Intelligent Methods in Engineering Sciences 3(1): 022-036, 2024

INTELLIGENT METHODS
IN ENGINEERING SCIENCES

https://www.imiens.org

International Open Access

March, 2024

e-ISSN 2979-9236

Research Article

https://doi.org/10.58190/imiens.2024.84

Classification of Diseases in Tomato Leaves Using Deep Learning Methods

Muslume Beyza YILDIZ [a,*], *Mustafa Fatih HAFIF* [a], *Emre Kagan KOKSOY* [a], *Ramazan KURSUN* [b]

[a] *Department of Computer Engineering, Faculty of Technology, Selcuk University, Konya, TÜRKİYE*
[b] *Guneysinir Vocational School, Selcuk University, Guneysinir, TÜRKİYE*

ARTICLE INFO

Article history:
Received 20 January 2024
Accepted 28 March 2024

Keywords:
convolutional neural networks,
image processing,
machine learning algorithms,
mobile application integration,
tomato leaf diseases

ABSTRACT

The automatic detection of diseases in tomato leaves significantly contributes to tomato production and enables farmers to manage these issues more effectively. Tomatoes are a crucial commercial crop for local markets and exports, representing a significant agricultural sector in our country. Diseases affecting tomato leaves directly influence tomato yield and quality, making early detection and intervention paramount. Our study aims to address tomato losses due to leaf diseases using computer technology. Recently, Convolutional Neural Networks (CNN) have been employed in various fields including agriculture, military, robotics, and medicine for classification, object detection, and segmentation tasks. The integration of computer vision and image processing with deep learning architectures has led to notable advancements in these areas, offering solutions with higher accuracy and reducing human error. In our research, a dataset was created using images of tomato leaf diseases selected from Kaggle. Algorithms such as k-Nearest Neighbors (KNN), Random Forest (RF), Logistic Regression (LR), Support Vector Machine (SVM), and Neural Networks (NN) were applied using Orange, a data visualization and analysis software. Moreover, a custom algorithm developed in Python demonstrated the highest accuracy. while the highest classification accuracy of classification made with machine learning algorithms was 95.6%, the classification accuracy was achieved about 96% with the developed deep learning model. This system was integrated into an Amazon Web Services (AWS) Lambda function, subsequently utilized in a mobile application developed using Flutter for the UI and Dart for backend, ensuring connectivity with the Lambda function.

1. INTRODUCTION

Agriculture is an indispensable element in meeting the global food needs of humanity, playing a pivotal role in the economic development of nations [1]. It also holds a critical role in contributing to the development of countries economically [2]. The substantial increase in population densities has escalated the need for food security [3]. Nonetheless, to meet this demand, the agricultural sector is in continuous need of innovative solutions and effective methods.

Due to their high nutritional value, tomatoes play a significant role in agricultural trade and production [4]. They are of great importance among vegetable crops worldwide, with their production increasing visibly each year. With the rising demand, farmers face losses due to diseases [2].

It is anticipated that approximately 40% of crops are lost due to plant diseases [5]. Similarly, leaf diseases in tomato cultivation cause significant losses in production. The early diagnosis of these diseases is of critical importance for sustaining efficiency and sustainability in the agricultural economy. Previous research indicates that 80-90% of plant diseases manifest on leaves [6, 7], thus, diagnoses are commonly made by examining leaf symptoms. However, the accuracy and precision of these diagnoses require extensive knowledge and experience, in addition to incurring greater time and costs [8].

Incorrect use of agricultural pesticides for disease treatment can increase the resistance of pathogens affecting plants, making the accurate diagnosis of plant diseases extremely important. In today’s world, where mobile technology is prevalent in many aspects of our lives, this article presents the detection of diseases in tomato leaves via a mobile application, enabling producers themselves to control the quality and efficiency of tomato production. Pre-trained machine learning models and deep learning algorithms are stored on the server. Through our application, producers can photograph diseased leaves with their mobile phones and upload these images to the server. Then, the server performs analyses to diagnose the

*** Corresponding Author:** rkursun@selcuk.edu.tr

disease. The results are provided back to the user as a decision support mechanism for determining tomato leaf diseases. In short, the application performs disease prediction through an application interface using a pre-trained model via AWS Lambda. Figure 1 illustrates the operational logic of the application.

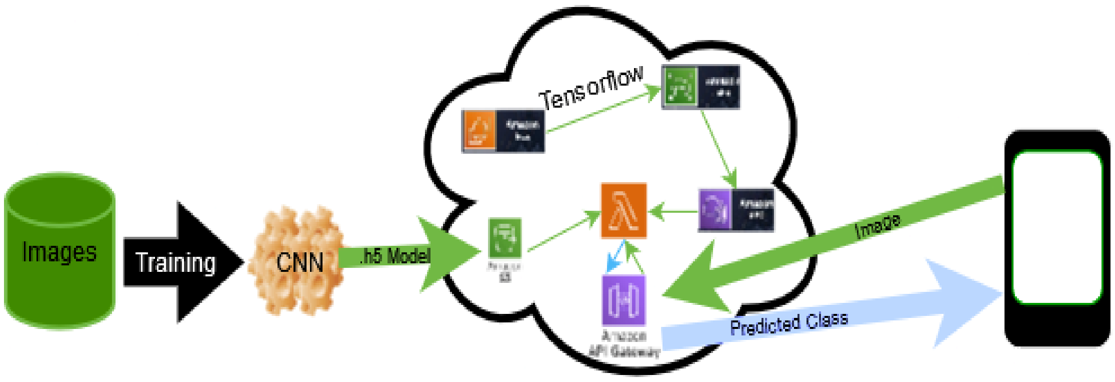


Figure 1. Block diagram of the study

2. Related Work

During the literature review, articles focusing on the detection of diseases in tomato leaves were examined. Additionally, prior studies directly related to our research scope and those indirectly contributing to this topic were included.

Agarwal et al., utilized the Plant Village dataset for the detection of diseases on tomato leaves in their study, employing a Convolutional Neural Network (CNN)-based model for disease detection and classification. This model comprises 3 convolutional, 3 max-pooling, and 2 fully connected layers, achieving an average accuracy rate of 91.2% across 9 diseases and one healthy class [9].

Ashok et al., developed a method using deep learning techniques for tomato leaf disease detection, preferably using plant images. They proposed a CNN algorithm for hierarchical feature extraction, reaching a 98% accuracy level when tested on OpenCV [10].

Kaushik et al., developed a deep CNN model using PyTorch to classify 6 tomato diseases. They validated their test dataset with parameters obtained from the ResNet-50 model, enhancing the model's accuracy to 97% through data augmentation [11].

Jiang et al., used 1000 photographs for three different tomato leaf diseases. They replaced ResNet-50's activation function with Leaky-ReLU and altered the kernel size of the first convolutional layer to 11x11, achieving a 98% accuracy rate [12].

Das et al., created a dataset for a tomato leaf disease detection system comprising 7 diseased and 1 healthy tomato leaf image. The dataset includes 60 features derived from each image. Among the experiments conducted with SVM, RF, and LR algorithms, the highest accuracy rate of 87.6% was obtained by SVM [13].

Wang and Liu focused on the development of the YOLOv3 model for detecting complex natural environment tomato anomalies in their study. The developed YOLO-Dense model achieved the highest accuracy of 96.41% compared to SSD, Faster R-CNN, and YOLO V3 models [14].

Kibriya et al., employed two CNN-based models for diagnosing tomato leaf diseases, using the Plant Village dataset containing 10735 leaf images. VGG16 achieved 98% accuracy, while GoogLeNet reached 99.23% accuracy [15].

Trivedi et al., used CNNs to classify tomato diseases, utilizing a dataset of 3000 images comprising 9 types of diseased tomato leaves and one healthy leaf on Google Colab. After processing the CNN model with varying hyperparameters, they achieved an accuracy rate of 98.49% [7].

Kursun et al., demonstrates that using the U-Net architecture for segmenting agricultural images, followed by classification with deep learning models like DenseNet201, significantly improves the accuracy of detecting plant diseases in bean plants, achieving up to 100% accuracy [16].

Anandhakrishnan and Jaisakthi used a total of 18,160 images of tomato leaf diseases from the Plant Village dataset, allocating 60% for training and 40% for testing. The DCNN (Deep Convolutional Neural Network) model achieved an accuracy rate of 98.4% in the test set [17].

Rahman et al., collected their dataset from different tomato fields in Pakistan for their study on tomato leaf disease detection. Using the Gray Level Co-occurrence Matrix (GLCM) GLCM algorithm to calculate 13 different features, they classified tomato diseases with the Support Vector Machine (SVM), showing that the method works with 100% accuracy for healthy leaves, 95% for early blight, 90% for Septoria leaf spot, and 85% for late blight [18].

Janarthan et al., used a total of 18160 images for the detection of 10 different diseases in their project, employing Adam Stochastic optimization with Python's

OpenCV library [19].

R. Sujatha et al., prepared 609 photographs for training with 10-fold cross-validation, focusing on a performance comparison between ML and DL algorithms. They considered SVM, RF, and SGD as machine learning algorithms, and Inception-V3, VGG-16, and VGG-19 as deep learning algorithms, with the highest accuracy rate of 89.5% achieved by VGG-16 [20].

M. Sardogan et al., used a total of 500 photographs, 400 for training and 100 for testing, in their research. They employed ReLU as the activation function and the LVQ algorithm for classification and supervised learning, achieving an average accuracy rate of 86% [21].

