






Article

AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf

Hsing-Chung Chen ^{1,2}, Agung Mulyo Widodo ^{1,3}, Andika Wisnujati ^{1,4}, Mosiur Rahaman ¹,
Jerry Chun-Wei Lin ⁵, Liukui Chen ⁶ and Chien-Erh Weng ^{7,*}

- ¹ Department of Computer Science and Information Engineering, Asia University, Taichung City 413, Taiwan; cdma2000@asia.edu.tw or shin8409@ms6.hinet.net (H.-C.C.); agung.mulyo@esaunggul.ac.id (A.M.W.); andikawisnujati@umy.ac.id (A.W.); mosiurahaman@gmail.com (M.R.)
 - ² Department of Medical Research, China Medical University Hospital, China Medical University, Taichung City 404, Taiwan
 - ³ Department of Computer Science, Universitas Esa Unggul, Jakarta 11510, Indonesia
 - ⁴ Department of Mechanical Technology, Universitas Muhammadiyah Yogyakarta, Bantul 55183, Indonesia
 - ⁵ Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, 5063 Bergen, Norway; jerrylin@ieee.org
 - ⁶ School of Intelligent Technology and Engineering, Chongqing University of Science & Technology, Chongqing 401331, China; newque_chen@cqust.edu.cn
 - ⁷ Department of Telecommunication Engineering, National Kaohsiung University of Science and Technology, Kaohsiung City 825, Taiwan
- * Correspondence: ceweng@nkust.edu.tw

Abstract: With limited retrieval of reserves and restricted capability in plant pathology, automation of processes becomes essential. All over the world, farmers are struggling to prevent various harm from bacteria or pathogens such as viruses, fungi, worms, protozoa, and insects. Deep learning is currently widely used across a wide range of applications, including desktop, web, and mobile. In this study, the authors attempt to implement the function of AlexNet modification architecture-based CNN on the Android platform to predict tomato diseases based on leaf image. A dataset with 18,345 training data and 4,585 testing data was used to create the predictive model. The information is separated into ten labels for tomato leaf diseases, each with 64×64 RGB pixels. The best model using the Adam optimizer with a realizing rate of 0.0005, the number of epochs 75, batch size 128, and an uncompromising cross-entropy loss function, has a high model accuracy with an average of 98%, a strictness rate of 0.98, a recall value of 0.99, and an F1-count of 0.98 with a loss of 0.1331, so that the classification results are good and very precise.

Keywords: AlexNet modification; tomato diseases; leaf image; AI



Citation: Chen, H.-C.; Widodo, A.M.; Wisnujati, A.; Rahaman, M.; Lin, J.C.-W.; Chen, L.; Weng, C.-E. AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf. *Electronics* **2022**, *11*, 951. <https://doi.org/10.3390/electronics11060951>

Academic Editor: Hirokazu Kobayashi

Received: 28 December 2021

Accepted: 15 March 2022

Published: 18 March 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Tomatoes grow in almost any moderately dry soil. Nine out of ten farmers grow tomatoes on their farms. One of the efforts in the cultivation of tomato plants is the prevention of diseases that attack these plants, which can cause crop failure. However, sometimes farmers and gardeners are not able to manage the growth of the plants properly [1,2]. Diseases of this plant could be recognized by changes in leaves, roots, stems, and seeds. However, the most easily observed are the changes that occur in tomato leaves. Often, diseases that attack tomato plants cannot be immediately overcome because they must first be examined and analyzed for the type of disease [3]. Errors in treatment can occur, such as detecting the wrong type of disease, causing the treatment method to be incorrect.

With the development of artificial intelligence (AI) technology, especially deep learning with the discovery of the convolutional neural network (CNN) algorithm, it is hoped that disease identification in tomato plants can be carried out based on changes in leaf appearance [4]. Thus, treatment can be carried out immediately according to the type of

disease. CNN is a deep artificial neural network used for image classification, similarity grouping, and presenting object recognition based on picture visualization [2].

The application of CNNs to extract feature characteristics in an interpretable format ensures not only its reliability but also allows validation of model authenticity based on training data without involving individual intuition interference. While some image approaches were used as is, others had to be improved to achieve a limited layer that completely captures the features to produce the appropriate results. In addition, by figuring out the prevented responsiveness maps, we identified several layers that did not contribute to inference and removed these layers from the network, reducing the number of limitations by 75% without affecting classification accuracy [5–8]. Aside from academic research, deep learning is quite common in sectors with a lot of data.

AlexNet is a deep neural network created in 2012 by Alex Krizhevsky et al. [9]. It was created to classify photos for the ImageNet LSFRC-2010 competition, and it came in the first place [10]. It can also handle multiple GPUs. More and deeper neural networks, such as the outstanding VGG and GoogleLeNet, were proposed following that year. The accuracy rate of its official data model is 57.1%, and the top 1–5 have an accuracy rate of 80.2%. For typical machine learning classification algorithms, this is already impressive. AlexNet is a deep convolutional network designed to handle large-scale colored images ($224 \times 224 \times 3$). It has over 62 million trainable parameters in total.

Currently, Android as a platform for mobile phones provides its users with a wealth of smart functions as well as a pleasing appearance. The Android operating system can be used as a multimedia player (music and video). A camera, an accelerometer, a gyroscope, and other sensors are among the hardware included. Furthermore, there are various features that distinguish Android as an operating system worthy of being used by users or developed by developers [11,12]. An Android application with a high-resolution camera could aid in the detection of plant diseases by photographing leaves and sending them off to a cloud server for treatment and detection [13]. However, due to the limited capacity of smartphones, both in terms of memory size and other features, especially cameras, further research is needed in the application of the CNN algorithm in mobile-based applications. An example is implementing CNN with AlexNet architecture in an application that can help tomato farmers identify the type of disease based on the appearance of leaf images from smartphone camera shots. Of course, it requires research related to the CNN architecture that will be used so that the process can still be carried out on smartphone devices.

This motivates the author to create an application that utilizes deep learning embedded in an Android-based application. The goal of this study is to create a mobile-based application that could forecast the disease that a tomato plant would develop based on a picture of the tomato leaves. Thus, this research contributes to:

- Operating deep learning methods, especially the use of the modified AlexNet algorithm as a classifier algorithm.
- Create a mobile-based application where the CNN algorithm with a modified AlexNet architecture is embedded and can be installed and used by tomato farmers.
- Helping tomato farmers easily obtain disease information about the condition of tomato plants based on abnormal leaf appearances through scans or photos of the leaves through an Android-based smartphone platform.

