Journal of
Innovation, Technology and Sustainability

Design of a classifier for tomato leaf disease identification

Hellen Wasike[1]

Stephen Njenga Thiru[1]

Geoffrey Mariga Wambugu[1]

[1]*Murang'a University of Technology, Kenya*

Abstract

Plants are the backbone of human existence for they are directly depended on for food. Plant infections and diseases are thus a major concern. Technology can promote food production in several ways through the application of computer vision technology that employs image processing to determine several aspects. Faster and timely plant disease recognition could immensely aid in the early application of appropriate treatment methods that fundamentally reduce economic losses. The introduction of machine learning techniques in image classification has revolutionized digital imaging and learning systems. Presently, convolutional neural networks have been found to provide the most accurate results while grey level co-occurrence is a popularly used descriptor. However, Convolution Neural Network (CNN) requires numerous learning iterations which lead to high computation costs whereas Grey Level Co-Occurrence Matrix (GLCM) cannot be used alone as a descriptor because a classifier is required to carry out the classification of the texture features extracted. This study proposed a hybrid model that combines CNN and GLCM techniques to classify plant diseases from a set of plant images. The research methodology used a systematic literature review and experimental research design. The systematic literature review was employed to determine and identify the existing techniques in digital plant images and the features to be used in classifying plant diseases. In experimentation, the study evaluated GLCM contrast, energy, and correlation features. The classification was carried out in three phases; 100, 150, and 200 iterations where the GLCM-CNN network had the best accuracy of 96.09% and F1 score of 0.8884 using energy texture images with 200 iterations.

Keywords: *CNN, Grey Level Co-occurrence Matrix, Machine Learning, Plant Features, Texture*

1. Introduction

Crop output and quality have grown significantly as a result of contemporary approaches and technology. To satisfy the demands of an expanding population, agricultural production has to increase at a faster rate. Despite technological advancements, agriculture is not immune to problems. The many sorts of diseases that afflict crops ever and then are one of the primary problems that they must almost certainly encounter. Bacteria, fungi, and other microorganisms cause many of these diseases. Over time, several measures to avoid plant diseases have been implemented. The use of pesticides on plants is one of the oldest and most primitive ways (Bharali, Bhuyan, and Boruah 2019). Conversely, excessive usage of these compounds has begun to have negative consequences, proving to be lethal to animals (including humans) and plants. The quest for alternatives to the utilization of excessive and unneeded chemicals in food has gotten a lot of attention in recent years all around the world.

Timely and faster recognition of these diseases could immensely aid in the early application of appropriate treatment methods that will fundamentally reduce economic losses. Plant stress recognition and classification have traditionally been done by researchers. The recognition and classification exercise has not only been time-consuming but also, to a great extent subjective (Ghosal et al. 2018; Kruse et al. 2014). Research findings (Fuentes et al. 2017; Voulodimos et al. 2018) have proposed machine learning (ML) techniques as the most efficient methods in image processing in several areas of study e.g. medicine, agriculture, language processing, and computer vision. The use of ML in agriculture has made it sustainable through the application of information technology in managing the farm by detecting, examining, and handling changes that may arise to maximize profits, easy maintenance, and protection of the farm resources. Improved and better decision-making on various aspects of the farm has been brought about (Banu 2015), for instance, by using farm inputs only when and where they are required e.g., fertilizers and pesticides.

The effectiveness of machine learning (ML) has been improving over time and this has made it conceivable to protect and maintain the world's food supply through smart farming. For all types of farms as well as different crop varieties, automated systems used to identify plant diseases in the leaves are efficient and precise. ML has made it feasible to manipulate and obtain hidden information from digital images. Deep neural networks have outperformed the conventional ML approaches in image classification.

Tomatoes are the most often consumed crop daily. They are found in almost every kitchen in the world. Conversely, their quality and quantity suffer as a result of different types of diseases. Tomato plants are particularly susceptible to a variety of diseases and as a result, detecting and classifying these diseases is critical. This method assists farmers in identifying a myriad of diseases that affect their crops. This research proposes a method of classifying

different types of tomato leaf diseases by combining Grey Level Co-Occurrence Matrix (GLCM) and Convolution Neural Network (CNN).