Harakannanavar et al., used contour tracking for boundary extraction of leaf samples. Furthermore, Discrete Wavelet Transform (DWT), Principal Component Analysis (PCA), and GLCM were used for feature extraction, and SVM, KNN, CNN were used for classification, with CNN achieving the highest accuracy rate of 99.6% [22].

De Luna et al., worked on three different diseases in a tomato species named Diamete Max: Phoma Rot, Leaf Miner, and Target Spot, using approximately 5000 images in total. Classification with DCNN and anomaly detection with F-RCNN were conducted, finding the anomaly test's confidence score to be 80%. The classification and disease recognition accuracy rate was found to be 95.75%, and diseases in tomato leaves were captured with 91.67% accuracy using an automatic image capture system [23].

The literature review indicates that some studies have provided high accuracy in performance. However, the portability of these systems has been found lacking. The mission of this project is to develop smaller and faster versions of existing disease diagnosis algorithms and make these versions usable on small-scale devices like mobile phones.

3. Materials and Methods

Agriculture plays a critical role in the survival of humanity on Earth, serving as both a primary source of nourishment and a cornerstone of economic growth for regions. However, it is well-documented that plants are susceptible to adverse effects from various factors, including bacteria, viruses, fungi, or the excessive use of agricultural chemicals. The misdiagnosis of diseases affecting agricultural products often leads to the unnecessary and excessive application of pesticides. This practice contributes to the development of resistant pathogen strains, incurs higher costs, and exacerbates the frequency of outbreaks, resulting in significant financial and environmental losses [24, 25].

In response to these challenges, our study has developed a dataset using photographs of tomato leaf diseases selected from Kaggle. Through the application of KNN, RF, LR, SVM, and NN algorithms via Orange, an open-source data visualization, and analysis tool, we have explored various computational approaches. Additionally, a custom algorithm was crafted using the Python programming language and applied to the same dataset to enhance diagnostic accuracy. Subsequently, this algorithm was integrated into an Amazon Web Services (AWS) Lambda function, which served as a backend for a newly developed mobile application. The mobile application, designed to facilitate the detection of tomato leaf diseases, connects seamlessly with the AWS Lambda function, embodying our commitment to leveraging advanced technologies for sustainable agricultural practices.

3.1. Raw Dataset

Kaggle is a community of data scientists and machine learning practitioners. It actively organizes competitions with prizes for various projects. During the initial development stages, a small-scale dataset from Kaggle was chosen. A dataset composed of tomato leaves found on Kaggle (Kaustubh B., 2020) was utilized. This dataset contains 10 classes, out of which only 5 were subjected to testing. As the testing process continued, datasets obtained by different Kaggle users (Ashish Motwani et al., 2023; Nouaman Lamhari et al., 2019) were merged with the initial dataset. This integration allowed for an examination of the model's performance on dense data sets.

The preferred 4 different tomato plant leaf diseases are given together with their descriptions.

Bacterial Spot: Bacterial spot is a disease caused by the bacterium Xanthomonas campestris pv. vesicatoria. It affects both peppers and tomatoes, manifesting as circular or irregularly shaped lesions on the leaves. In advanced stages, these lesions become slightly sunken, dark brown, and scab-like in texture. If numerous, the spots can coalesce and grow together. The primary consequences include leaf lesions, leaf drop, fruit lesions, and significantly, a severe reduction in marketable fruit yield. This disease can inflict substantial economic damage [26, 27].

Late Blight: The disease caused by the fungus Phytophthora infestans was first observed in 1843 in Philadelphia and New York. Late blight can easily affect leaves other than the initially infected ones. It begins as irregular greenish spots on the fruit, giving it the appearance of frost damage. In humid conditions, a white, fluffy growth appears on the underside of leaves. Infected fruits rapidly become foul-smelling masses. Late blight typically occurs in mid to late August during cool nights. This fungus also affects potatoes and can spread from potatoes to tomatoes. Without intervention, it can devastate all plants in a cultivation area within 7 to 10 days [28, 29].

Powdery Mildew: First observed in the United Kingdom in 1986, powdery mildew has since spread worldwide.

This fungal infection, commonly known as powdery mildew, affects the above-ground parts of the plant but does not infect the fruit. It causes white lesions on leaves and significantly reduces the amount of marketable produce, despite not spreading to the fruit [30, 31].

Early Blight: Host invasion is facilitated by enzymes that break down the host's cell wall and a series of toxins that kill host cells and allow the pathogen to feed on the dead cells. Lesions become visible 2-3 days after infection, and spore production occurs within 3-5 days. This disease, caused by the fungus Alternaria solani, leads to the formation of spots on leaves, stems, fruits, and petioles [32].

3.2. Machine Learning Models

In the realm of precision agriculture, the deployment of machine learning algorithms stands at the forefront of innovative approaches to enhance crop health monitoring and disease management. This study leverages a suite of sophisticated machine learning algorithms, namely k-Nearest Neighbors (KNN), Random Forest (RF), Logistic Regression (LR), Support Vector Machine (SVM), and Neural Networks (NN), to automate the detection and classification of diseases in tomato plant leaves. These algorithms are selected for their proven efficacy in pattern recognition and classification tasks, promising to significantly improve the accuracy and efficiency of disease diagnosis in agricultural practices. The following sections will delve into the theoretical underpinnings and practical applications of each algorithm, illustrating their contributions to the advancement of agricultural technology [33].

Logistic Regression (LR): Logistic Regression is a paramount statistical and data mining technique utilized by statisticians and researchers for the analysis and classification of datasets. It offers a method to classify according to probability rules by calculating the estimated values of the dependent variable as probabilities. Some of the principal advantages of LR include its inherent capability to provide probabilities and its applicability to multi-class classification problems [34, 35].

K-Nearest Neighbors (KNN): KNN is an algorithm used for classification in supervised learning. The algorithm initially selects a reference value as the number of data points close to the desired data (K value) and then finds the points closest to it using the Euclidean formula. For instance, if we have two classes, A and B, and we input new data with K=10, if the algorithm finds 6 instances of B and 4 of A, then the new data point is classified as belonging to class B (Figure 2). It performs exceptionally well with data classes where the relationship between variables is complex [36, 37].

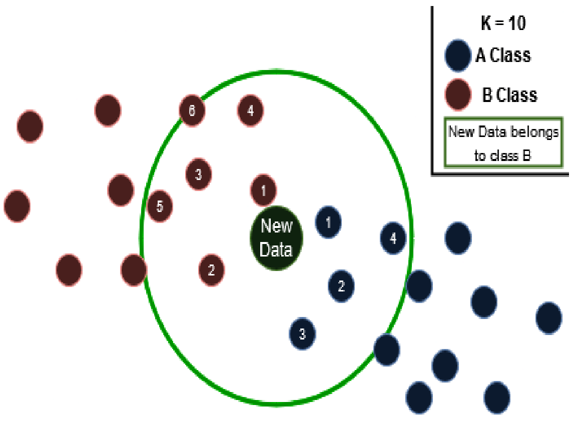


Figure 2. The KNN Model

Random Forest (RF): Fundamentally, Random Forest is constituted by the collective use of a series of decision trees. Decision trees attempt to divide the dataset using a tree structure and eventually reach a conclusion or prediction (Figure 3). The tree structure begins at the root node, continuing to split into branches according to certain criteria. Each branch represents a decision point and may culminate in a class label or prediction value. RF is utilized to assess the outcome of all decision trees to reach a more accurate conclusion [33, 38-40].

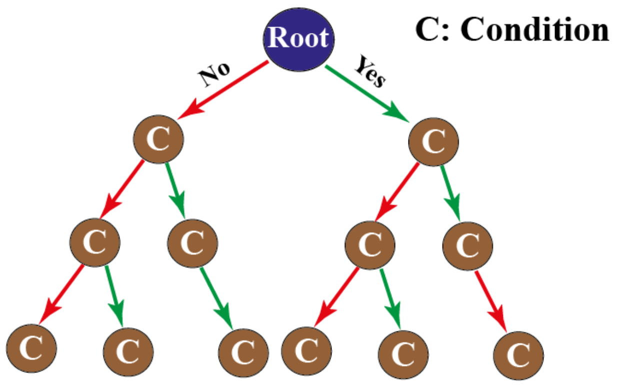


Figure 3. Decision Tree

Support Vector Machine (SVM): SVM is a machine learning algorithm predominantly used for classification problems. It determines decision boundaries to classify data accordingly. SVM can operate with boundaries in various dimensions, such as lines, planes, or hyper-planes. Class labels are used to determine which class data points belong to, based on their position relative to these boundaries (Figure 4). SVM is also applicable to non-linear datasets [41, 42].

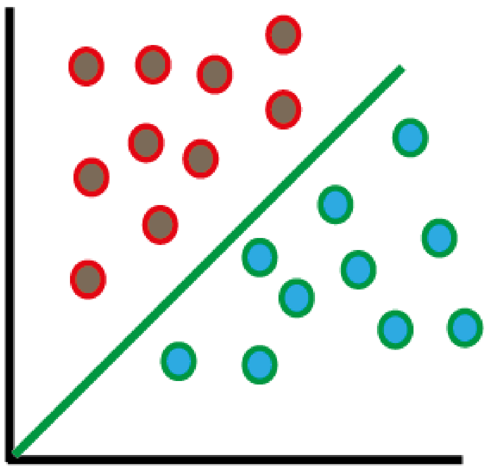


Figure 4. SVM Model

Neural Network (NN): A model that emulates the functioning system of the human brain. Currently, "feedforward" forms are predominantly used. Data enters the network through the input layer. Each node in every layer has its own weight multiplier and threshold value. These values are selected randomly when the network is initially created. Every activated node at the threshold receives distinct data from each connected node. The received data are multiplied by the node's weight, summed up, and it's checked whether they meet the threshold value of the next node. If the next layer's threshold is met, this sequence of operations is repeated until the output layer. These intermediate layers are referred to as "hidden layers" and are not limited in number. Input data processed by all hidden layers generates an output. This process continues until data with the same labels produce similar outputs (Figure 5). It is a model of supervised learning [43].