This study is structured as follows: Section 1 discusses artificial intelligence (AI), especially deep learning, the Android platform, and the motivation and contributions of this research. Section 2 explains related work used in this research. Section 3 explains how to develop the application of the AlexNet modified-algorithm-based prediction on an Android platform. Section 4 explains the plot results of the application development, followed by the discussion. Finally, the conclusion of the paper is presented in Section 5.

2. Related Works

To overcome the above problems, authors have developed some respective solutions. In machine learning, different types of characters can be implemented to classify plant

diseases [2]. Automatic detection of diseases from leaves in the plant is an important milestone in the land of agriculture. Furthermore, timely detection of plant diseases has a positive impact on crop yield and quality [2,14]. The requirement of CNN models should have many parameters and higher computational expense. In this work, we replace the common convolution with depth convolution or separable convolution, which reduces the number of parameters and computational cost. The patterns used in this study have been prepared on an accessible data source that included 14 different plant species, 38 different categorical disease classes, and healthy plant leaves. The experimental studies were carried out with the help of the Plant Village Dataset, which includes three pathogens: scar, late scourge, and leaves fungus. The research framework applied the figure discovered by the CNN in various processing hierarchies using consideration methods and achieved an overall rate of 98% on the authentication sets in five-fold cross-validation [1,4,15–17].

Six categories of data enhancement methods were applied: image inversion, gamma correction, noise injection, principal component analysis (PCA), color enhancement, rotation, and scaling [6]. The proposed model [6] achieved a classification accuracy of 96.46% after extensive simulation. The accuracy of the proposed work is higher than the accuracy of traditional machine learning approaches [6]. The data are gathered from the internet and divided into three sets for use in the experiments: a training set, a validation set, and a test set. The experimental outcome shows that the recommended models could correctly and rapidly identify the eleven types of tomato diseases, as well as segment the locations and shapes of infected areas. For classification, original data of 6208 pictures of four types of leaf disease detection was obtained from the Plant Village dataset [18–20]. CNN is regarded as a valuable solution for an extensive kind of image processing briefs, but its architecture is somewhat complex. To reduce the limitations in CNN, a binary explanation coding arrangement based on the improved crossover-based Monarch Butterfly Optimization (ICRMBO) algorithm is proposed [21]. The detection precisions are 86.35%, 99.74%, and 98.54%, respectively, which are greater than those of the deep residual network (ResNet) and the squeeze-and-excitation network (SENet) before enhancement [10,21–23].

The method for detecting tomato leaf spots early, developed based on the MobileNetV2-YOLOv3 methods, realizes better stability between accurateness and actual-time detection of gray leaf spots on tomatoes. F1 score and AP value are used to evaluate the detection effect of the models, and the testing is compared with Faster RCNN and SSD methods. The experiment results show that the proposed model's detection effect is significantly improved [24].

3. Methods

This section explains the steps how to design the Android-based application we proposed, which are generally concentrated on using AI technology to improve the application. Thus, it has been designed according to Human-centered Concept and Machine Learning (HCML) in one or more objects in the lifecycle [25].

Next, for application development, Google Colaboratory (Colab) is used in collaboration with the Python programming language and Integrated Development Environment (IDE), as well as several supporting libraries such as NumPy, Scikit-learn, Pandas, and all of Python's built-in libraries. All these libraries are used to manipulate and engineer raw data for it to match the requirements for input into the main library. The steps carried out in this research are described in Figure 1.

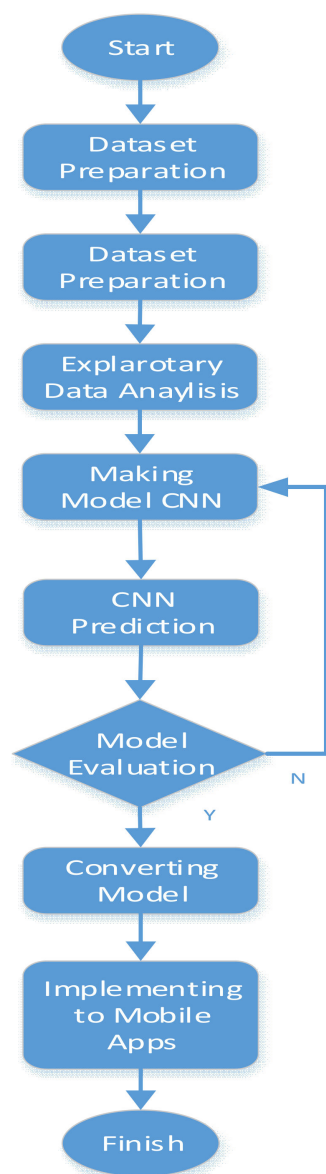


Figure 1. Flowchart of the steps in this research.

3.1. Data Preparation and Exploratory Data Analysis (EDA)

A dataset is a collection of interconnected data that may be viewed individually or in combination as well as maintained as a single entity. A data type is how a dataset is arranged. The dataset [26] consists of the image data of tomato plant leaf diseases used in this research. The dataset is obtained from Kaggle and can be accessed through the application programming interface (API). On the kaggle.com webpage [27], any user could easily obtain the “kaggle.json” API files.

By using the Google Collab IDE, the installation of the Kaggle API can be completed and the dataset loaded. The image data of tomato plant leaf diseases are separated into training data totaling 18,345 photos and test data totaling 4585 images. *Bacterial spot*, *Early blight (scar)*, *Late blight*, *Leaves Mold (fungus)*, *Mosaic virus*, *Yellow Leaf Curl Virus*, *Septoria leaf spot*, *Target spot*, *Healthy*, and *Spider mites* are among the ten forms of tomato leaf diseases found in the dataset. The specifics of the dataset consisting of tested dataset and trained dataset are presented in Table 1.

Table 1. The details of tomato diseases in the dataset [26] consisting of tested dataset and trained dataset.

