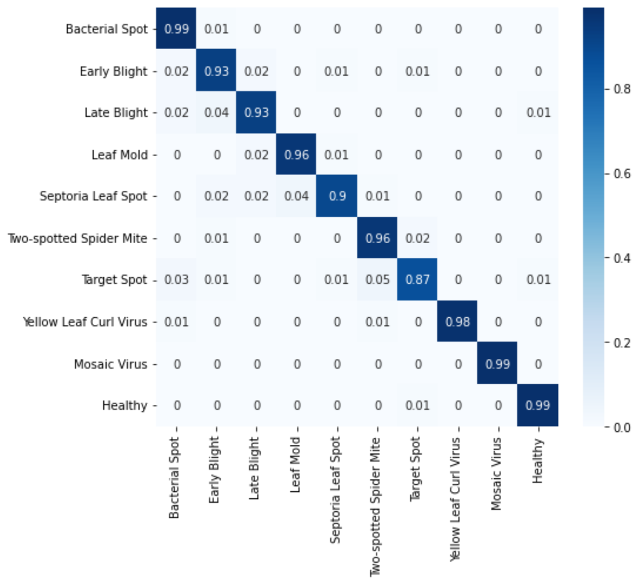


Figure 12. Improved LeNet-5 ROC-AUC

The ROC plot above exemplifies that three among the ten given classifiers of foliage diseases had an area under the curve (AUC) of 1.000 while the rest are above 0.995, meaning these classifiers can seamlessly distinguish between all the positive and the negative class accurately.



	Bacterial Spot	Early Blight	Late Blight	Leaf Mold	Septoria Leaf Spot	Two-spotted Spider Mite	Target Spot	Yellow Leaf Curl Virus	Mosaic Virus	Healthy
Bacterial Spot	0.99	0.01	0	0	0	0	0	0	0	0
Early Blight	0.02	0.93	0.02	0	0.01	0	0.01	0	0	0
Late Blight	0.02	0.04	0.93	0	0	0	0	0	0	0.01
Leaf Mold	0	0	0.02	0.96	0.01	0	0	0	0	0
Septoria Leaf Spot	0	0.02	0.02	0.04	0.9	0.01	0	0	0	0
Two-spotted Spider Mite	0	0.01	0	0	0	0.96	0.02	0	0	0
Target Spot	0.03	0.01	0	0	0.01	0.05	0.87	0	0	0.01
Yellow Leaf Curl Virus	0.01	0	0	0	0	0.01	0	0.98	0	0
Mosaic Virus	0	0	0	0	0	0	0	0	0.99	0
Healthy	0	0	0	0	0	0	0.01	0	0	0.99

Figure 13. Improved LeNet-5 Confusion Matrix

The calculated accuracy of most classifiers is approximately above 0.92, meanwhile, the Target Spot and Septoria Leaf Spot are rather susceptible to false positives and negatives.

4.1.3 MobileNetV2

On the completion of MobileNetV2 model training, the model is then applied to the test dataset to check whether it is the best fit for diagnosing tomato foliage diseases and the corresponding test dataset. The results from the evaluation show that the model achieved a test loss of 42.83% and a test accuracy of 84.78%, therefore, it is viable to identify the diseases of tomato foliage. The reason why the accuracy is low in MobileNetV2 compared to other CNN models is that it is designed to work on mobile phones hence the name MobileNetV2, which has relatively low computational power compared to computers.