Grey Level Co-Occurrence Matrix and convolution neural networks have popularly been used in computer vision because of their great ability in extracting textural features and identifying different objects respectively. The texture features have extensively been utilized in image analysis since it has been demonstrated that it improves the effectiveness of the classification systems (Moya et al. 2019). However, GLCM cannot be used alone as a descriptor because a classifier is required to carry out the classification of the texture features extracted. CNN falls behind in classification accuracy (Adly and Moustafa 2016) as it requires a large number of learning repetitions for it to perform better (Santoni et al. 2015). Research findings have concluded that a modified CNN can produce improved classification accuracy than when used alone (Santoni et al. 2015) with fewer iterations.

The paper unfolds as follows: section two presents relatable work in this domain, section three describes the methods proposed in this study, section four presents the findings, and Section Five concludes.

2. Related work

CNN has gained considerable attention in recent years. It has extensively been applied in the image processing domain because of its effectiveness and efficiency in classification tasks. Its performance has outshone the traditional ML approaches.

A CNN model with three convolution and max-pooling layers, one flattening layer, and one fully connected layer was used to detect leaf diseases (Agarwal et al. 2020). The convolution and max-pooling layers had $3x3$ and $2x2$ filters respectively. A total of 50,000 leaf images from PlantVillage with 14 different crops were employed. The model achieved an accuracy performance of 91.2%.

Ahmad et al. (2020) compared the performance of four pre-trained CNN models; VGG-16, VGG-19, Resnet, and Inception V3. The performance of the models was evaluated using two divergent datasets. The first dataset was from a controlled environment while the second one was from the field. 2364 images with four types of tomato diseases were utilized. VGG-16 had 16 layers while VGG-19 had 19 layers. Resnet employed residual learning while Inception V3 data preprocessing was carried out using histogram equalization. The classification was conducted in two ways, using feature extraction and parameter tuning. The 10-fold cross-validation was used to validate the models. Inception V3 outperformed the other three on both datasets.

Salih et al. (2020) used 6202 images from PlantVillage that had six categories from which five had diseased and one had healthy leaf images. CNN had three blocks and the first block had convolution, batch normalization, activation function, and max-pooling layers. The second block had convolution and max-pooling layers and the third block had fully connected, SoftMax, and classification layers. The CNN model achieved an accuracy of 9.34.

CNN Xception model was modified and trained using different optimization algorithms and their performances were evaluated (Thangaraj, Vishnu, and Kaliappan 2020). Adaptive moment estimation (Adam), Stochastic Gradient Descent (SGD), and Root Mean Square Propagation (RMSProp) algorithms were utilized. The modified xception model had feature extraction and classification components. The feature extraction component had convolution layers, pooling layers, and rectified linear units while the classification component had a global average pooling layer and SoftMax classifier. Adam had a better performance of 99.55% with the lowest loss value outperforming SGD and RMSProp which had 99.01% and 81.77% respectively.

Pre-trained AlexNet and VGG16 models were modified to classify seven classes of tomato leaves images that had six diseased and one healthy class (Rangarajan, Purushothaman, and Ramesh 2018). A total of 13262 images from PlantVillage were segmented and augmented and used for training. Three layers of each model were modified and transfer learning was utilized during the training. AlexNet and VGG16 achieved an accuracy level of 97.49% and 97.23% respectively. Zhao et al. (2021) utilized deep CNN with 101 layers. The model integrated attention mechanism which included residual blocks and attention extraction modules. The study used 4585 images acquired from PlnatVillage with ten categories. The accuracy performance of the model was 96.81%.