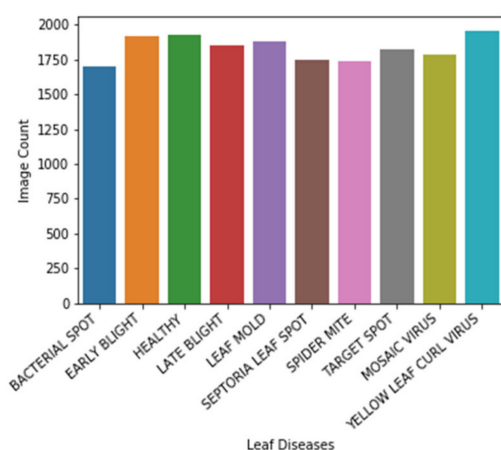
Tomato Disease	Trained Data	Tested Data
<i>Bacterial spot</i>	1702	425
<i>Early blight</i>	1921	481
<i>Late blight</i>	1852	464
<i>Leaves Mold</i>	1881	471
<i>Mosaic virus</i>	1791	449
<i>Yellow Leaf Curl Virus</i>	1962	491
<i>Septoria leaf spot</i>	1744	436
<i>Target spot</i>	1827	457
<i>Spider mites</i>	1741	435
<i>Healthy</i>	1926	481
Total	18,345	4585

Then, the data collected during the exploration phase, EDA, are analyzed. The goal of EDA is to deliver data in such a way that the amount of data in the training and test data, as well as the relationship between variables, can be comprehended. The variables referred to here are the pixel values at the same location of the images that have been transformed along the red, green, and blue axes. The color image is the composite result of the basic colors, such as red, green, and blue.

The process of importing the Python libraries required for EDA is carried out for this purpose. By importing Python libraries and the exploratory data analysis function, the dataset can be divided into two categories of data: training data and valid data, each of which contains 10 different forms of leaf diseases in tomato plants.

The result of the exploratory data analysis function produces a graph which is listed in the form of the total training data set and test data set presented as follows.

Figure 2a shows the total training data from the image of leaf diseases in tomato plants. For example, the most of the data which is came from the yellow leaf curl virus with a total of 1961 images. The least amount of data which is came from bacterial spots with a total of 1702 images. The part of the code for tomato leaf disease detection using CNN [26] is shown in Figure 2b.



(a)

```
Counter({'Tomato_Bacterial_Spot':1702,
        'Tomato_Early_blight':1920,
        'Tomato_Late_blight':1851,
        'Tomato_Leaf_Mold':1882,
        'Tomato_Septoria_leaf-spot':1745,
        'Tomato_Spider_mites Two-spotted_spider_mite':1741,
        'Tomato_Target_Spot':1827,
        'Tomato_Yellow_Leaf_Curl_Virus':1961,
        'Tomato_mosaic_virus':1790,
        'Tomato_healthy':1926 })
```

(b)

Figure 2. (a) Graph of Total Training Data. (b) The part of the code for tomato leaf disease detection using CNN [26].

Furthermore, Figure 3a shows the total valid images data for the leaf diseases on tomato plants. From the picture, it could be seen that most of the image data came from the yellow leaf curl virus, with a total of 490 images. The least amount of data came from

bacterial spots, with a total of 425 images. Figure 3b shows the part of the code for tomato leaf disease detection using CNN [26].

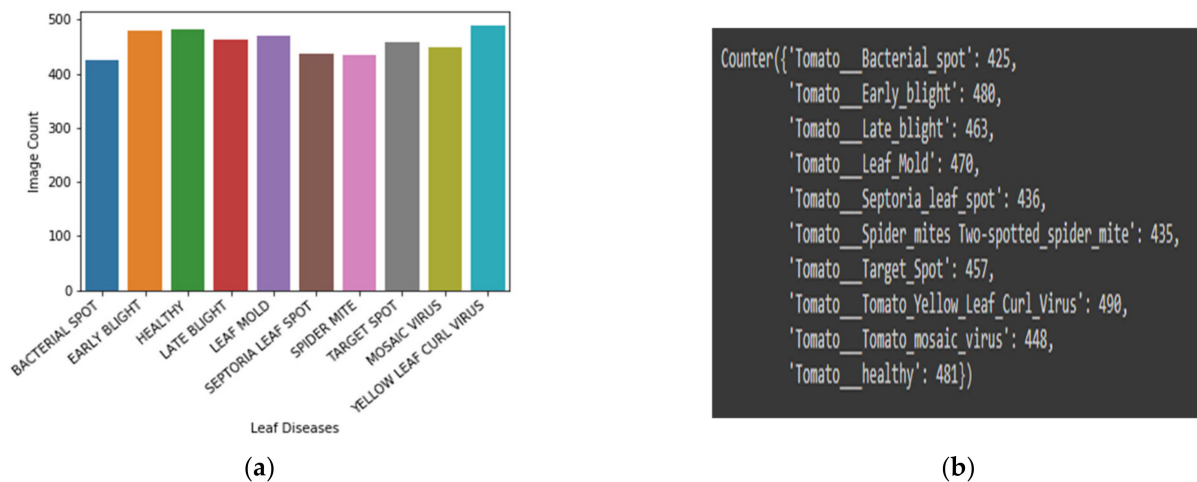


Figure 3. (a) The image count of each tomato leave disease in testing data. (b) The part of the code for tomato leaf disease detection using CNN [26].

3.2. Making the CNN Model

The CNN model was created with the goal of simulating the biological processes that occur in human vision. Multiple layers were used in the networks, which were organized in different ways to create diverse network topologies. However, it can be divided into three different sorts of layers:

- **Layers of Convolution:** The convolution operations are performed using two-dimensional convolution layers using trainable kernels or filters, which may include an optional trainable bias for each kernel. The kernels are moved over the input in “strides” during these convolution operations. The larger the stride, the more space the kernels skipped between each convolution in general. As a result, there were fewer overall convolutions and a smaller output size. A multiplication operation was done between the input section and the kernel for each placement of a given kernel, with the bias summed to the result. It resulted in a feature map with the convolved outcome. To give input for the next layer, the feature maps were usually routed through an activation function. The output size of the feature map was calculated by Equation (1), where N is the input size, F is the kernel size, P is the padding, and S is the stride.

$$\text{Output size} = \frac{N - F + 2P}{S} + 1 \quad (1)$$

- **Layers of subsampling:** Non-trainable kernels or windows are used in two-dimensional subsampling layers to down-sample input characteristics. This often minimizes the amount of features and helps to eliminate a network’s reliance on position. Average pooling and maximal pooling were the two most used kinds of subsampling. The average or maximum of the values present in each kernel to be included in the resultant feature map is computed by both approaches. For subsampling layers, the feature map size was determined in the same way as for the convolution layers. To aid with overall model learning, some implementations of these layers incorporate certain trainable parameters.
- **Fully Connected Layers:** CNNs had the only fully connected layers. They are usually found in the last few layers of most CNNs, appearing after multiple convolution and subsampling processes. The fully connected layer has several hidden layers, an activation function, an output layer, and a loss function. Their operations consisted of multiplying their inputs by trainable weight vectors with a trainable bias and

summing the outcomes. Activation functions, similar to convolution layers, were used to deliver the output of these layers in the past. To avoid overfitting, CNN has dropped regularization. A dropout will randomly assign a value of 0 to a neuron during the backpropagation and forward learning processes. This is a simple technique so that the neural network does not get caught up in overfitting.