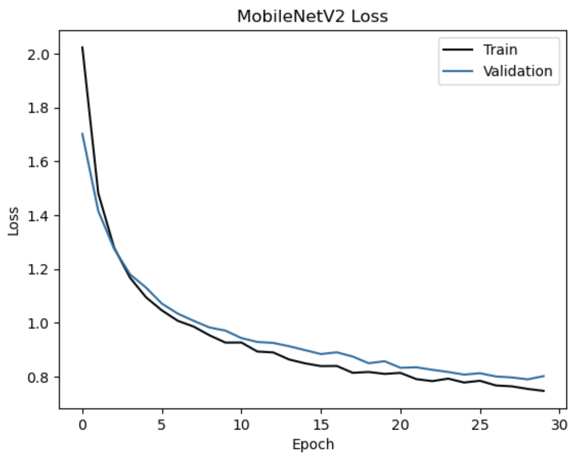


Figure 14. MobileNetV2 Optimization Learning Curve

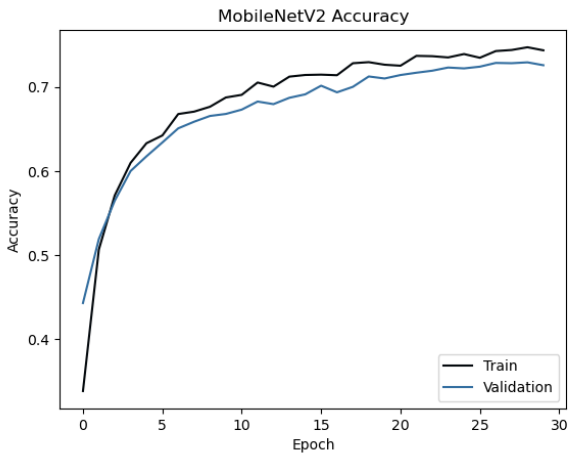


Figure 15. MobileNetV2 Performance Learning Curve

4.1.4 *Final Verdict*

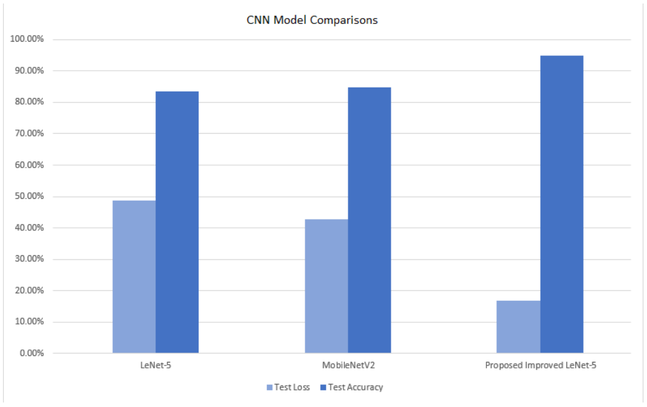


Figure 16. CNN Model Comparisons

According to the evaluations from the learning curve plots, ROC-AUC, and Confusion Matrix from previous sections proved that the proposed improved LeNet-5 is superior than the original LeNet-5 architecture and MobileNetV2. Not only that, as seen in Figure 16, the improved LeNet-5 has the greatest performance with 92.93% accuracy compared to the original LeNet-5 architecture and MobileNetV2 with 83.51% and 84.78%, respectively.

5. SUMMARY, CONCLUSION, AND RECOMMENDATION

According to the results and evaluation of the proposed model, the algorithm's performance in terms of accuracy and efficiency is highly satisfactory and outperforms the other two. The proposed model satisfied the specification and the performance evaluation. Therefore, the proposed deep learning algorithm is better and improved than the other two and can be a more viable solution to the research problem.

In order to produce more accurate results and perform better model training, the image dataset of future research must be of higher quality and hardware specification used to train the models must be higher as well since the training phase of the models requires a lot of computing power.

The farmers must also be well-equipped with proper agricultural knowledge about treating and preventing the spread of the disease as well as plant and crop production. Farmers who are relying only on their experiences without proper scientific knowledge in agriculture would have higher chances of failing or destroying their crops. Agricultural research and extension services are currently offered by state universities in the Philippines, encouraging farmers to attend seminars and training workshops such as handling diseased crops and disposal. Furthermore, additional research must be conducted on integrating the said deep learning algorithm into a mobile phone application, as mobile phones are the most accessible piece of technology anyone can get their hands into.

6. ACKNOWLEDGMENTS

First and foremost, we would like to give thanks to our most gracious and Almighty God who guided us and gave us strength in working on this project and the path of life. Secondly, our parents and relatives have always loved and supported us throughout our lives. Lastly, we would like to thank our course instructor, Mrs. Maureen M. Villamor, PhD, for imparting to us the knowledge of the fundamental lessons of Machine Learning that is considered one of the most important cornerstones of our program, Computer Science, and for advising us throughout this learning evidence.

7. REFERENCES

[1] OECD. 2017. *Safety Assessment of Transgenic Organisms in the Environment, Vol. 1: OECD Consensus Documents. Harmonization of Regulatory Oversight in Biotechnology*, OECD Publishing, Paris, France. DOI: http://dx.doi.org/10.1787/9789264279728-en

[2] Trivedi, M., Singh, R., Shukla, M., and Tiwari, R. 2016. *GMO and Food Security. In Ecofriendly Pest Management for Food Security*, 703-726. DOI: https://doi.org/10.1016/B978-0-12-803265-7.00023-3

[3] Kelley, W., Boyhan, G., Harrison, K., Summer, P., Langston, D. Jr., Sparks, A. Jr., Culpepper, S., Hurst, W., and Fonsah, E. 2017. *Commercial Tomato Production Handbook*. University of Georgia, Athens, GA, USA.