The existing literature offers valuable perspectives on the various techniques. CNNs have yielded promising results, but only after the input data has been preprocessed. Image data preprocessing takes time and might result in the loss of essential data. In this article, CNN is utilized to classify ten classes of tomato leaf images in which nine classes have diseased and one class has healthy images. The study combines GLCM and CNN approaches in learning and classifying the images in a dataset that has not been preprocessed to examine the performance of CNN with a reduced number of layers and fewer iterations.

3. Methodology

3.1 Data collection

The data used in this study was collected from an online image dataset from PlantVillage. The dataset had already been divided into training and testing samples. The samples had nine types of tomato leaf diseases with each disease having 1000 and 100 images for training

and validation respectively. The diseases were bacterial spot, early blight, late blight, leaf mold, Septoria leaf spot, spider mite, target spot, tomato mosaic virus, and tomato yellow leaf curl virus. It also contained a set of healthy tomato leaves with 1000 and 100 images in training and validation correspondingly. The dataset had a total of 11,000 images.

3.2 Data pre-processing

There was no pre-processing performed on the image dataset gathered before training the model. The built proposed model was fed with raw images as downloaded from PlantVillage.

3.3 GLCM

The Grey Level Co-occurrence Matrices (GLCMs) were first proposed by Haralick in 1973 (Haralick, Dinstein, and Shanmugam 1973) as a texture descriptor. It continues to be widely used in texture classification which determines the relationship between pairs of pixels in an image through computation. It is a presentation of the frequency of occurrences of several combinations of pixel-grey intensities in an image. It gives the information regarding the texture of the image.

It contains columns and rows representing a series of potential values of the image. GLCM considers the spatial relationship $(\Delta x, \Delta y)$ of two pixels with grey levels at a given distance d and orientation angle θ. GLCM comprises information regarding pixel points with the same grey intensity values (Ding 2017). The textural features are obtained from the computation of the grey level co-occurrence matrices generated. Technically, GLCM uses a distinctive matrix function to establish a joint occurrence matrix of the image data (Hall-Beyer 2017) The textural features intuitively present quality features on pixel distribution in the texture like regularity, coarseness, and smoothness (Cavalin and Oliveira 2018). Over the years, rather than using GLCM independently, it has been combined with other techniques.

The first step was to convert the images into greyscale. A greyscale is an image whose every pixel is a representation of the intensity information of the amount of light it carries. The grey scaling process involves the conversion of the tone of the original image to that which a computer can use. This step is necessary since the original image values of pixels have to be scaled to an appropriate range of values. It plays the role of filtering noise which decreases the GLCM sparsity. Texture analysis using GLCM was performed on the images converted into greyscale to acquire contrast, energy, and correlation texture parameters.

A GLCM was generated by putting together the incidences of a grey value of two pixels (i, j) separated by the offset $(\Delta x, \Delta y)$ of the greyscale image. The GLCMs were generated from four different offset angles, 0^0, 45^0, $90^0, and\ 135^0$. The offsets represent the connection between the central pixel and the immediate pixels in the four directions. The results of the

GLCM elements obtained from the offsets were then summed up to get one direction-invariant GLCM. The GLCM constructed from the summation is un-normalized, which calculates the frequency of occurrence of grey values of each pair of pixels as neighbours.

Uniform quantization was used to determine the dimension of the GLCM image by giving the maximum number of grey tones. A uniform quantization technique was applied because it is the easiest to implement Quantization helps reduce noise-induced effects to some degree by merging grey intensities that are similar within the image. The level of quantization of the GLCM was set to 256 since the intensity of the central pixel and that of its neighbour pixel ranged between 0 and 255 and hence, the dimensions of the generated GLCM were set to 256 x 256.

Finally, each GLCM constructed was normalized to ensure that they have similar dimensions. Hu and Zheng (2019) note there is no given theoretical evidence that yields optimum classification through the use of certain GLCM features. Consequently, it is upon the researcher to establish the features that suit their studies. So, this study employed contrast, energy, and correlation features after GLCM normalization. The generated GLCM images were then fed into the CNN network as the input. Once fed into the model, the CNN reduced the image dimensions to a size of 128x128.