This section describes the CNN design used in this research, which includes dataset separation and CNN learning.

- **Dataset separation** The obtained dataset is separated into two portions, training data and test data, prior to CNN training. Data that becomes CNN's learning material is referred to as training data. The training data has a different composition than the test data. Using the 10-fold cross-validation approach, the training data is separated into two subsets, a training subset, and the validation subset, after being loaded into the software. The best CNN model will be reloaded when the learning process is complete to predict from each test dataset.
- **CNN learning** It is crucial to determine which CNN architecture will be employed before starting the learning process. The highest performance will come from an architecture that is appropriate for the dataset. The CNN design used in this study is basically a modification of the AlexNet architecture. The choice of this architecture is based on the good performance of this architecture but also its applicability to platforms with limited capacities, such as Android mobile. The architecture consists of three convolution layers, three fully connected layers, and one output layer. The gradient descent method, which is included in the parameters employed in the neural network, is the most significant algorithm in the learning process (optimizer). The weights of each neuron will be updated using this algorithm. There are many types of optimizers in deep learning, such as *adagrad*, *sgd*, *adadelta*, *adam*, *rmsprop*, and others. In this research, the Adam optimizer was used with a learning rate of 0.005. Adaptive moment estimation (Adam) is a method that calculates the adaptive learning rate for each parameter. Adam keeps the decay rate exponentially on the previous gradient m_t , similar to momentum:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (2)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (3)$$

Equations (2) and (3) show that m_t and v_t are estimates of the first moment (mean) and second moment (decentralized variance) of each gradient. Then, the final formula for the parameter is updated to:

$$\theta_t + 1 = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} - \epsilon} \hat{m}_t \quad (4)$$

The value of β_1 is 0.9, β_2 is 0.9999, and ϵ is 10^{-8} .

3.2.1. The Architecture of CNN Is based on Alex Net Modification

The CNN in this study is based on the AlexNet architecture which has eleven layers. However, in this research, the authors modified by fewer layers. The AlexNet modification architecture in this research is made up of six layers: three convolution layers, three fully related levels, and one output layer [28]. Figure 4 describes the AlexNet architecture modification used in this research.

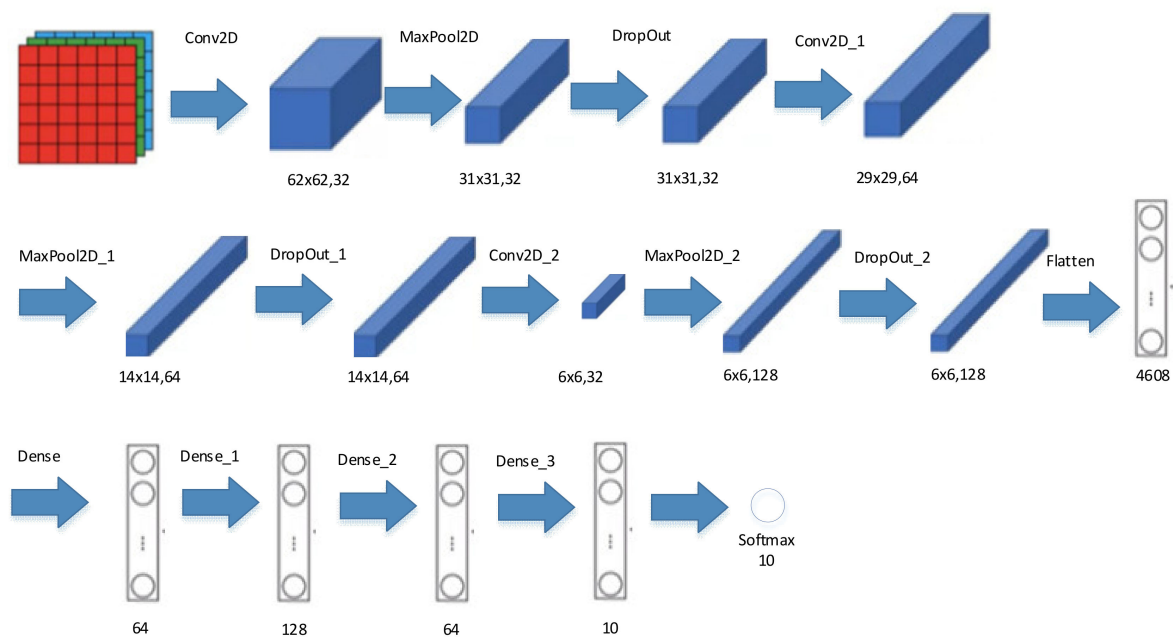


Figure 4. The AlexNet modification architecture.

The size of the input image is 64×64 RGB pixels. In the first convolution layer's filter, which is 32 with a kernel size of 3×3 , the rectified linear unit (ReLU) initiation operation is applied. ReLU is an activation function, which is shown by Equation (5).

$$f(x) = \max(0, x) \quad (5)$$

The activation function is a non-linear transformation performed on the signal. The output of this transformation is then sent from a neuron as an input to the next layer. It runs with a threshold value of 0. If x is less than zero, the value of $f(x)$ is zero; otherwise, the value of $f(x)$ is the value of x itself. The first max pooling layer after an initial Conv2D layer is 2×2 with a dropout value of 0.2. The ReLU activation function is also used in the secondary convolution layer, which has 64 filters with a kernel size of 3×3 .