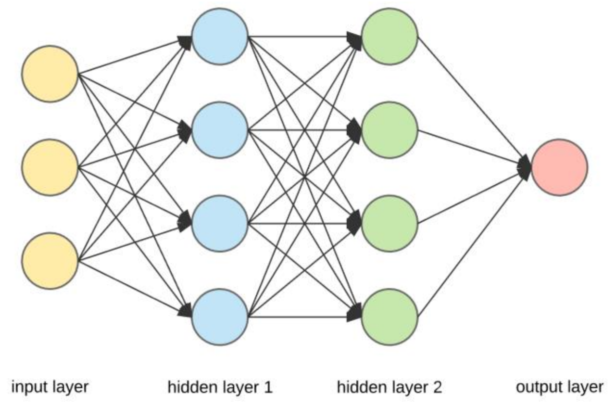


Figure 5. Neural Network Model

3.3. Development of CNN model

Convolutional Neural Networks (CNN) are deep learning algorithms used for analyzing visual data through a series of layers to effectively identify and separate features [44]. In the application of Orange, the implementation of a Convolutional Neural Network (CNN) algorithm demonstrated a remarkable accuracy rate of 99.6%. To further streamline the process of model preservation and to augment the control over the model, this CNN model has been meticulously transcribed into code using the Python programming language, specifically leveraging the Tensorflow and Keras libraries for this purpose. As an initial step, a selection of pre-designed models housed within the Keras library were subjected to a process of transfer learning. This approach allowed these models to be adeptly trained and then rigorously tested against the dataset at hand, ensuring a comprehensive evaluation of their performance.

VGG-16: The Visual Geometry Group (VGG) has developed a network model renowned for its depth and architectural complexity. This model, known specifically as VGG16, is characterized by its assembly of sixteen 3x3 convolutional layers, which are strategically interspersed with a total of five 2x2 max-pooling layers, positioned after each set of convolutional operations to reduce spatial dimensions and to enhance feature extraction. Following the final max-pooling layer, the architecture transitions into a sequence of three fully connected layers, where the initial two layers boast 4096 channels each, culminating in a third layer equipped with 1000 channels. This design facilitates a comprehensive processing of image features before culminating in a SoftMax layer, which is employed for the classification task. Figure 3 illustrates the simplified architecture of the VGG16 model, showcasing its layered structure and the flow of data through the network [45, 46].

Transfer Learning with VGG-16 in the Python-Keras Library: Initially, the VGG-16 model, available within the Keras library, was enhanced by incorporating several fully connected layers, resulting in a refined model. The VGG-16 model (Figure 6) commenced with "imagenet" model weights and adapted through additional layers to learn from the current dataset. Essential parameters within the VGG-16 model, such as the activation function for convolutional layers and the input size for images, are predetermined. The ReLU activation function, provided by the Keras library, is utilized as demonstrated in Formula 1.

$$relu(x) = \frac{x + |x|}{2} \tag{1}$$

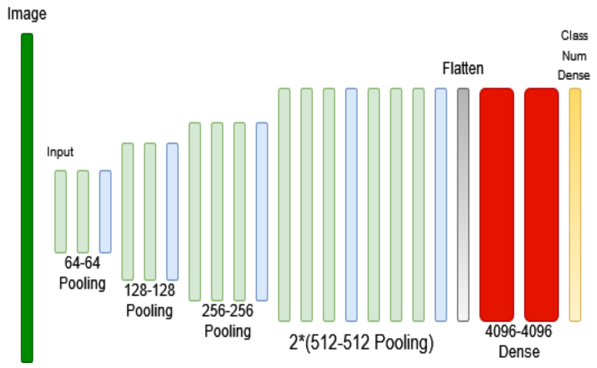


Figure 6. VGG-16 model structure

To ascertain the model's response and enhance its performance, various combinations of activation functions for the fully connected layers, the activation function for the output layer of the fully connected layers, and the optimizer used during the model compilation were tested within Keras.

The initial parameter tested was the model optimizer. Various optimizers available within the Keras library, including Adam, NAdam, Adadelta, RMSProp, and SGD, were evaluated. During testing, the activation function for the fully connected layer was fixed to LeakyReLU (Formula 2), and the activation function for the output layer of the fully connected layers was set to Softmax [47]. Under these conditions, the SGD optimizer achieved the highest accuracy rate of 83%.

$$L - relu(x) = \frac{a \times x + |x|}{2} \qquad (2)$$

Given the optimum performance achieved with the SGD formula as the optimizer, experiments were conducted with the other two parameters. The next test parameter was the activation functions for the fully connected layers. During the optimizer selection phase, where LeakyReLU was initially chosen, various activation functions available in the Keras library, including ReLU, Sigmoid, Softmax, Softplus, Softsign, Tanh, SELU, ELU, and Exponential, were tested. In these tests, the optimizer was SGD, and the activation function for the output layer of the fully connected layers was Softmax [48]. The activation function test for the fully connected layers achieved the highest success with an accuracy rate of 87% using the Softsign function (Formula 3).

$$softsign(x) = \frac{x}{|x| + 1} \qquad (3)$$

The performance tests identified the most promising solutions for both the optimizer and the activation function for the fully connected layers. The final parameter to be tested was the activation function for the output layer of the fully connected layers. During this test, the optimizer was set to SGD, and the activation function for the fully connected layers was determined to be Softsign. The activation functions used in the testing of the fully connected layers remained valid during this testing phase [49]. The highest success was achieved with an accuracy rate of 88% using the Sigmoid activation function (Formula 4).

$$sigmoid(x) = \frac{1}{1 + e^{-x}} \qquad (4)$$

Based on the results of the performance tests, the most optimal outcome for the CNN model developed through the transfer learning method using the VGG-16 with fully connected layers for the current dataset is achieved by employing Softsign as the activation function for the fully connected layers, Sigmoid as the activation function for the output layer of the fully connected layers, and SGD as the optimizer during model compilation.

This development process also incorporated other pre-designed models available in Keras, aside from VGG-16. The models subjected to testing included VGG-19, DenseNet-121, ResNet-152, RegNet-320X, NASNetLarge, and ConvNextBase. Among these, the VGG-19 model exhibited the highest efficiency with an accuracy rate of 85%, while the other models failed in the performance test due to their high percentages in training and prediction losses.

CNN In Layers: Upon the completion of the transfer learning process, the resulting model was recorded to have an approximate size of 60 MB with an accuracy rate of 85%. When assessing the ratio of model size to its efficacy, it is hypothesized that an evaluation and enhancement of each layer's parameters could lead to improvements in at least one of the outcomes, be it model size or accuracy rate. In alignment with this hypothesis, instead of utilizing pre-built models available within the Keras library, a comprehensive model was constructed layer by layer utilizing the model layers found in Keras. The ensuing sections will detail each Keras layer utilized in the development of the model, including explanations for their selection and integration.

Keras-Sequential Layer: The Sequential layer serves as a linear framework for Keras models, facilitating the orderly compilation of the model's functional layers according to their sequence of addition. It can be conceptualized as a list specifying the layers to be executed for the model (Figure 7).

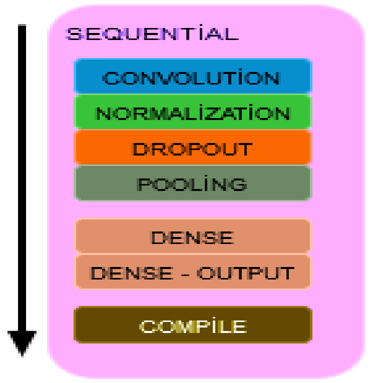


Figure 7. Sequential Layer

Keras-Convolution Layer: The convolution layer is instrumental in transforming the pixel values of an input image through a convolution process, thereby generating a "feature" matrix. During this process, a kernel matrix traverses across the pixels of the image by its dimensions, at each step multiplying its values with the current pixel values of the image, summing these products, and inscribing the result into an outcome matrix.

There are three critical parameters within the convolution layer. The first parameter is the filter size, which dictates the dimensions of the resulting feature matrix. Consequently, setting the filter size too low or too high in relation to the input size can adversely affect the learning process. The second significant parameter is the kernel matrix size, represented by a pair of integers that determine the dimensions of the matrix as it steps across the image. A kernel matrix that is too large can consume an excessive amount of computational power. Another parameter of paramount importance in the convolution layer is the activation function. This function is engaged following the convolution operations between the kernel matrix and the image, at the point when values are to be recorded into the feature matrix. The results of the convolution-processed values are relayed to this activation function, whose output is then documented in the result matrix. An example of a convolution layer can be seen in the following illustration (Figure 8) [50].