1. **Contrast** gives the number of local variations between the reference pixel and its neighbour within the entire image. It uses the GLCM weighted average values. The frequency of occurrence of a pair of pixels constructs the weight. An increase in weight associated with commonness yields a textured aspect that rises concerning the order and vice versa.

$$Contrast = \sum_{i,j=0}^{n_g-1} P_{i,j}(i-j)^2 \qquad (Eq.\ 1)$$

   Where i and j represent the spatial coordinates of the GLCM matrix function $P(i,j)$ whereas n_g is the grey tone.

2. **Angular second moment (ASM) or energy** utilize every P_{ij} as a weight value. Homogeneity brings about high values in energy. Energy is also known as uniformity and it measures the intensity of grey tones in the GLCM and textural uniformity. Energy gives the aggregate of the squared parts in GLCM.

$$ASM = \sum_{i,j=0}^{n_g-1} {P_{i,j}}^2 \qquad (Eq.\ 2)$$

   Where i and j represent the spatial coordinates of the GLCM matrix function $P(i,j)$ whereas n_g is the number of defined grey tones in the quantized image

3. **Correlation** attribute provides the amount of grey level linear dependences of neighbouring pixels in the image. correlation among pixels implies that there is a linear association between two adjacent pixels. A big correlation in the texture implies a high pixel relationship predictability.

$$Correlation = \frac{\sum_{i,j=0}^{n_g-1}(i,j)P(i,j)-\mu_i\mu_j}{\sigma_i\sigma_j} \quad \textit{(Eq. 3)}$$

Where i and j represent the spatial coordinates of the GLCM matrix function $P(i,j)$; μ is the GLCM mean whereas n_g is the number of defined grey tones in the quantized image

3.4 CNN Structure

Convolution neural networks are a class of multilayer neural systems, which are trained by the use of backpropagation. CNNs are made of three fundamental components; convolution layers, pooling layer, and rectified linear units (ReLUs) which serve as the activation functions. A CNN can have several convolution and pooling layers. Every convolution layer is succeeded by a max-pooling layer, after which fully connected layers follow. Most of the computation in CNN happens in the convolution layer and it is the first layer that the image goes through after input. Inside a convolutional layer, some filters inspect a segment of an image. (Beysolow II 2017). The product of the output generated is a feature map and will be used in the next layer. During the forward pass, every filter is convolved using input that generates a map. When all these maps are stacked, they produce the output of this layer.

The pooling layer is placed in between the successive convolutional layers. This layer "pools" the feature maps generated in convolution layers into an image. The pooling layer reduces the complexity of the model thereby enhancing spatial representation by effectively carrying out dimensionality reduction through subsampling (Beysolow II 2017; Voulodimos et al. 2018). The layer does not impact the dimension of the depth of the volume. The down sampling (subsampling) results in simultaneous information loss. In any case, such a loss is valuable for the system because the size reduction results in reduced computational overhead for the succeeding layers of the system. Furthermore, it prevents overfitting. Max pooling and average pooling are the most ordinarily utilized functions. Non-linearity layer consists of nodes that employ several activation functions which present nonlinearities that make multi-layered systems desirable. Rectified Linear Units (ReLU) are the most preferable functions since they enhance faster training of the neural networks. The fully connected layer comes after the convolutional, pooling, and nonlinearity layers which are user-determined. There is perhaps at least one fully connected layer that implements high-level reasoning by taking each activation of the preceding layer and linking them to every neuron in the current layer (Bhandare et al. 2016). The images that enter this layer are significantly smaller than the initial inputs because of the image reductions stated in the previous operations. The

reduced images are then scanned and should match every feature map and transform every one of the qualities given into a list of qualities. The fully connected layer finally transforms the two-dimensional feature maps into a one-dimensional feature vector. The resulting vector can either be served into a specific number of classifications or be considered for further processing.