The first convolution layer's output and the prior pooling are fed into the second convolution layer. The pooling size in the second max pooling layer is 2×2 with a dropout value of 0.2. The ReLU activation function is used in the third convolution layer, which has a filter size of 64 and a kernel size of 3×3 . The second convolution layer's output and the prior pooling are fed into the third convolution layer. In the second max pooling layer, the pooling size is also 2×2 with a dropout value of 0.4. The ReLU activation function is used in the third convolution layer, which has a filter size of 128 and a kernel size of 3×3 .

The AlexNet architecture for all datasets up to the flattening layer produces the same number of inputs and outputs. The output of this flatten layer will be used for the input of the dense layer. Table 2 shows the specifications of CNN with the AlexNet architecture used in the implementation of the leaf disease detection system in tomato plants. The neurons in the first hidden layer are 64 units, the second hidden layer are 128 units, and the third hidden layer are 256 units. The activation function used in the hidden layer is ReLU. One output with a hidden layer is as much as the number of existing classes, namely, 10 units with a Softmax activation function.

Softmax serves to distribute learning results or predictions from previous layers in the range $[0, 1]$, with the result of the total value of all elements being 1. The highest value produced by Softmax is the class predicted by CNN. The formula of the Softmax function

can be seen in Equation (6), where j is the index of each element z and N is the total number z of all elements. The Softmax equation [29] is translated into Equation (6).

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^N e^{z_k}}, j = \{1, 2, \dots, N\} \quad (6)$$

The architecture employed in this study was determined through multiple experiments based on the number of layers to be used to find the best CNN model.

Table 2. AlexNet Architecture.

Layer (Type)	Output Shape	Parameters
Conv2D	(None, 62, 62, 32)	896
MaxPooling2D	(None, 31, 31, 32)	0
Dropout	(None, 31, 31, 32)	0
Conv2D_1	(None, 29, 29, 64)	18,496
MaxPooling2D_1	(None, 14, 14, 64)	0
Dropout_1	(None, 14, 14, 64)	0
Conv2D_2	(None, 6, 6, 32)	73,856
MaxPooling2D_2	(None, 6, 6, 128)	0
Dropout_2	(None, 6, 6, 128)	0
Flatten	(None, 4608)	0
Dense	(None, 64)	294,976
Dense_1	(None, 128)	8320
Dense_2	(None, 64)	8256
Dense_3	(None, 10)	650

3.2.2. Implementation of CNN Based on AlexNet Modification Architecture

After the data set is downloaded and the EDA process is carried out, it is continued with the implementation of the modified AlexNet-based CNN Architecture. Because the architecture in this research is a modification of the existing AlexNet model, the authors tried to implement it starting from the use of 1 layer up to 4 layers, which is the most optimal architecture in this CNN model.

The modified AlexNet implementation uses three convolution layers, three fully connected layers, and one output. The functions used in layer convolution are ReLu with dropout values of 0.2, 0.2, and 0.4. Meanwhile, the output layer uses the Softmax activation function [29,30] and the number of the dense layer.

After the model is made, the next step is to compile it with three parameters, namely: optimizer, loss, and metrics. By initializing the Adam optimizer, the implementation of the model compilation is started.

The implementation of the training model is carried out by creating the steps per epoch argument, which contain the value of 75 data points for one epoch. Then there is the batch size argument. The batch size argument is the number of images entered in each training step, which is 128. Finally, there is a validation split argument. A validation split is a validation technique by dividing the data into two parts randomly. The authors make a value of 0.2 for the validation split.

3.3. Prediction and Evaluation Using CNN Model

This step is carried out to measure the ability of the best CNN model to predict from the results of previous training. The CNN model was given input from the test data. For each incoming leaf disease image, the CNN output layer will give the probability of each class using the Softmax activation function formula [29,30]. The result of CNN is an i -th neuron with the highest output probability value prediction.

The next step is to measure the performance of the model. At this stage, the prediction results of the model are compared with ground truth. To make it easier, a confusion

matrix was made to calculate the accuracy, precision, recall, and F-measure values for each evaluation model.

3.3.1. Implementation of CNN Prediction Model

Implementation of model predictions is carried out to measure the ability of the training model to produce results. The Keras library has provided a function to predict the model, where its name is *predict* (·) function [31]. The input image is loaded with a size of 64×64 to be converted into an array. Then, the predicted value and prediction result class are calculated.

3.3.2. Implementation of CNN Evaluation Model

The model is evaluated to determine the loss and accuracy values that are contained in the model. The implementation of the model evaluation is carried out by using the *evaluate* (·) function in Keras libraries [31].

The model prediction outcomes are displayed in the form of accuracy, precision, recall, and F-measure in the confusion matrix generated during the training.

3.4. Converting Model

The final step is to convert the CNN model so that the CNN model becomes an entity that can provide predictive results from live data. Then, the CNN model is converted to Tensorflow Lite [31].

Lite is a set of tools that enable on-device machine learning by helping developers run their models on mobile devices. Before converting, the model is saved first to a directory with the extension h5 format model [31].

3.5. Implementing to Mobile Application

Leaf disease image recognition in tomato plants (image recognition) is implemented into a mobile-based application. This application is implemented using the Android Image Recognition with Deep Learning library, in which there are algorithms and other libraries related to the image recognition process of leaf diseases in tomato plants. The activities carried out at this stage are as follows:

- Create user interfaces and user experiences.
- Create use case diagrams.
- Create activity diagrams.
- Create class diagrams.

4. Results and Discussion

The data were obtained by photographing the object of tomato leaves suspected of having the disease using a mobile phone as shown in Figures 5 and 6.

Next, by using the input data processing procedure, the trial begins by transforming the RGB image to a vector image. To check for correctness, the classification results were compared with the leaf disease database's training data on tomato plants. The accuracy, precision, recall, and F-measure values of the classification findings are evaluated as part of the truth test. The epoch value, the optimizer, the batch size, the loss function, and the learning rate used are all parameters used in the pilot scenario in this study. The test parameters are listed in Table 3. Based on various research, the learning rate value of 0.001 was chosen. The best value of the model was found in the learning rate range of 0.0005 to 0.001. Because the dataset employed is a multi-label dataset, definite cross entropy was chosen as a loss function.