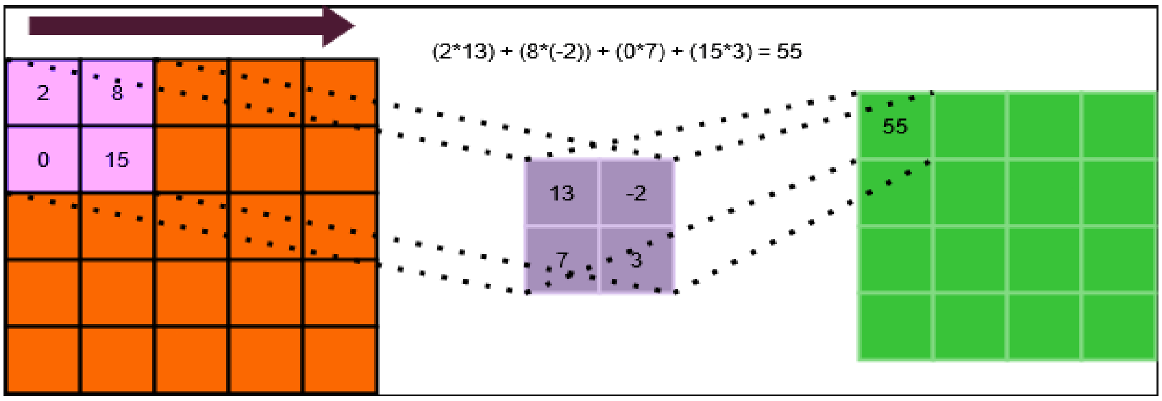


Figure 8. Convolution Layer

Keras-BatchNormalization Layer: The batch normalization layer normalizes its incoming inputs towards a normal distribution, aiming to adjust the mean of the data to zero and its standard deviation to one. This layer behaves differently during training and prediction phases; during training, it processes inputs dynamically based on the output from the preceding layers, whereas, during prediction, it utilizes fixed values obtained from the training phase. The batch normalization process involves calculating the mean and variance of the values entering the batch layer. The current value is then subtracted by the mean, and a constant epsilon number (~1x10^(-5)) is added to the variance before dividing by the square root of this sum. The resultant value is multiplied by a learned gamma value and added to another learned value, beta (Formula 5). The term "learned parameters" refers to the ability of batch normalization to update these values autonomously in response to incoming data during the training period [51].

$$BN(x) = \gamma \times \frac{x - mean(x)}{\sqrt{(var(x) + \in)}} + \beta \qquad (5)$$

Keras-Pooling Layer: The pooling layer reprocesses the matrix given as input, using a pooling matrix of a specified size. Among the most favored types of pooling are max pooling, which selects the highest value, and average pooling, which calculates the mean value within the pool. In max pooling, the size of the pooling matrix is typically 2x2, which means the image is scanned with a 2x2 matrix, and at each step, the highest value within that kernel matrix is recorded into the result matrix (Figure 9) (Max Pooling2D Layer*). Conversely, in average pooling, the determined kernel matrix records the average of its contents into the result matrix at each step (Figure 10) (Average Pooling2D Layer*) [52].

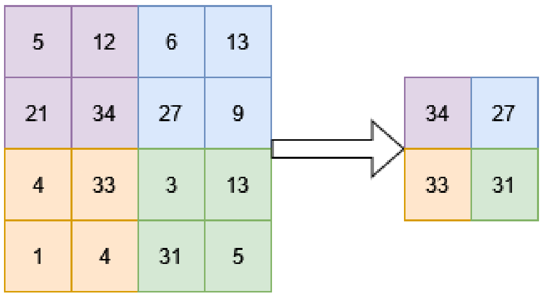


Figure 9. Max Pooling Layer

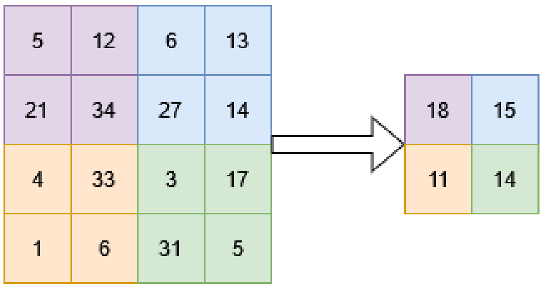


Figure 10. Average Pooling Layer

Keras-Dropout Layer: The dropout layer nullifies the input values at a predetermined rate. The input values of the layers that are not zeroed are incremented by a factor of 1/(1−rate)1/(1−rate). This mechanism ensures the conservation of the total sum of inputs (Figure 11).

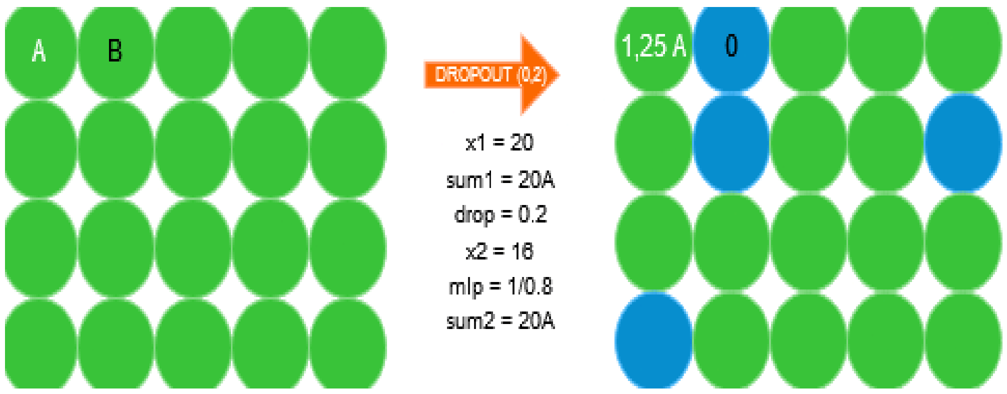


Figure 11. Dropout Layer

Keras-Flatten Layer: The flatten layer reduces the multi-dimensional input values to a single dimension, thereby flattening the input. Consequently, this process transforms the data into a format that can be processed by fully connected layers.

Keras-Dense Layer: Fully connected layers are a critical component of the neural network architecture, constructed based on the numerical value entered as a parameter. Each fully connected layer corresponds to a network layer within the neural architecture. The numerical parameter specified for each layer indicates the number of artificial neurons in that layer.

Registration of the Trained Model in the Service and AWS Services: In the initial sections of the materials and methods, it was mentioned that from the plants specified—apple, corn, wheat, tomato, tea, potato, and grape—tomato and wheat were removed from the training and test datasets for the training of the mentioned model. Due to significant variations in the random data from the tomato dataset, the model's predictive accuracy was notably adversely affected. Consequently, the tomato dataset was entirely removed from the training dataset. The inconsistency in the size of the data from the wheat dataset, coupled with the loss of the diseased portion that the model needs to learn when dimensional normalization was performed, necessitated its complete removal from the training dataset as well. The model, trained with the remaining classes, has been saved with a ".h5" extension. The completed model has been uploaded to the AWS Lambda platform for predictive operations, enabling its use by the mobile application.

3.4. Cross-validation

Cross-validation is a method of data partitioning employed to assess the generalization capability of models and to prevent overfitting. Traditionally, the data splitting approach reserves between 10-30% of the dataset for testing, while 70-90% is allocated for training. If the dataset is sufficiently large, the resulting training and testing data will also be substantial, thereby enhancing the reliability of the observed test error [53].

In the cross-validation system, the data is divided into i segments. One segment is isolated for testing, and the remaining $i - 1$ segments are utilized for training. Subsequently, the segment previously used for testing is included in the training, another segment is set aside for testing, and the model is retrained. This cycle continues until each segment has participated in testing and the training is complete, iterating i times. Consequently, the average of each classification outcome is calculated, and this mean value is recorded as the measure of the classification model's success [54, 55].

3.5. Confusion Matrix and Evaluation Metrics

The confusion matrix succinctly summarizes the actual versus predicted classifications of a classification model in a tabular format. This table displays the counts of instances correctly and incorrectly classified by the model, thereby providing a clearer understanding of the model's performance [56-58].

The term true positive (TP) denotes the positive examples that the model has correctly classified. Likewise, the term true negative (TN) represents the negative examples that have been accurately classified. The terms false positive (FP) and false negative (FN), on the other hand, refer to the positive and negative examples that the model has misclassified, respectively [59, 60]. Table 1 presents a 5-class confusion matrix.

Table 1. Five Class confusion matrix

		PREDICTED				
		Bacterial Spot	Early Blight	Late Blight	Healthy	Powdery Mildew
ACTUAL	Bacterial Spot	$\mathbf{T_1}$	F_{12}	F_{13}	F_{14}	F_{15}
	Early Blight	F_{21}	$\mathbf{T_2}$	F_{23}	F_{24}	F_{25}
	Late Blight	F_{31}	F_{32}	$\mathbf{T_3}$	F_{34}	F_{35}
	Healthy	F_{41}	F_{42}	F_{43}	$\mathbf{T_4}$	F_{45}
	Powdery Mildew	F_{51}	F_{52}	F_{53}	F_{54}	$\mathbf{T_5}$

In addition to accuracy, precision, recall, F-score, and specificity are other preferred metrics for evaluating the performance of classification models [61]. The calculation of performance metrics for 5 classes is provided in Table 2.

Table 2 Performance metrics equations for five-class.