The proposed CNN model structure is comprised of three convolutional layers, two max-pooling layers, and two fully connected (dense) layers. The first convolutional layer had 32 filters the second layer had 16 filters and the third layer had 8 filters. The size of each filter in all three convolutional layers was $3\ x\ 3$. Each of these layers used rectified linear unit (ReLU) activation function. Each convolutional layer was succeeded by a max-pooling layer. A $2\ x\ 2$ max-pooling kernel size pooled the feature maps generated by the convolution layer and then fed them as input to the succeeding convolution layer. The convolution, activation, and max-pooling process was repeated for all three convolution layers as shown in Table 1. The kernels in the convolution layer and max-pooling moved with a $1\ x\ 1$ stride.

After the third convolutional layer, the network was flattened to a one-dimensional feature vector. Flattening was performed on the output from the convolutional layers to generate a distinct long feature vector. The flattening layer was then followed by two dense layers where one dense layer served as the final classification layer with a SoftMax activation function used for classification since the class model was categorical. SoftMax can be suitably applied in multiclass classification problems that require more than two labelled classes. SoftMax gives probability distribution to every class and then sums up these probabilities to 1.0. In doing so, this helps to enhance the training convergence faster than it normally would.

To prevent the network from overfitting, the dropout technique was employed in the dense layer. The network set the drop probability to 0.5. The batch size was set to 64 with 50, 100, and 200 epochs with ten steps per epoch. This meant that 64 samples were passed through the network 50, 100, and 200 times. The adaptive moment estimation (Adam) method was utilized as the optimizer for the network with a learning rate of 0.001. It is a commonly used optimization algorithm as its implementation is straightforward. Computationally, the Adam optimizer is very effective and can effectively deal with large sets of data and parameters.

Table 1: Model architecture

Layer (Type)	**Output Shape**	**Param #**	**Filter size**	**Activation Function**
Convolution (Conv2D) 1	None, 126, 126, 32	896	3 x 3	ReLU
Max_Pooling (MaxPooling 2D) 1	None, 63, 63, 32	0	2 x 2	-

Convolution (Conv2D) 2	None, 61, 61, 16	4624	3 x 3	ReLu
Max_pooling (MaxPooling 2D) 2	None, 30, 30, 16	0	2 x 2	-
Convolution (Conv2D) 3	None, 28, 28, 8	1160	3 x 3	ReLu
Max_Pooling (MaxPooling 2D) 3	None, 14, 14, 8	0	2 x 2	-
Flatten	None, 1568	0	-	-
Dense 1	None, 128	200832	-	ReLu
Dropout	None, 128	0	-	-
Dense 2	None, 10	1290	-	SoftMax

3.5 Training Environment

The training and validation were carried out on Nvidia GeForce GTX 1080 running on 64-bit Windows 10 operating system. The model was developed using Python 3.7 in Anaconda 4 with Spyder Notebook as the main Python Integrated Development Environment (IDE). The CNN network was built on Keras on a TensorFlow graphics processing unit (GPU) backend. Keras running on TensorFlow backend enables the enhancement of fast experimentation. Other important libraries used included NumPy, skimage, sklearn, greycomatrix, and greycoprops for the implementation of GLCM. Most of the parameters like kernel size, batch size, learning rate, number of epochs, and stride dimension were all established through a trial-and-error approach.

Table 2: Hardware/Software Characteristics

Hardware/Software	**Specifications**
Operating System	Windows 10 (64-bit)
Environment	Python, Keras (TensorFlow)
Graphics (GPU)	Nvidia GeForce GTX 1080
Processor (CPU)	Intel Core i7, 3.6GHz
Graphics RAM Type	GDDR5X
RAM Memory	32 GB
Hard Disk Memory	2TB 7200RPM + 512GB SSD

4. Results and Discussion

The experimental validation was conducted in two modes; using CNN and GLCM-CNN. These are two networks whose performance was compared. The first mode of classification used the original CNN network while the second applied the GLCM-CNN network. The aim of performing classification with the two approaches was to establish and evaluate the performance of the proposed model and then compare it with the CNN network using similar datasets. All instances of the image dataset were used for both modes of classification.