[4] Baranski, R., Goldman, I., Nothnagel, T., and Scott, J. 2016. Improving Color Sources by Plant Breeding and Cultivation. In *Handbook on Natural Pigments in Food and Beverages: Industrial Applications for Improving Food Color*, 429-472. DOI: https://doi.org/10.1016/B978-0-08-100371-8.00019-1

[5] Hanssen, I., and Lapidot, M. 2012. Major Tomato Viruses in the Mediterranean Basin. *Advances in Virus Research*, 84 (2012), 31-66. DOI: https://doi.org/10.1016/B978-0-12-394314-9.00002-6

[6] Ramasamy, S., and Ravishankar, M. 2018. Integrated Pest Management Strategies for Tomato Under Protected Structures. In *Sustainable Management of Arthropod Pests of Tomato*, 313-322. DOI: https://doi.org/10.1016/B978-0-12-802441-6.00015-2

[7] Gajanana, T., Moorthy, P., Anupama, H., Raghunatha, R., and Kumar, G. 2016. Integrated Pest and Disease Management in Tomato: *An Economic Analysis. Agricultural Economics Research Review,* 19, 2 (2006). 269-280. DOI: https://doi.org/10.22004/ag.econ.57763

[8] Holmes, L., Manjiny, S., and Upadhyay, D. 2016. Biological Control of Agriculture Insect Pests. *European Scientific Journal*, 12, 10 (2016). DOI: https://doi.org/10.19044/ESJ.2016.V12N10P%25P

[9] Melanson, R. 2017. *Common Diseases of Tomatoes*. Mississippi State University Extension Service Publication 3175. Mississippi State University, Starkville, MS, USA.

[10] Shankar, R., Harsha, S., and Bhandary, R. 2014. *A Practical Guide to Identification and Control of Tomato Diseases*. Tropica Seeds Pvt Ltd, Bangalore, India.

[11] Goodfellow, I., Bengio, Y., and Courville, A. 2016. *Deep Learning*, MIT Press, Massachusetts Institute of Technology, Cambridge, MA, USA

[12] LeCun, Y., Bengio, Y., and Hinton, G. 2015. Deep learning. *Nature*, 521 (2015), 436-444. DOI: https://doi.org/10.1038/nature14539

[13] Rinu, R., and Manjula, S. 2021. Plant Disease Detection and Classification using CNN. *International Journal of Recent Technology and Engineering*, 10, 3 (2021). 152-156. DOI: https://doi.org/10.35940/ijrte.C6458.0910321

[14] Wu, Y., Xu, L., and Goodman, E. 2021. Tomato Leaf Disease Identification and Detection Based on Deep Convolutional. *Intelligent Automation & Soft Computing*, 28, 2 (2021), 561–576. DOI: https://doi.org/10.32604/iasc.2021.016415

[15] Zhao, S., Peng, Y., Liu, J., and Wu, S. 2021. Tomato Leaf Disease Diagnosis Based on Improved Convolution Neural Network by Attention Module. *Agriculture*, 11 (2021), 651. DOI: https://doi.org/10.3390/agriculture11070651

[16] Tan, L., Lu, J., and Jiang, H. 2021. Tomato Leaf Diseases Classification Based on Leaf Images: A Comparison between Classical Machine Learning and Deep Learning Methods. *AgrilEngineering*, 3 (2021), 542–558. DOI: https://doi.org/10.3390/agriengineering3030035

[17] Gadade, H., and Kirange, D. 2020. Machine Learning Approach towards Tomato Leaf Disease Classification. *International Journal of Advanced Trends in Computer Science and Engineering*, 9 (2020), 490–495. DOI: https://doi.org/10.30534/ijatcse/2020/67912020

[18] Brahimi, M., Boukhalifa, K., and Moussaoni, A. 2017. Deep Learning for Tomato Diseases: Classification and Symptoms Visualization. *Applied Artificial Intelligence*, 31, 4 (2017), 1–17. DOI: https://doi.org/10.1080/08839514.2017.1315516