METRICS	EQUANTION
Average accuracy	$\frac{TP_i + TN_i}{TP_i + FP_i + TN_i + FN_i}$
Average precision (P)	$\frac{TP_i}{TP_i + FP_i}$
Average recall (R)	$\frac{TP_i}{TP_i + FN_i}$
Average F-score	$\frac{2 \times P_i \times R_i}{P_i + R_i}$
Average specificity	$\frac{TN_i}{FP_i + TN_i}$

4. Experiments and Results

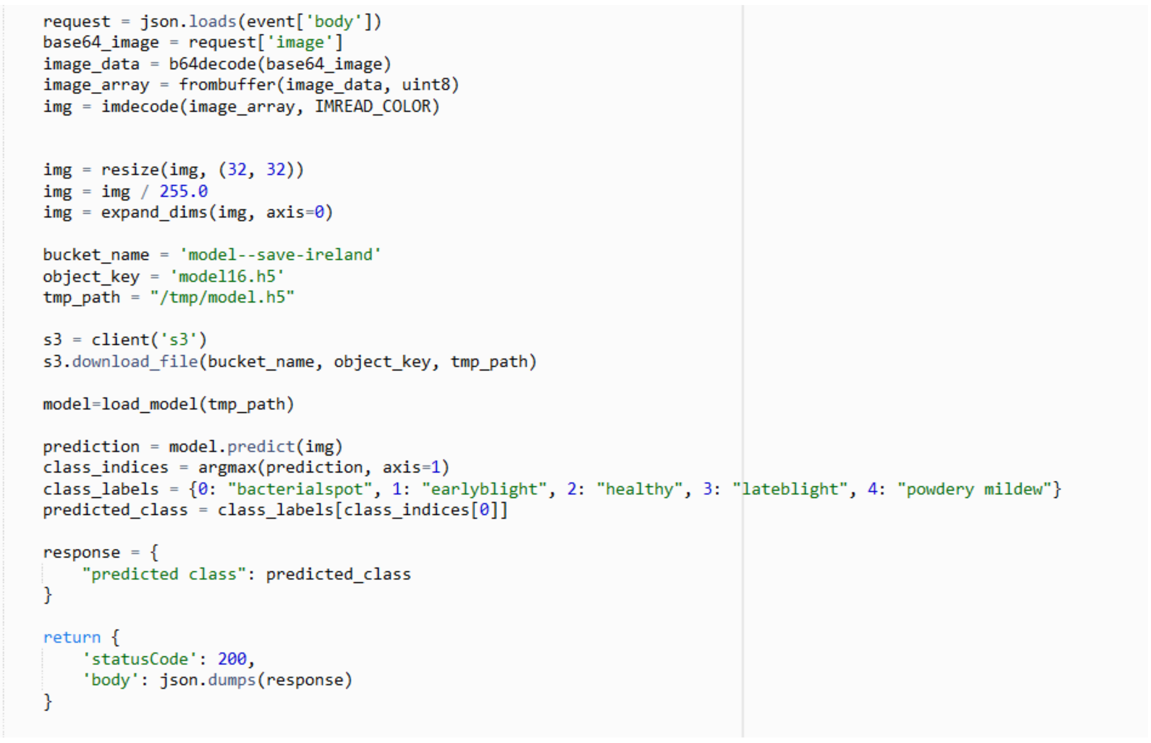
In this study, it was investigated how feature extraction and classification processes can be optimized through the integration of deep learning and machine learning techniques over a data set. Firstly, features were extracted from the samples in the dataset using the VGG16 model, which is one of the convolutional neural networks and is widely accepted for visual feature extraction. This deep learning model has shown high success, especially in the analysis of visual content, and has found a wide range of applications. The obtained characteristics were used for comparative evaluation of the performance of five different machine learning models - KNN, RF, SVM, NN and LR. These models are widely preferred methods for classification problems and have given successful results in applications in various fields. The main purpose of the study is to determine the most appropriate classification model for the data set by evaluating the performance of different machine learning algorithms on the same feature set. This process lays the foundations of a methodology that will be an important guide in model selection and can be used in future studies. The results obtained are given in Table 3.

Table 3. The results obtained from machine learning models as a result of feature extraction

Model	**Accuracy**	**F-score**	**Precision**	**Recall**	**Specificity**
SVM	0.812	0.811	0.813	0.812	0.761
KNN	0.889	0.887	0.891	0.889	0.859
Neural Network	0.956	0.956	0.956	0.956	0.943
Random Forest	0.888	0.888	0.888	0.888	0.856
Logistic Regression	0.897	0.897	0.898	0.897	0.869

This dataset presents a comparative analysis of various classification models based on metrics like Accuracy, F-score, Precision, Recall, and Specificity. The Neural Network model demonstrates superior performance across all metrics, with an impressive accuracy and F-score of 0.956, suggesting its exceptional capability in both precision and handling true positives. Close behind, the Logistic Regression model shows high efficiency, particularly in balancing precision and recall, evidenced by its nearly uniform scores across the metrics, culminating in an accuracy of 0.897. The KNN and Random Forest models exhibit commendable performance, with accuracies around 0.889, indicating their reliability in classification tasks. The SVM model, while slightly lagging in accuracy at 0.812, still provides a solid baseline for comparison. Overall, these results highlight the strengths and potential areas for improvement of each model in handling classification tasks, with the Neural Network model standing out for its robustness and effectiveness.

The model's architectural framework has been meticulously refined to an optimal level through a series of experimental combinations. To elucidate this process in straightforward terms, the procedure begins with the transformation of incoming data into a format amenable to processing. This is followed by the execution of an image prediction task, employing a model that is dynamically fetched from the S3 storage service. Finally, the outcome of this predictive analysis is disseminated back to the requester through an API interface. The schematic representation of the AWS Lambda Prediction Function is depicted in Figure 12, providing a visual overview of this integrated workflow.



```
request = json.loads(event['body'])
base64_image = request['image']
image_data = b64decode(base64_image)
image_array = frombuffer(image_data, uint8)
img = imdecode(image_array, IMREAD_COLOR)


img = resize(img, (32, 32))
img = img / 255.0
img = expand_dims(img, axis=0)

bucket_name = 'model--save-ireland'
object_key = 'model16.h5'
tmp_path = "/tmp/model.h5"

s3 = client('s3')
s3.download_file(bucket_name, object_key, tmp_path)

model=load_model(tmp_path)

prediction = model.predict(img)
class_indices = argmax(prediction, axis=1)
class_labels = {0: "bacterialspot", 1: "earlyblight", 2: "healthy", 3: "lateblight", 4: "powdery mildew"}
predicted_class = class_labels[class_indices[0]]

response = {
    "predicted class": predicted_class
}

return {
    'statusCode': 200,
    'body': json.dumps(response)
}
```

Figure 12. AWS Lambda Prediction Function

The table delineates a subset of models that have undergone training for the tomato plant dataset, exhibiting both significant variations and satisfactory outcomes in their results. These models include VGG-16 and variants of HM-batch(N) with varying hyperparameters, as detailed in the associated training conditions. It's worth noting that the total number of tests conducted exceeds the quantity presented in the table. In this study, we categorize our dataset into four distinct sizes—Extra Large (XL), Small (S), Medium (M), and Large (L)—to systematically evaluate the impact of data volume on our analysis and outcomes. This selection has been made to highlight models that provide a representative overview of the experimental conditions and outcomes (Table 3).

The provided table (Table 4) delineates a comprehensive evaluation of various deep learning models tailored for tomato plant disease detection, detailing their respective performance metrics and training conditions. The VGG-16 models, occupying the initial two rows of the table, demonstrate a substantial range in both training and prediction accuracy, indicating variability in performance which may be attributed to different data volume (XL for 't-1' and S for 't-2') or perhaps parameter tuning. Despite the smaller dataset, 't-2' yields a higher prediction accuracy, a testament to VGG-16's robustness or possibly more refined training iterations.

Subsequent entries reveal a consistent use of a hybrid model, hereafter referred to as HM-batch(N), across a variety of dataset sizes (S to L). The HM-batch(N) exhibits a commendable consistency in training and prediction accuracy, especially in instances 't-5' through 't-13', where the model achieves over 90% prediction accuracy, a clear indication of the model's effectiveness in generalizing from the given data.

Additionally, the table highlights the disparities in training duration ranging from a mere 9 seconds to an extensive 906 seconds, suggesting that the models' computational demands vary significantly, possibly reflecting the complexity of the model architecture or the efficiency of the training process.

In analyzing training and prediction losses, the HM-batch(N) models maintain a relatively stable loss, with minor variations that do not seem to correlate directly with the size of the training data. This stability in loss metrics may point to a well-regularized model that is resistant to overfitting despite the volume of training data.

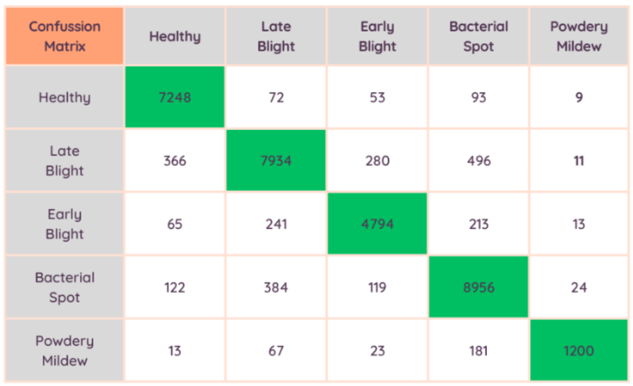
The model's configurations and the corresponding training times suggest a trade-off between model complexity and training efficiency. Models with shorter training times may benefit from simplified architectures or reduced parameter spaces, while those requiring longer training periods likely capitalize on more complex feature representations, which could be essential for capturing nuanced patterns within the data.

In summary, the collated data presented in the table provides valuable insights into the relative performance of different models under varying conditions, serving as a crucial reference for selecting the most appropriate model configuration for real-world application in the domain of plant disease classification. The evaluations conducted provide a substantial foundation for further refinement and optimization of deep learning models in agricultural technology.