In the first mode, the CNN network was fed with the raw downloaded RGB image dataset from PlantVillage. The second mode involved passing the images through GLCM first. Consequently, the GLCM-generated images served as input data into the CNN, forming the GLCM-CNN network. The GLCM images' texture features were then extracted from the raw RGB images. For both modes, the classification process was performed three times. This depended on the number of iterations per classification. Round one had 50 iterations, round two had 100 iterations and round three had 200 iterations. For all classifications, the networks used categorical classification modes.

The CNN network had an overall best accuracy performance of 94.66% for 200 iterations while GLCM-CNN had 96.09% for 200 iterations using energy texture images. Table 3 shows the performance of the two networks using different numbers of iterations with accuracy percentage as a unit of measure. GLCM-CNN contrast had an accuracy of 95.89% for 200 iterations which was higher compared to CNN. The accuracy levels for both classifiers were compared. In both cases, GLCM-CNN generally performed better in all iterations compared to CNN. However, correlation had the lowest performance with 91.88% for 200 iterations.

In all the classification rounds, the training and validation losses started at the peak and decreased as the number of iterations increased. They however fluctuated in the first half of iterations before dropping consistently until the end. Conversely, the training and validation accuracies increased correspondingly in the classification for both networks. Both networks attained the highest validation accuracy and lowest validation loss during the classification with 200 iterations.

Table 3: Classification accuracy for CNN and GLCM-CNN

Number of iterations	CNN	GLCM-CNN		
		Contrast	Energy	Correlation
50	78.73	90.49	91.08	84.60
100	86.86	93.24	94.58	87.12
200	94.66	95.89	96.97	91.88

The F1 score for GLCM-CNN and CNN was fairly consistent in all rounds of classification since the variations were very small. On average CNN had an F1 score of 0.8770 while GLCM-CNN had an average of 0.8763. Table 4 shows the F1 score performance of the CNN and GLCM-CNN networks. CNN attained the highest F1 score of 0.8801 during the classification with 100 iterations while GLCM-CNN achieved the highest F1 score of 0.8884 with energy in the classification with 200 iterations. The classification with 100 iterations and correlation recorded the lowest score of 0.8649. In all three classification rounds, contrast had an average score of 0.8754, while energy and correlation had 0.8801 and 0.8735 respectively.

Table 4: F1 score values for CNN and GLCM-CNN networks

Number of iterations	**CNN**	**GLCM-CNN**		
		Contrast	**Energy**	**Correlation**
50	0.8732	0.8671	0.8729	0.8742
100	0.8801	0.8803	0.8790	0.8649
200	0.8776	0.8787	0.8884	0.8815

The primary objective of the research was to develop a model that can effectively recognize and classify ten classes of the tomato plant which included nine diseased and one healthy leaf image. The experiment was conducted using the PlantVillage image dataset that had already been split into training and validation sets. Based on an accuracy performance comparison with the original CNN, GLCM-CNN had a better performance of 96.09% using energy images with 200 iterations. CNN networks are known to perform better with an increasing number of iterations. As can be seen from the results, the performance of CNN improved with an increased number of iterations. With 50 iterations, CNN had its poorest performance of 78% accuracy while the proposed model outperformed it with an accuracy of 90.49%, 91.08%, and 84.60% for contrast, energy, and correlation images respectively. The GLCM-CNN model had a good performance with only 50 iterations and it successfully achieved its goal of effectively identifying and classifying plant images with few iterations.

The image dataset used was highly skewed about training and validation sets they had 1000 and 100 images respectively. The unevenness in the class image distribution discourages on only focusing on accuracy as a metric since accuracy works best with a balanced class. This implies that relying on accuracy alone may not give the correct performance of the model because accuracy is maximized when false negatives and false positives have the cost. Skewness in class data can make a classifier achieve a low misclassification rate by simply selecting the majority training sample. As a result, the F1 score which is viewed as a better metric when working with imbalanced data was used to assess the performance of the model. CNN had an F1 score of 0.8763 while GLCM-CNN had an average score of 0.8770. The

F1 score for the two models is almost similar which was a good performance since a score of 0 is the poorest while 1 is a perfect score. Comparing these scores with the accuracy achieved, the proposed model performed well implying that the number of false negatives and false positives during the classifications was low.