(a)



(b)



(c)

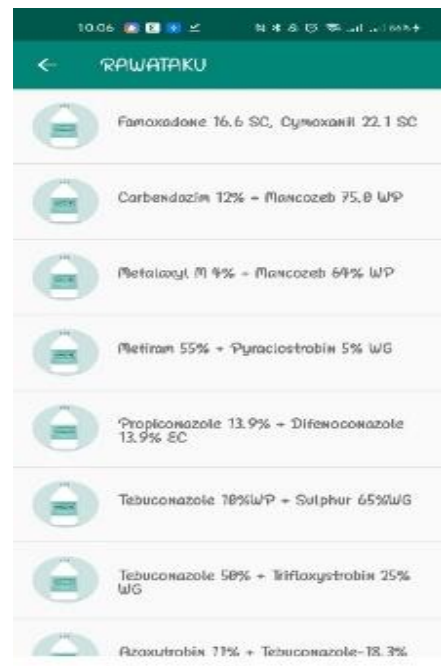


(d)

Figure 5. Cont.



(e)

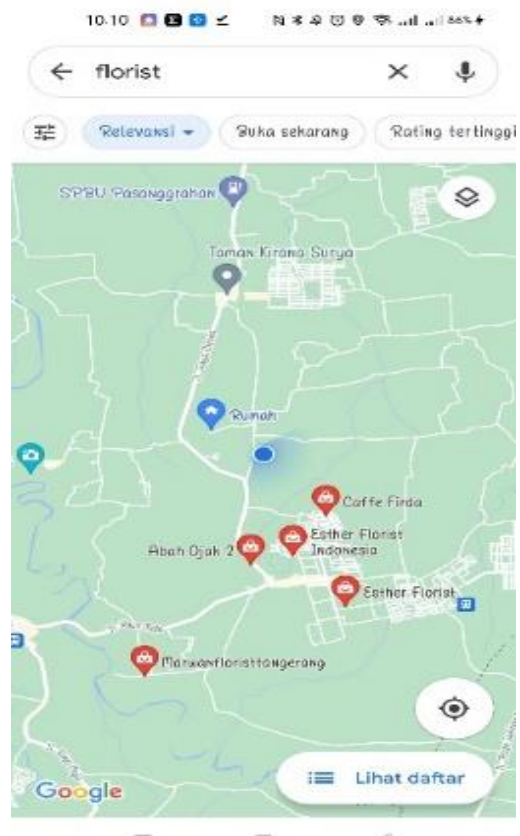


(f)

Figure 5. Mobile apps user interface. (a) Welcome page; (b) login page; (c) registry page; (d) home page; (e) disease recognition page; and (f) disease treatment page.



(a)



(b)

Figure 6. Mobile apps user interface. (a) Disease detection page; (b) shop search page.


Table 3. The Parameters.

Number of Epochs	75
Optimizer	Adam
Batch size	128
Loss function	Categorical cross entropy
Learning rate	0.0005


Some parts of the test results of cases by using the built application are shown in Table 4.

Table 4. Some parts of the test results were outputted from the built application by using the valid dataset [27].

Case 1: Through dealing with the valid testing data, this case is identified the tomato with Bacterial Spot disease. The result shows that the accuracy rate equals 0.9999993.

Image with JPG Size 64 × 64	:	
Classification result	:	Bacterial Spot
Accuracy	:	0.9999993
Remark	:	It matched to the valid data because the accuracy equals to 1.



Case 2: This case is dealing with the valid data in the testing phase for identifying the tomato with Spider mites disease. The result shows that the accuracy rate equals 0.9893751.

Image with JPG Size 64 × 64	:	
Classification result	:	Spider_mites (with Two-spotted_spider_mites)
Accuracy	:	0.9893751
Remark	:	It matched to the valid data because the accuracy equals to 1.

Case 3: This case is identified the tomato with Yellow Leaf Curl Virus disease via dealing with the valid testing data,. The result shows that the accuracy rate equals 0.99838257.

Image whit JPG Size 64 × 64	:	
Classification result	:	Yellow Leaf Curl Virus
Accuracy	:	0.99838257
Remark	:	It matched to the valid data because the accuracy equals to 1.

Table 4. Cont.

<p>Case 4: This case is dealing with the valid data in the testing phase for identifying the tomato with no disease (Healthy). The result shows that the accuracy rate equals 1.</p>	
Image with JPG Size 64 × 64	
Classification result	: Healthy Leaf
Accuracy	: 1
Remark	: It matched to the valid data because the accuracy equals to 1.
<p>Case 5: This case is identified the tomato with Septoria leaf spot disease through dealing with the valid testing data,. The result shows that the accuracy rate equals 1.</p>	
Image with JPG Size 64 × 64	
Classification result	: Septoria leaf spot
Accuracy	: 1
Remark	: It matched to the valid data because the accuracy equals to 1.

The performance of the model using the optimizer on each dataset is utilized as the test parameter. The accurateness, precision, recall, and f-measure confusion matrix are used to test the model’s ability. Figure 7 shows the confusion matrix to predict the model.