Table 4. The Results of the Trained Models

Test Series	Training Accuracy	Predicted Accuracy	Training Loss	Predicted Loss	Training Time	Model Structure	Amount of Data
t-1	89.95	88.46	25.30	22.80	85	VGG-16	XL
t-2	86.77	90.90	32.99	65.98	682	VGG-16	S
t-3	93.83	64.14	17.17	92.39	8	ResNet152V2	S
t-4	93.97	64.86	16.78	84.72	9	HM-batchN()	M
t-5	96.79	91.84	13.20	18.20	85	HM-batchN()	M
t-6	81.84	82.89	45.65	43.57	76	HM-batchN()	L
t-7	96.79	91.87	13.19	18.98	84	HM-batchN()	L
t-8	96.90	95.90	13.18	13.60	84	HM-batchN()	L
t-9	96.98	95.90	13.17	13.40	84	HM-batchN()	L
t-10	96.87	96.94	15.32	15.32	97	HM-batchN()	L
t-11	96.73	95.03	13.20	13.20	84	HM-batchN()	L
t-12	97.04	96.83	13.16	13.13	112	HM-batchN()	L
t-13	94.53	98.05	14.47	36.54	906	HM-batchN()	L

The confusion matrix given in Figure 13 is used to evaluate the prediction success of a model and represents the average of the performances of the t-12 training test and the t-13 prediction test. The matrix includes five different classes: 'Healthy', 'Late Blight', 'Early Blight', 'Bacterial Spot', and 'Powdery Mildew'. The model correctly classified the 'Healthy' class 7,248 times, 'Late Blight' 7,934 times, 'Early Blight' 4,794 times, 'Bacterial Spot' 8,956 times and 'Powdery Mildew' 1,200 times. These values show that the percentage of correct predictions of the model for each class is quite high. However, Deceptions between the 'Late Blight' and 'Bacterial Spot' classes are notable; for example, 'Bacterial Spot' has been misclassified as 'Late Blight' 384 times. In addition, 'Healthy' plants have a relatively low number of false positive and false negative results, which shows that the model distinguishes this class better than others. In general, the model was able to Decipher with high accuracy, but improvements can be made to reduce confusion between some classes.



Confussion Matrix	Healthy	Late Blight	Early Blight	Bacterial Spot	Powdery Mildew
Healthy	7248	72	53	93	9
Late Blight	366	7934	280	496	11
Early Blight	65	241	4794	213	13
Bacterial Spot	122	384	119	8956	24
Powdery Mildew	13	67	23	181	1200

Figure 13. The Complexity Matrix of the Model Written in Layers is

This predictive service has been meticulously designed to interface with a mobile application (Figure 14) via an API connection, the application itself being developed with Flutter and Dart. This sophisticated application is specifically tailored for the classification of tomato leaf diseases, integrating cutting-edge machine learning models to facilitate rapid and accurate diagnosis.

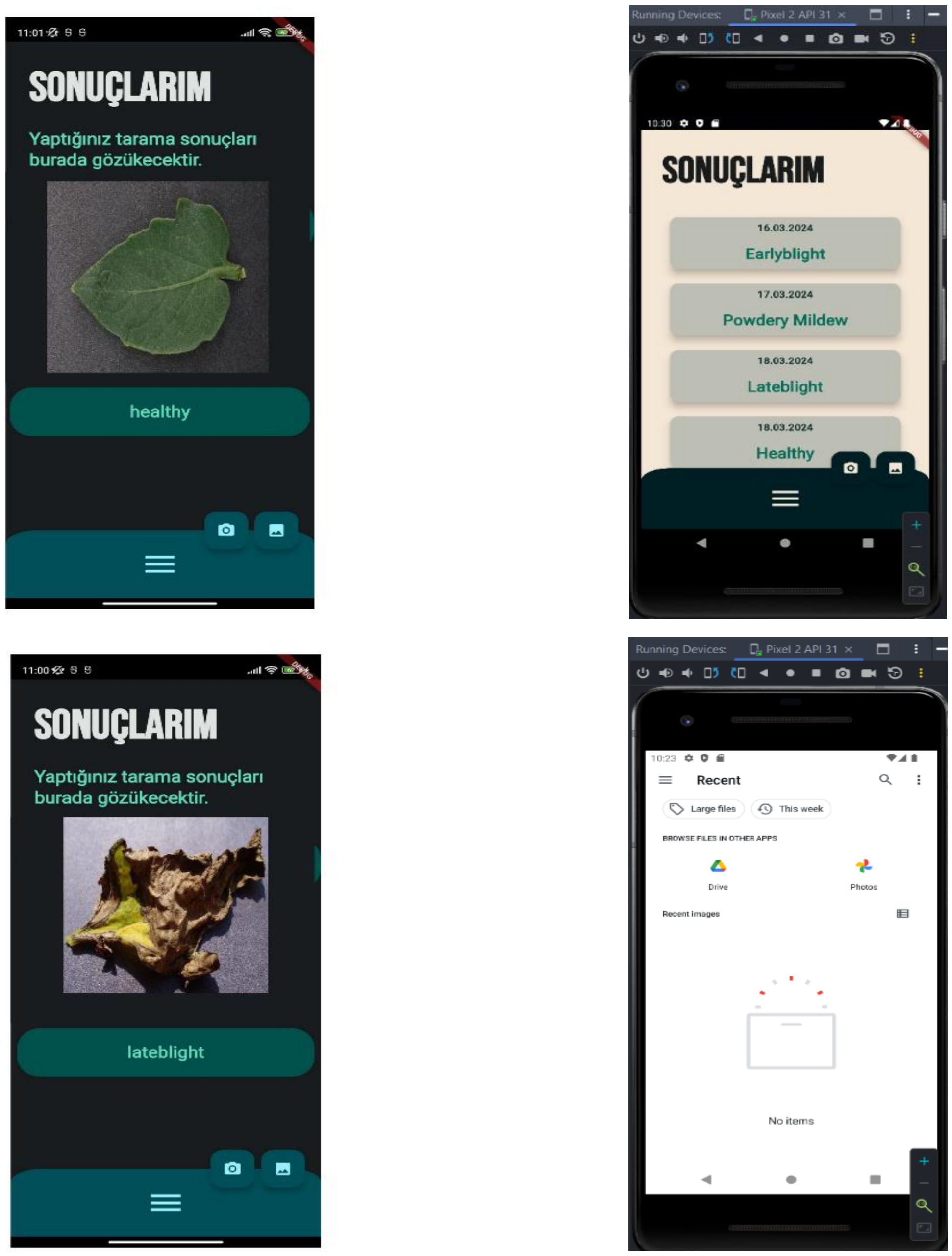
In conclusion, the amalgamation of the most efficacious layers from tests t-12 and t-13 into a singular model has culminated in an exemplary display of classification proficiency, achieving an approximate accuracy of 96%. This achievement is significant, considering the model underwent a meticulous training regimen over approximately 10 hours and was validated across a dataset comprising some 33,000 entries. Comparative analysis among diverse classification methodologies—encompassing Neural Networks, Logistic Regression, KNN, Random Forest, and SVM—underscores the Neural Network's paramount performance, distinguished by its unparalleled accuracy and F-score of 0.956. This metric evidences the model's exceptional capability in precision and true positive handling. Subsequent models, including Logistic Regression, KNN, and Random Forest, also exhibit laudable performance metrics, thereby affirming their applicability and reliability in classification tasks, albeit with the SVM model trailing slightly in terms of accuracy.

This scholarly inquiry not only delineates the potency of deep learning through the lens of Neural Network superiority but also illuminates the nuanced performance characteristics across a spectrum of traditional and contemporary classification models. It thereby contributes to the academic discourse on predictive modeling, advocating for a nuanced approach to model selection and optimization predicated on comprehensive performance evaluation. This research underscores the pivotal role of strategic layer integration in bolstering model performance, thereby charting a path for future explorations into predictive accuracy and model efficiency in the realm of machine learning.

Drawing on the robust foundation of the VGG-16 architecture, the application has been optimized for mobile deployment, ensuring that the complexities of deep neural networks are seamlessly adapted to the constraints and operational paradigms of mobile devices. This optimization entails not only the model’s conversion to a mobile-compatible format but also its synchronization with the mobile application to provide real-time predictive analytics.

The harmonization of the application with the VGG-16 model-based service exemplifies the confluence of mobile technology and artificial intelligence in the pursuit of agricultural health. This synergy allows for the immediate processing and classification of tomato plant diseases, presenting a significant advancement in agricultural informatics. The utility of such an application lies in its ability to offer immediate, field-level diagnostic support to farmers and agronomists, thereby enabling informed decision-making that can lead to timely and effective disease management interventions.

In essence, the developed mobile application serves as an exemplar of how machine learning can be leveraged in practical, user-oriented scenarios, bringing the profound capabilities of AI to the fingertips of users in diverse environments. The integration of such technology into the agricultural domain paves the way for smarter, data-driven approaches to crop management and disease prevention, ultimately contributing to the sustainability and efficiency of food production systems.


SONUÇLARIM

Yaptığınız tarama sonuçları burada gözükecektir.

healthy

SONUÇLARIM

16.03.2024 Earlyblight

17.03.2024 Powdery Mildew

18.03.2024 Lateblight

18.03.2024 Healthy

SONUÇLARIM

Yaptığınız tarama sonuçları burada gözükecektir.

lateblight

Figure 14. Mobile Application Screens

5. Conclusion

In this comprehensive study, we employed Convolutional Neural Networks (CNN), particularly the

VGG16 model, alongside various machine learning algorithms such as KNN, RF, SVM, NN, and LR, to optimize the feature extraction and classification processes for tomato leaf disease detection. The neural network model, in particular, exhibited exceptional performance, achieving an accuracy and F-score of 0.956, standing out among the evaluated models. This high level of accuracy underscores the neural network's capabilities in precision and managing true positives effectively. Following closely, the Logistic Regression, KNN, and Random Forest models also showed commendable performance with accuracy rates around 0.897 and 0.889, respectively, proving their reliability in classification tasks. The integration of these models with a mobile application demonstrates a practical approach to applying advanced computational models in agricultural settings, facilitating timely and accurate disease detection. This research not only highlights the potential of combining deep learning with traditional machine learning techniques but also sets a benchmark for future studies in agricultural informatics and disease management.