[19] Agarwal, M., Singh, A., Arjaria, S., Sinha, A., and Gupta, S. 2020. ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network. *Procedia Computer Science*, 167 (2020), 293–301. DOI: https://doi.org/10.1016/j.procs.2020.03.225

[20] Mohanty, S., Hughes, D., and Salathé, M. 2016. Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science*, 7 (2016), 1419. DOI: https://doi.org/10.3389/fpls.2016.01419

[21] Khurana, J., Sharma, A., Chhabra, H., and Nijhawan, R. 2019. An Integrated Deep Learning Framework of Tomato Leaf Disease Detection. *International Journal of Innovative Technology and Exploring Engineering*, 8, 11S (2019), 46-50. DOI: https://doi.org/10.35940/ijitee.K1010.09811S19

[22] Saranya, S., Rajalaxmi, R., Prabavathi, R., Suganya, T., Mohanapriya, S., and Tamilselvi, K. 2021. Deep Learning Techniques in Tomato Plant – A Review. *Journal of Physics: Conference Series*, 1767 (2021). DOI: https://doi.org/10.1088/1742-6596/1767/1/012010

[23] Atabay, H. 2017. Deep Residual Learning for Tomato Plant Leaf Disease Identification. Journal of Theoretical and *Applied Information Technology*, 95, 24 (2017), 6800-6808.

[24] Xian, T., and Ngadiran, R. 2021. Plant Diseases Classification using Machine Learning. *Journal of Physics: Conference Series*, 1962 (2021). DOI: https://doi.org/10.1088/1742-6596/1962/1/012024

[25] Jayanthi, M., and Shashikumar, D. 2020. Automatic Tomato Plant Leaf Disease Classification using Multi-Kernel Support Vector Machine. *International Journal of Engineering and Advanced Technology*, 9, 5 (2020), 560-565. DOI: https://doi.org/10.35940/ijeat.E9689.069520

[26] Ashqar, B., and Abu-Naser, S. 2018. Image-Based Tomato Leaves Diseases Detection Using Deep Learning. *International Journal of Academic Engineering Research*, 2, 12 (2018), 10-16.

[27] Naik, S., Chhajed, P., Trivedi, J., and Davare, S. 2021. Plant Disease Detection and Suggestions using Mobile Application. In *2nd International Conference on IoT Based Control Networks and Intelligent Systems* (ICICNIS 2021). DOI: http://dx.doi.org/10.2139/ssrn.3883334

[28] Zhang, K., Wu, Q., Liu, A., and Meng, X. 2018. Can Deep Learning Identify Tomato Leaf Disease? *Advances in Multimedia*, (2018). DOI: https://doi.org/10.1155/2018/6710865

[29] Zhang, S., Zhou, H., and Zhang, L. 2018. Recent Machine Learning Progress in Image Analysis and Understanding. *Advances in Multimedia*, (2018). DOI: https://doi.org/10.1155/2018/1685890

[30] Liu, J., and Wang, X. 2020. Early Recognition of Tomato Gray Leaf Spot Disease Based on Mobilenetv2-Yolov3 Model. *Plant Methods*, 16, 83 (2020). DOI: https://doi.org/10.1186/s13007-020-00624-2

[31] Wu, J. 2017. *Introduction to Convolutional Neural Networks*. Nanjing University, Nanjing, Jiangsu, China.

[32] LeCun, Y., Haffner P., Bottou, L., & Bengio, Y. 1998. Object Recognition with Gradient-Based Learning. *Proceedings of the IEEE*. DOI: https://doi.org/10.1109/5.726791

[33] Howard, A., Sandler, M., Chu, G., Chen, L., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q., & Adam, H. 2019. *Searching for MobileNetV3*. arXiv:1905.02244v5. Retrieved from https://arxiv.org/abs/1905.02244v5

[34] Hughes, D., and Salathé, M. 2015. An open-access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv:1511.08060. Retrieved from https://arxiv.org/abs/1511.08060

[35] Mohanty, S. PlantVillage Dataset. Github. Retrieved November 1, 2021, from https://github.com/spMohanty/PlantVillage-Dataset

[36] Lamrahi, N. New Plant Diseases Dataset (Augmented). Kaggle. Retrieved December 1, 2022, from https://www.kaggle.com/datasets/noulam/tomato

[37] Viering, T. and Loog, M. 2021. *The Shape of Learning Curves: a Review.* arXiv.2103.10948. Retrieved from https://arxiv.org/abs/2103.10948