5. Conclusion and Future Work

The conclusion drawn from the results encourages using GLCM-CNN in classifying tomato plant leaf diseases and it consequently fulfills the research objectives and answers the research questions. The use of raw images which were not pre-processed made the model more efficient enabling it to work better in the recognition and classification of new data. The accuracy and F1 score values indicate that the model performed well given that the images were used raw.

Finally, number of epochs is the most significant factor that determines the accuracy and efficiency of the model. The number of epochs refers to how many iterations the model takes to learn the entire data set. Research studies have not specified the number of epochs a model ought to be trained with, although it is apparent that the proposed model learned more as it was trained more and more with different iterations.

To get a more accurate performance of the findings of this study, the model can be applied to different types of crops in a large and complex real-world setting. Furthermore, no pre-processing was required to refine the input of the approach utilized in this research. Therefore, in the future, if adequate processing speed is available, a huge number of datasets can be added to do analysis. The process of recognizing plant diseases can further be enhanced to include the analysis of several parts of the plant and different types of leaves. The neural network system may be improved more to consider dataset scalability and additional environmental diversities, reducing the size and complexity of the deep model for small machines. The suggested GLCM-CNN technique may also be evaluated with other parameters such as three-dimensional GLCM.

References

Adly, M. Hussein, and Mohamed Moustafa. 2016. “A Hybrid Deep Learning Approach for Texture Analysis.” The American University in Cairo.

Agarwal, Mohit, Abhishek Singh, Siddhartha Arjaria, Amit Sinha, and Suneet Gupta. 2020. “ToLeD: Tomato Leaf Disease Detection Using Convolution Neural Network.” *Procedia Computer Science* 167(2019):293–301. doi: 10.1016/j.procs.2020.03.225.

Ahmad, Iftikhar, Muhammad Hamid, Suhail Yousaf, Syed Tanveer Shah, and Muhammad Ovais Ahmad. 2020. “Optimizing Pretrained Convolutional Neural Networks for Tomato

Leaf Disease Detection." *Complexity* 2020. doi: 10.1155/2020/8812019.

Banu, S. 2015. "Precision Agriculture: Tomorrow's Technology for Today's Farmer." *Journal of Food Processing & Technology* 06(08):8–13. doi: 10.4172/2157-7110.1000468.

Beysolow II, Taweh. 2017. *Introduction to Deep Learning Using R - A Step-by-Step Guide to Learning and Implementing Deep Learning Models Using R*. New York: Apress.

Bhandare, Ashwin, Maithili Bhide, Pranav Gokhale, and Rohan Chandavarkar. 2016. "Applications of Convolutional Neural Networks." *International Journal of Computer Science and Information Technologies* 7(5):2206–15.

Bharali, Parismita, Chandrika Bhuyan, and Abhijit Boruah. 2019. "Plant Disease Detection by Leaf Image Classification Using Convolutional Neural Network." Pp. 194–205 in *4th International Conference, Communications in Computer and Information Science (ICICCT 2019)*. Vol. 1025 CCIS, edited by A. Bin Gani, P. K. Das, L. Kharb, and D. Chahal. New Delhi,: Springer Singapore.

Cavalin, Paulo, and Luiz S. Oliveira. 2018. "A Review of Texture Classification Methods and Databases." *Proceedings - 2017 30th SIBGRAPI Conference on Graphics, Patterns and Images Tutorials SIBGRAPI-T 2017* 2018-Janua:1–8. doi: 10.1109/SIBGRAPI-T.2017.10.