The integration of intensive learning styles, such as deep learning, with lightweight hardware components found in mobile applications, heralds a promising avenue for powerful yet accessible solutions. Enhancing the application's user interface with a history section on the result page can render it more user-friendly, allowing for in-field utility and storage of historical data, which can be preserved in conjunction with external storage for long-term analysis. The choice to save results and the determination of the save location would be at the user’s discretion, with the default being the last used storage unit.

Individual growers cultivating such plants in their pots or backyard gardens can also protect their produce by utilizing a mobile application on their personal devices. This system offers both types of users—a commercial grower and a private individual—a swift, efficient, and cost-effective means of disease monitoring.

Given that spraying operations via sensor monitoring and professional support can be economically burdensome, some producers opt for regular pesticide applications. However, this can reduce the nutritional value of plants due to unnecessary chemical exposure and consequently pose risks to consumer health. Additionally, excessive application of chemicals may not only affect the plant itself but also degrade the quality of the cultivated land over time. With the deployment of this application, rather than conducting preemptive and unnecessary spraying, treatment can be initiated precisely when the initial signs of disease are detected.

For future enhancements, a model offering greater control over hyperparameters could be employed. Improvements in both image processing methods and precise hyperparameter tuning will significantly enhance the quality of the model used within the service. While the depth of the model and the power of each model layer usually have a directly proportional effect on the success of the model, overly deep networks and excessively complex layers can complicate model training due to their high computational demands. One of the primary objectives of the project is to maintain the model's agility for mobile execution with minimal power loss, even if predictions are conducted through a service-based model. Therefore, any enhancements in hyperparameters and depth must keep the model size within a specified range to preserve its lightweight nature for mobile utility.

Acknowledgments

This article was extracted from the project numbered 1919B012221354 carried out within the scope of TUBITAK 2209/A UNIVERSITY STUDENTS RESEARCH PROJECTS SUPPORT PROGRAM.

Compliance with ethics requirements

This study does not contain any studies with human participants or animals performed by any of the authors.

Availability of Supporting Data

The dataset used in this study, which was performed to help diagnose tomato plant leaf diseases, can be accessed at https://www.kaggle.com/datasets/ashishmotwani/tomato and https://www.kaggle.com/datasets /kaustubhb999 /tomatoleaf/data.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Thangaraj, R., S. Anandamurugan, P. Pandiyan, and V.K. Kaliappan, *Artificial intelligence in tomato leaf disease detection: a comprehensive review and discussion.* Journal of Plant Diseases and Protection, 2022. **129**(3): p. 469-488. DOI: 10.1007/s41348-021-00500-8.

[2] Joosten, F., Y. Dijkxhoorn, Y. Sertse, and R. Ruben, *How does the fruit and vegetable sector contribute to food and nutrition security?* 2015, LEI. DOI: 10.32604/iasc.2021.016415.

[3] Tarek, H., H. Aly, S. Eisa, and M. Abul-Soud, *Optimized deep learning algorithms for tomato leaf disease detection with hardware deployment.* Electronics, 2022. **11**(1): p. 140. DOI: 10.3390/electronics11010140.

[4] Schreinemachers, P., E.B. Simmons, and M.C. Wopereis, *Tapping the economic and nutritional power of vegetables.* Global food security, 2018. **16**: p. 36-45. DOI: 10.1016/j.gfs.2017.09.005.

[5] Han, L., M.S. Haleem, and M. Taylor. *A novel computer vision-based approach to automatic detection and severity assessment of crop diseases*. in *2015 Science and Information Conference (SAI)*. 2015. IEEE. DOI: 10.1109/SAI.2015.7237209.

[6] Zhang, S., Y. Shang, and L. Wang, *Plant disease recognition based on plant leaf image.* 2015.

[7] Trivedi, N.K., V. Gautam, A. Anand, H.M. Aljahdali, S.G. Villar, D. Anand, N. Goyal, and S. Kadry, *Early detection and classification of tomato leaf disease using high-performance deep neural network.* Sensors, 2021. **21**(23): p. 7987. DOI: 10.3390/s21237987.

[8] Gadade, H.D. and D. Kirange. *Machine learning based*

identification of tomato leaf diseases at various stages of development. in *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*. 2021. IEEE. DOI: 10.1109/ICCMC51019.2021.9418263.

[9] Agarwal, M., A. Singh, S. Arjaria, A. Sinha, and S. Gupta, *ToLeD: Tomato leaf disease detection using convolution neural network.* Procedia Computer Science, 2020. **167**: p. 293-301. DOI: 10.1016/j.procs.2020.03.225.

[10] Ashok, S., G. Kishore, V. Rajesh, S. Suchitra, S.G. Sophia, and B. Pavithra. *Tomato leaf disease detection using deep learning techniques*. in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*. 2020. IEEE. DOI: 10.1109/ICCES48766.2020.9137986.

[11] Kaushik, M., P. Prakash, R. Ajay, and S. Veni. *Tomato leaf disease detection using convolutional neural network with data augmentation*. in *2020 5th International Conference on Communication and Electronics Systems (ICCES)*. 2020. IEEE. DOI: 10.1109/ICCES48766.2020.9138030.

[12] Jiang, D., F. Li, Y. Yang, and S. Yu. *A tomato leaf diseases classification method based on deep learning*. in *2020 chinese control and decision conference (CCDC)*. 2020. IEEE. DOI: 10.1109/CCDC49329.2020.9164457.

[13] Das, D., M. Singh, S.S. Mohanty, and S. Chakravarty. *Leaf disease detection using support vector machine*. in *2020 International Conference on Communication and Signal Processing (ICCSP)*. 2020. IEEE. DOI: 10.1109/ICCSP48568.2020.9182128.

[14] Wang, X. and J. Liu, *Tomato anomalies detection in greenhouse scenarios based on YOLO-Dense.* Frontiers in Plant Science, 2021. **12**: p. 634103. DOI: 10.3389/fpls.2021.634103.

[15] Kibriya, H., R. Rafique, W. Ahmad, and S. Adnan. *Tomato leaf disease detection using convolution neural network*. in *2021 International Bhurban Conference on Applied Sciences and Technologies (IBCAST)*. 2021. IEEE. DOI: 10.1109/IBCAST51254.2021.9393311.

[16] Kursun, R., K.K. Bastas, and M. Koklu, *Segmentation of dry bean (Phaseolus vulgaris L.) leaf disease images with U-Net and classification using deep learning algorithms.* European Food Research and Technology, 2023. **249**(10): p. 2543-2558. DOI: 10.1007/s00217-023-04319-5.

[17] Anandhakrishnan, T. and S. Jaisakthi, *Deep Convolutional Neural Networks for image based tomato leaf disease detection.* Sustainable Chemistry and Pharmacy, 2022. **30**: p. 100793. DOI: https://doi.org/10.1016/j.scp.2022.100793.

[18] Rahman, S.U., F. Alam, N. Ahmad, and S. Arshad, *Image processing based system for the detection, identification and treatment of tomato leaf diseases.* Multimedia Tools and Applications, 2023. **82**(6): p. 9431-9445. DOI: 10.1007/s11042-022-13715-0.

[19] Janarthan, S., S. Thuseethan, S. Rajasegarar, and J. Yearwood, *P2OP—Plant Pathology on Palms: A deep learning-based mobile solution for in-field plant disease detection.* Computers and Electronics in Agriculture, 2022. **202**: p. 107371. DOI: 10.1016/j.compag.2022.107371.

[20] Sujatha, R., J.M. Chatterjee, N. Jhanjhi, and S.N. Brohi, *Performance of deep learning vs machine learning in plant leaf disease detection.* Microprocessors and Microsystems, 2021. **80**: p. 103615. DOI: 10.1016/j.micpro.2020.103615.

[21] Sardogan, M., A. Tuncer, and Y. Ozen. *Plant leaf disease detection and classification based on CNN with LVQ algorithm*. in *2018 3rd international conference on computer science and engineering (UBMK)*. 2018. IEEE. DOI: 10.1109/UBMK.2018.8566635.

[22] Harakannanavar, S.S., J.M. Rudagi, V.I. Puranikmath, A. Siddiqua, and R. Pramodhini, *Plant leaf disease detection using computer vision and machine learning algorithms.* Global Transitions Proceedings, 2022. **3**(1): p. 305-310. DOI: 10.1016/j.gltp.2022.03.016.

[23] De Luna, R.G., E.P. Dadios, and A.A. Bandala. *Automated image capturing system for deep learning-based tomato plant leaf disease detection and recognition*. in *TENCON 2018-2018 IEEE Region 10 Conference*. 2018. IEEE. DOI: 10.1109/TENCON.2018.8650088.

[24] Deng, Y., H. Xi, G. Zhou, A. Chen, Y. Wang, L. Li, and Y. Hu, *An effective image-based tomato leaf disease segmentation method using MC-UNet.* Plant Phenomics, 2023. **5**: p. 0049. DOI: 10.34133/plantfenomik.0049.

[25] Ally, N.M., H. Neetoo, V.M. Ranghoo-Sanmukhiya, and T.A. Coutinho, *Greenhouse-grown tomatoes: microbial diseases and their control methods: a review.* International Journal of Phytopathology, 2023. **12**(1): p. 99-127. DOI: 10.33687/phytopath.012.01.4273.