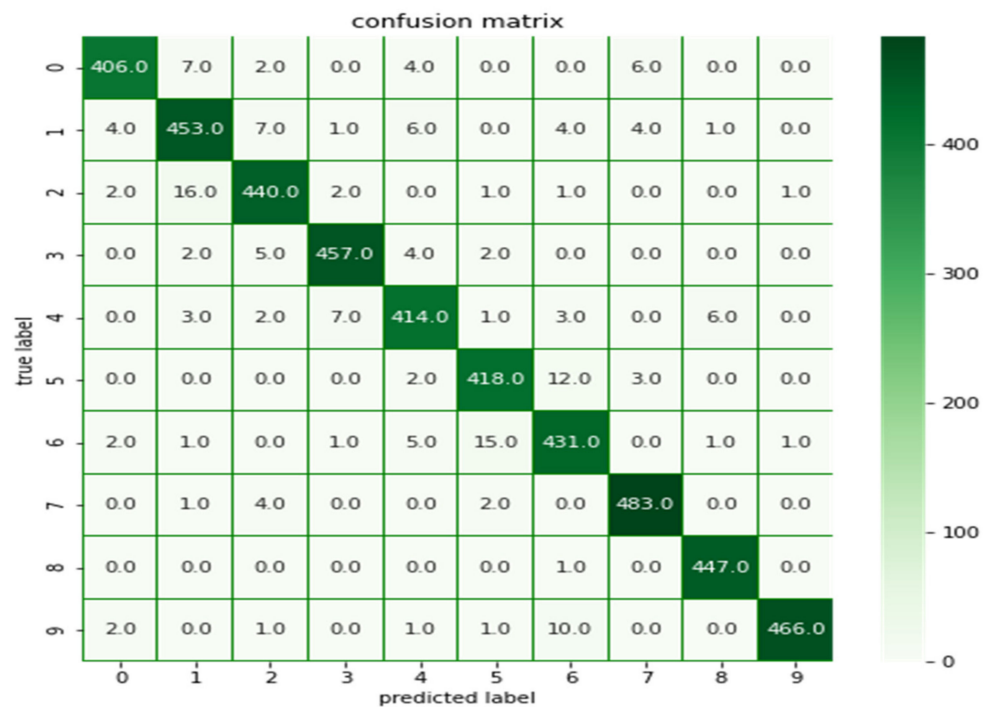


Figure 7. The Confusion Matrix.

The description of the confusion matrix on the ability of model to predict the type of disease is shown in Table 5.

Table 5. Precision, Recall, and F1-Score Values of the Model.

No	Disease	Precision	Recall	F-Measure	Accuracy
1	Bacterial Spot	0.98	0.96	0.97	
2	Early blight	0.94	0.94	0.94	
3	Late blight	0.95	0.95	0.95	
4	Leaves mold	0.98	0.97	0.97	
5	Septoria leaf mold	0.95	0.95	0.95	
6	Spider mites	0.95	0.96	0.96	0.96
7	Target spot	0.93	0.94	0.94	
8	Yellow leaf curl virus	0.97	0.99	0.98	
9	Tomato mosaic virus	0.98	1	0.99	
10	Healthy	1	0.97	0.98	

5. Conclusions

According to various trials conducted in this research, the preprocessing approach and classification method by using the CNN algorithm with an architecture based on the modified AlexNet can be relied upon to determine the correctness of object image classification. The average accuracy value of 96%, precision value of 98%, recall value of 95%, and F-Measure value of 97% demonstrate this. This algorithm can be embedded in applications that have a mobile-based platform, which in this research uses Android as a platform. This is very appropriate because of the limited memory capacity of Android-based gadgets. On the other hand, the results of this study are expected to be useful for farmers or planters of tomatoes. Users can identify the types of diseases suffered by these plants in a timely manner by uploading photos of the leaves via Android gadgets. In this way, it is possible to determine how to manage disease at an early stage.

More experimental setups and structural modifications to other CNN architectures that perform better than AlexNet, such as GoogLeNet, ResNet, etc., are recommended so that they can be embedded in mobile-based applications in future research to accelerate the training process and generate faster classifications and reliable predictions.

Author Contributions: Conceptualization, H.-C.C., A.M.W. and C.-E.W.; methodology, H.-C.C. and A.M.W.; software, A.M.W. and M.R.; validation, A.M.W. and A.W.; formal analysis, A.M.W. and A.W.; investigation, M.R.; resources, H.-C.C., C.-E.W., J.C.-W.L. and L.C.; data curation, A.M.W., A.W. and M.R.; writing—original draft preparation, A.M.W., H.-C.C. and A.W.; writing—review and editing, H.-C.C., A.M.W., A.W., M.R., J.C.-W.L., L.C. and C.-E.W.; visualization, M.R. and J.C.-W.L.; supervision, H.-C.C., J.C.-W.L., L.C. and C.-E.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Ministry of Science and Technology (MOST), Taiwan, under MOST grant numbers below. 110-2218-E-468-001-MBK, 110-2221-E-468-007, and 110-2218-E-002-044. This work was also supported in part by the Ministry of Education under grant number. I109MD040. This work was supported in part by Asia University, Taiwan, and China Medical University Hospital, China Medical University, Taiwan, under grant nos. ASIA-110-CMUH-22, ASIA-108-CMUH-05, ASIA-106-CMUH-04, and ASIA-105-CMUH-04. This work was also supported in part by Asia University, Taiwan, UMY, Indonesian, under grant no. 107-ASIA-UMY-02. This work was also supported in part by Asia University, Taiwan, UMY, Indonesian, under Grant 107-ASIA-UMY-02. This study was also supported in part by the Universitas Muhammadiyah Yog-yakarta, Indonesia.