Ding, Xuejing. 2017. "Texture Feature Extraction Research Based on GLCM-CLBP Algorithm." Pp. 167–71 in *7th International Conference on Education, Management, Information and Mechanical Engineering (EMIM)*. Vol. 76.

Fuentes, Alvaro, Sook Yoon, Sang Cheol Kim, and Dong Sun Park. 2017. "A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition." *Sensors (Switzerland)* 17(9). doi: 10.3390/s17092022.

Ghosal, Sambuddha, David Blystone, Asheesh K. Singh, Baskar Ganapathysubramanian, Arti Singh, and Soumik Sarkar. 2018. "An Explainable Deep Machine Vision Framework for Plant Stress Phenotyping." *Proceedings of the National Academy of Sciences of the United States of America* 115(18):4613–18. doi: 10.1073/pnas.1716999115.

Hall-Beyer, Mryka. 2017. "GLCM TEXTURE: A TUTORIAL." *University of Calgary* 3.0(March):1–75.

Haralick, Robert M., Its'hak Dinstein, and K. Shanmugam. 1973. "Textural Features for Image Classification." *IEEE Transactions on Systems, Man and Cybernetics* SMC-3(6):610–21. doi: 10.1109/TSMC.1973.4309314.

Hu, Yifan, and Yefeng Zheng. 2019. "A GLCM Embedded CNN Strategy for Computer-Aided

Diagnosis in Intracerebral Hemorrhage." *ArXiv* 1–9.

Kruse, Ole Mathis Opstad, José Manuel Prats-Montalbán, Ulf Geir Indahl, Knut Kvaal, Alberto Ferrer, and Cecilia Marie Futsaether. 2014. "Pixel Classification Methods for Identifying and Quantifying Leaf Surface Injury from Digital Images." *Computers and Electronics in Agriculture* 108:155–65. doi: 10.1016/j.compag.2014.07.010.

Moya, Luis, Homa Zakeri, Fumio Yamazaki, Wen Liu, Erick Mas, and Shunichi Koshimura. 2019. "3D Gray Level Co-Occurrence Matrix and Its Application to Identifying Collapsed Buildings." *ISPRS Journal of Photogrammetry and Remote Sensing* 149(May 2018):14–28. doi: 10.1016/j.isprsjprs.2019.01.008.

Rangarajan, Aravind Krishnaswamy, Raja Purushothaman, and Aniirudh Ramesh. 2018. "Tomato Crop Disease Classification Using Pre-Trained Deep Learning Algorithm." *Procedia Computer Science* 133:1040–47. doi: 10.1016/j.procs.2018.07.070.

Salih, Thair A., Ahmed J. Ali, and Mohammed N. Ahmed. 2020. "Deep Learning Convolution Neural Network to Detect and Classify Tomato Plant Leaf Diseases." *OALib* 07(05):1–12. doi: 10.4236/oalib.1106296.

Santoni, Mayanda Mega, Dana Indra Sensuse, Aniati Murni Arymurthy, and Mohamad Ivan Fanany. 2015. "Cattle Race Classification Using Gray Level Co-Occurrence Matrix Convolutional Neural Networks." *Procedia Computer Science* 59(Iccsci):493–502. doi: 10.1016/j.procs.2015.07.525.

Thangaraj, Rajasekaran, S. Anandamurugan Vishnu, and Kumar Kaliappan. 2020. "Automated Tomato Leaf Disease Classification Using Transfer Learning - Based Deep Convolution Neural Network." *Journal of Plant Diseases and Protection* (0123456789). doi: 10.1007/s41348-020-00403-0.

Voulodimos, Athanasios, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. 2018. "Deep Learning for Computer Vision: A Brief Review." *Computational Intelligence and Neuroscience* 2018. doi: 10.1155/2018/7068349.

Zhao, Shengyi, Yun Peng, Jizhan Liu, and Shuo Wu. 2021. "Tomato Leaf Disease Diagnosis Based on Improved Convolution Neural Network by Attention Module." *Agriculture (Switzerland)* 11(651). doi: 10.3390/agriculture11070651.