[26] Soto-Caro, A., G.E. Vallad, K.V. Xavier, P. Abrahamian, F. Wu, and Z. Guan, *Managing bacterial spot of tomato: do chemical controls pay off?* Agronomy, 2023. **13**(4): p. 972. DOI: 10.3390/agronomy13040972.

[27] Sharma, S. and K. Bhattarai, *Progress in developing bacterial spot resistance in tomato.* Agronomy, 2019. **9**(1): p. 26. DOI: 10.3390/agronomy9010026.

[28] GM, S.K., S. Sriram, R. Laxman, and K. Harshita, *Tomato late blight yield loss assessment and risk aversion with resistant hybrid.* Journal of Horticultural Sciences, 2022. **17**(2): p. 411-416. DOI: 10.24154/jhs.v17i2.1105.

[29] Hong, Y.-H., J. Meng, X.-L. He, Y.-Y. Zhang, and Y.-S. Luan, *Overexpression of MiR482c in tomato induces enhanced susceptibility to late blight.* Cells, 2019. **8**(8): p. 822. DOI: 10.3390/cells8080822.

[30] Sunarti, S., C. Kissoudis, Y. Van Der Hoek, H. Van Der Schoot, R.G. Visser, V. Der Linden, C. Gerard, C. Van De Wiel, and Y. Bai, *Drought stress interacts with powdery mildew infection in tomato.* Frontiers in Plant Science, 2022. **13**: p. 845379. DOI: 10.3389/fpls.2022.845379.

[31] Wang, H., W. Gong, Y. Wang, and Q. Ma, *Contribution of a WRKY transcription factor, ShWRKY81, to powdery mildew resistance in wild tomato.* International Journal of Molecular Sciences, 2023. **24**(3): p. 2583. DOI: 10.3390/ijms24032583.

[32] Gupta, V., V. Razdan, S. Sharma, and K. Fatima, *Progress and severity of early blight of tomato in relation to weather variables in Jammu province.* Journal of Agrometeorology, 2020. **22**(2): p. 198-202. DOI: 10.54386/jam.v22i2.168.

[33] Koklu, M., R. Kursun, E.T. Yasin, and Y.S. Taspinar. *Detection of Defects in Soybean Seeds by Extracting Deep Features with SqueezeNet*. in *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. 2023. DOI: 10.1109/IDAACS58523.2023.10348939.

[34] Torres, G.d.O., M.X. Guterres, and V.R.R. Celestino, *Legal actions in Brazilian air transport: A machine learning and multinomial logistic regression analysis.* Frontiers in Future Transportation, 2023. **4**: p. 1070533. DOI: 10.3389/ffutr.2023.1070533.

[35] Kursun, R., I. Cinar, Y.S. Taspinar, and M. Koklu. *Flower Recognition System with Optimized Features for Deep Features*. in *2022 11th Mediterranean Conference on Embedded Computing (MECO)*. 2022. DOI: 10.1109/MECO55406.2022.9797103.

[36] Cakir, M., M. Yilmaz, M.A. Oral, H.O. Kazanci, and O. Oral, *Accuracy assessment of RFerns, NB, SVM, and kNN machine learning classifiers in aquaculture.* Journal of King Saud University-Science, 2023. **35**(6): p. 102754. DOI: 10.1016/j.jksus.2023.102754.

[37] Koklu, M. and K. Sabancı, International Journal of Intelligent Systems and Applications in Engineering, 2016. **4**(Special Issue-1): p. 249-251. DOI: 10.18201/ijisae.281901.

[38] Cen, H., D. Huang, Q. Liu, Z. Zong, and A. Tang, *Application Research on Risk Assessment of Municipal Pipeline Network Based on Random Forest Machine Learning Algorithm.* Water, 2023. **15**(10): p. 1964. DOI: 10.3390/w15101964.

[39] Butuner, R., I. Cinar, Y.S. Taspinar, R. Kursun, M.H. Calp, and M. Koklu, *Classification of deep image features of lentil varieties with machine learning techniques.* European Food Research and Technology, 2023. **249**(5): p. 1303-1316. DOI: 10.1007/s00217-023-04214-z.

[40] Taspinar, Y.S., I. Cinar, and M. Koklu, *Prediction of computer type using benchmark scores of hardware units.* Selcuk University Journal of Engineering Sciences, 2021. **20**(1): p. 11-17.

[41] Pisner, D.A. and D.M. Schnyer, *Support vector machine*, in *Machine learning*. 2020, Elsevier. p. 101-121. DOI: 10.1016/B978-0-12-815739-8.00006-7.

[42] Tutuncu, K., I. Cinar, R. Kursun, and M. Koklu. *Edible and Poisonous Mushrooms Classification by Machine Learning Algorithms*. in *2022 11th Mediterranean Conference on Embedded Computing (MECO)*. 2022. DOI: 10.1109/MECO55406.2022.9797212.

[43] Agatonovic-Kustrin, S. and R. Beresford, *Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research.* Journal of pharmaceutical and biomedical analysis, 2000. **22**(5): p. 717-727. DOI: 10.1016/S0731-7085(99)00272-1.

[44] Kursun, R., E.T. Yasin, and M. Koklu, *The Effectiveness of Deep Learning Methods on Groundnut Disease Detection.*

[45] Geng, L., S. Zhang, J. Tong, and Z. Xiao, *Lung segmentation method with dilated convolution based on VGG-16 network.* Computer Assisted Surgery, 2019. **24**(sup2): p. 27-33. DOI: 10.1080/24699322.2019.1649071.

[46] Kursun, R. and M. Koklu. *Enhancing Explainability in Plant Disease Classification using Score-CAM: Improving Early Diagnosis for Agricultural Productivity*. in *2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. 2023. DOI: 10.1109/IDAACS58523.2023.10348713.

[47] Dubey, A.K. and V. Jain. *Comparative study of convolution neural network's relu and leaky-relu activation functions*. in *Applications of Computing, Automation and Wireless Systems in Electrical Engineering: Proceedings of MARC 2018*. 2019. Springer. DOI: 10.1007/978-981-13-6772-4_76.

[48] Waoo, A.A. and B.K. Soni. *Performance analysis of sigmoid and relu activation functions in deep neural network*. in *Intelligent Systems: Proceedings of SCIS 2021*. 2021. Springer. DOI: 10.1007/978-981-16-2248-9_5.

[49] Gulli, A. and S. Pal, *Deep learning with Keras*. 2017: Packt Publishing Ltd.

[50] Albawi, S., T.A. Mohammed, and S. Al-Zawi. *Understanding of a convolutional neural network*. in *2017 international conference on engineering and technology (ICET)*. 2017. Ieee. DOI: 10.1109/ICEngTechnol.2017.8308186.

[51] Kolarik, M., R. Burget, and K. Riha. *Comparing Normalization Methods for Limited Batch Size Segmentation Neural Networks*. in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*. 2020. DOI: 10.1109/TSP49548.2020.9163397.

[52] Ebert-Uphoff, I., R. Lagerquist, K. Hilburn, Y. Lee, K. Haynes, J. Stock, C. Kumler, and J.Q. Stewart, *CIRA Guide to Custom Loss Functions for Neural Networks in Environmental Sciences--Version 1.* arXiv preprint arXiv:2106.09757, 2021.

[53] Berrar, D., *Cross-validation*. 2019.

[54] Xu, Y. and R. Goodacre, *On splitting training and validation set: a comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning.* Journal of analysis and testing, 2018. **2**(3): p. 249-262. DOI: 10.1007/s41664-018-0068-2.

[55] Isik, M., B. Ozulku, R. Kursun, Y.S. Taspinar, I. Cinar, E.T. Yasin, and M. Koklu, *Automated classification of hand-woven and machine-woven carpets based on morphological features using machine learning algorithms.* The Journal of The Textile Institute: p. 1-10. DOI: 10.1080/00405000.2024.2309694.

[56] Deepak, S. and P. Ameer, *Brain tumor classification using deep CNN features via transfer learning.* Computers in biology and medicine, 2019. **111**: p. 103345. DOI: 10.1016/j.compbiomed.2019.103345.

[57] Gencturk, B., S. Arsoy, Y.S. Taspinar, I. Cinar, R. Kursun, E.T. Yasin, and M. Koklu, *Detection of hazelnut varieties and development of mobile application with CNN data fusion feature reduction-based models.* European Food Research and Technology, 2024. **250**(1): p. 97-110. DOI: 10.1007/s00217-023-04369-9.

[58] Taspinar, Y.S., M. Koklu, and M. Altin, *Classification of flame extinction based on acoustic oscillations using artificial intelligence methods.* Case Studies in Thermal Engineering, 2021. **28**: p. 101561. DOI: doi.org/10.1016/j.csite.2021.101561.

[59] Koklu, M., R. Kursun, Y.S. Taspinar, and I. Cinar, *Classification of date fruits into genetic varieties using image analysis.* Mathematical Problems in Engineering, 2021. **2021**: p. 1-13. DOI: 10.1155/2021/4793293.

[60] Yasin, E.T., I.A. Ozkan, and M. Koklu, *Detection of fish freshness using artificial intelligence methods.* European Food Research and Technology, 2023. **249**(8): p. 1979-1990. DOI: 10.1007/s00217-023-04271-4.

[61] Taspinar, Y.S., I. Cinar, R. Kursun, and M. Koklu, *Monkeypox Skin Lesion Detection with Deep Learning Models and Development of Its Mobile Application.* Public health. **500**: p. 5.