Acknowledgments: This work was supported by the Ministry of Science and Technology (MOST), Taiwan, under MOST grant numbers below. 110-2218-E-468-001-MBK, 110-2221-E-468-007, and 110-2218-E-002-044. This work was also supported in part by the Ministry of Education under grant number. I109MD040. This work was supported in part by Asia University, Taiwan, and China Medical University Hospital, China Medical University, Taiwan, under grant nos. ASIA-110-CMUH-22, ASIA-108-CMUH-05, ASIA-106-CMUH-04, and ASIA-105-CMUH-04. This work was also supported in part by Asia University, Taiwan, UMY, Indonesian, under grant no. 107-ASIA-UMY-02. This work was also supported in part by Asia University, Taiwan, UMY, Indonesian, under Grant 107-ASIA-UMY-02. This study was also supported in part by the Universitas Muhammadiyah Yog-yakarta, Indonesia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Agarwal, M.; Singh, A.; Arjaria, S.; Sinha, A.; Gupta, S. ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network. *Proc. Comput. Sci.* **2020**, *167*, 293–301. [CrossRef]
2. Hassan, S.; Maji, A.; Jasiński, M.; Leonowicz, Z.; Jasińska, E. Identification of Plant-Leaf Diseases Using CNN and Transfer-Learning Approach. *Electronics* **2021**, *10*, 1388. [CrossRef]
3. Chen, J.; Chen, J.; Zhang, D.; Sun, Y.; Nanekaran, Y.A. Using deep transfer learning for image-based plant disease identification. *Comput. Electron. Agric.* **2020**, *173*, 105393. [CrossRef]
4. Karthik, R.; Hariharan, M.; Anand, S.; Mathikshara, P.; Johnson, A.; Menaka, R. Attention embedded residual CNN for disease detection in tomato leaves. *Appl. Soft Comput.* **2020**, *86*, 105933.
5. Arya, S.; Singh, R. A Comparative Study of CNN and AlexNet for Detection of Disease in Potato and Mango leaf. In Proceedings of the 2019 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), Ghaziabad, India, 27–28 September 2019; IEEE: Piscataway, NJ, USA, 2019.
6. Geetharamani, G.; Pandian, A. Identification of plant leaf diseases using a nine-layer deep convolutional neural network. *Comput. Electr. Eng.* **2019**, *76*, 323–338.
7. Toda, Y.; Okura, F. How convolutional neural networks diagnose plant disease. *Plant. Phenom.* **2019**, *2019*, 9237136. [CrossRef]
8. Verma, S.; Chug, A.; Singh, A.P.; Sharma, S.; Rajvanshi, P. Deep learning-based mobile application for plant disease diagnosis: A proof of concept with a case study on tomato plant. In *Applications of Image Processing and Soft Computing Systems in Agriculture*; IGI Global: Hershey, PA, USA, 2019; pp. 242–271.
9. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
10. Gao, R.; Wang, R.; Feng, L.; Li, Q.; Wu, H. Dual-branch, efficient, channel attention-based crop disease identification. *Comput. Electron. Agric.* **2021**, *190*, 106410. [CrossRef]
11. Chen, L.; Wang, J.; Yang, S.; He, H. A Finger Vein Image-Based Personal Identification System with Self-Adaptive Illuminance Control. *IEEE Trans. Instrum. Meas.* **2016**, *66*, 294–304. [CrossRef]
12. Chen, L.; Chen, H.-C.; Li, Z.; Wu, Y. A fusion approach based on infrared finger vein transmitting model by using multi-light-intensity imaging. *Human-Centric Comput. Inf. Sci.* **2017**, *7*, 35. [CrossRef]
13. Anas, S.; Badhusha, I.; Zaheema, O.T.; Faseela, K.; Shelly, M. Cloud based automated irrigation and plant leaf disease detection system using an android application. In Proceedings of the 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 20–22 April 2017; IEEE: New York, NY, USA, 2017.
14. Sethy, P.K.; Barpanda, N.K.; Rath, A.K.; Behera, S.K. Deep feature based rice leaf disease identification using support vector machine. *Comput. Electron. Agric.* **2020**, *175*, 105527. [CrossRef]
15. Zhao, S.; Peng, Y.; Liu, J.; Wu, S. Tomato Leaf Disease Diagnosis Based on Improved Convolution Neural Network by Attention Module. *Agriculture* **2021**, *11*, 651. [CrossRef]
16. Verma, S.; Chug, A.; Singh, A.P. Application of convolutional neural networks for evaluation of disease severity in tomato plant. *J. Discret. Math. Sci. Cryptogr.* **2020**, *23*, 273–282. [CrossRef]
17. Lintas, A.; Rovetta, S.; Verschure, P.F.; Villa, A.E. Artificial Neural Networks and Machine Learning–ICANN 2017. In *Proceedings of the 26th International Conference on Artificial Neural Networks, Alghero, Italy, 11–14 September 2017*; Proceedings, Part II; Springer: Cham, Switzerland, 2017; Volume 10614.
18. Barbedo, J.G.A. Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification. *Comput. Electron. Agric.* **2018**, *153*, 46–53. [CrossRef]
19. Kamal, K.C.; Yin, Z.; Wu, M.; Wu, Z. Depthwise separable convolution architectures for plant disease classification. *Comput. Electron. Agric.* **2019**, *165*, 104948.
20. Too, E.C.; Yujian, L.; Njuki, S.; Yingchun, L. A comparative study of fine-tuning deep learning models for plant disease identification. *Comput. Electron. Agric.* **2019**, *161*, 272–279. [CrossRef]
21. Nandhini, S.; Ashokkumar, K. Improved crossover based monarch butterfly optimization for tomato leaf disease classification using convolutional neural network. *Multimed. Tools Appl.* **2021**, *80*, 18583–18610. [CrossRef]
22. Wang, Q.; Qi, F.; Sun, M.; Qu, J.; Xue, J. Identification of tomato disease types and detection of infected areas based on deep convolutional neural networks and object detection techniques. *Comput. Intell. Neurosci.* **2019**, *2019*, 9142753. [CrossRef]
23. Mim, T.T.; Sheikh, M.H.; Shampa, R.A.; Reza, M.S.; Islam, M.S. Leaves Diseases Detection of Tomato Using Image Processing. In Proceedings of the 2019 8th International Conference System Modeling and Advancement in Research Trends (SMART), Moradabad, India, 22–23 November 2019; IEEE: Piscataway, NJ, USA, 2019.
24. Liu, J.; Wang, X. Early recognition of tomato gray leaf spot disease based on MobileNetv2-YOLOv3 model. *Plant. Methods* **2020**, *16*, 83. [CrossRef]
25. Kaluarachchi, T.; Reis, A.; Nanayakkara, S. A Review of Recent Deep Learning Approaches in Human-Centered Machine Learning. *Sensors* **2021**, *21*, 2514. [CrossRef]
26. Kaustubh, B. Tomato Leaf Disease Detection Tomato Leaf Disease Detection using CNN. Available online: <https://www.kaggle.com/kaustubhb999/tomatoleaf> (accessed on 21 September 2021).

27. Ben, A.G. How to use the Kaggle API from Colab. Available online: https://colab.research.google.com/github/corrieann/kaggle/blob/master/kaggle_api_in_colab.ipynb (accessed on 21 September 2021).
28. Gao, Z.; Shao, Y.; Xuan, G.; Wang, Y.; Liu, Y.; Han, X. Real-time hyperspectral imaging for the in-field estimation of strawberry ripeness with deep learning. *Artif. Intell. Agric.* **2020**, *4*, 31–38. [[CrossRef](#)]
29. Goodfellow, I.; Bengio, Y.; Courville, A. 6.2.2.3 *Softmax Units for Multinoulli Output Distributions*; Deep Learning. 2016; MIT Press: Cambridge, MA, USA, 2016; pp. 180–184. ISBN 978-0-26203561-3.
30. Kouretas, I.; Paliouras, V. Hardware implementation of a softmax-like function for deep learning. *Technologies* **2020**, *8*, 46. [[CrossRef](#)]
31. Warden, P.; Situnayake, D. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*, 2019; O'Reilly Media: Sebastopol, CA, USA, 2019; ISBN 978-1-492-05204-3.