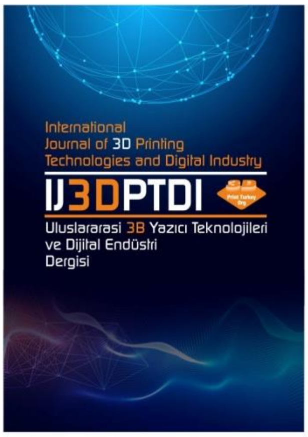

International
Journal of 3D Printing
Technologies and Digital Industry
IJ3DPTDI
Uluslararasi 3B Yazıcı Teknolojileri
ve Dijital Endüstri
Dergisi

ULUSLARARASI 3B YAZICI TEKNOLOJİLERİ
VE DİJİTAL ENDÜSTRİ DERGİSİ

INTERNATIONAL JOURNAL OF 3D PRINTING
TECHNOLOGIES AND DIGITAL INDUSTRY

ISSN:2602-3350 (Online)

URL: https://dergipark.org.tr/ij3dptdi

DISEASE DETECTION FROM CASSAVA LEAF IMAGES WITH DEEP LEARNING METHODS IN WEB ENVIRONMENT

Yazarlar (Authors): Sedat Metlek *

Bu makaleye şu şekilde atıfta bulunabilirsiniz (To cite to this article): *Metlek S., "Disease Detection From Cassava Leaf Images With Deep Learning Methods In Web Environment" Int. J. of 3D Printing Tech. Dig. Ind., 5(3): 625-644, (2021).*

DOI: 10.46519/ij3dptdi.1029357

Araştırma Makale/ Research Article

Erişim Linki: (To link to this article): https://dergipark.org.tr/en/pub/ij3dptdi/archive

DISEASE DETECTION FROM CASSAVA LEAF IMAGES WITH DEEP LEARNING METHODS IN WEB ENVIRONMENT

Sedat Metlek[a]*

[a] Department of Mechatronic, Vocational School of Technical Sciences, Burdur Mehmet Akif Ersoy University, Burdur, TURKEY

* *Corresponding Author: sedatmetlek@mehmetakif.edu.tr*

(**Received:** 28.11.2021; **Revised:** 10.12.2021; **Accepted:** 24.12.2021)

ABSTRACT

In this article, it is aimed to classify healthy and four different plant diseases from Cassava plant leaf images. For this purpose, the "Cassava-Leaf-Disease-Classification" data set, which is an up-to-date and difficult data set published in 2020, was used. The used data set includes a total of 21,397 images consisting of healthy and four different diseases. In the study, in the MATLAB environment, the images were first subjected to the Chan-Vese (CV) Segmentation method and the area of interest was determined. Features were extracted with the ResNet 50 and MobileNetV2 deep learning architectures from the detected areas. Extracted features are classified by Support Vector Machine and K-Nearest Neighbor algorithms.
The images are divided into two as training and testing according to the K-fold 5 value. The average highest success rates in training and test data were achieved by using the ResNet50 architecture and SVM classifier together, as a result of segmentation. As a result of training and testing processes, 85.4% and 84.4% success rates were obtained, respectively. At the end of the test process of the study, a trained network was obtained according to ResNet50, where the highest success rates were obtained, and MobileNetV2, another deep learning architecture used in the study. It has been compiled with MATLAB Builder NE in order to run these two networks in the web environment. Finally, it has been made available to manufacturers with a web-based embedded interface. Thus, a deep learning-based decision support system has been developed that can be easily used by all manufacturers in the web environment.

Key words: Deep learning, Convolution neural network, Image processing, Cassava leaf diseases.

1. INTRODUCTION

The world population is increasing every year by a quarter of the population of China [1]. As an inevitable result of this, there is an increasing demand in the food sector. In order to meet the increasing demand in these sectors, much more agricultural production is required. In addition, about 80 percent of agricultural production in the world is done by small-scale farmers [2]. More than half of the production is wasted due to diseases and other harmful organisms [3].

Along with this, there is a constant migration from rural areas where agricultural production is intense to urban centers where living standards are higher [1]. As a result of these migrations, it is predicted that approximately 70% of the world population will live in the city center until 2050 [4]. In the light of the information stated, it has become a necessity to implement smart agriculture practices in order to meet the increasing food demand [5]. With smart applications to be used during agricultural production, it is possible to be protected from many negative factors, from plant diseases to pests, from unconscious irrigation to pesticides [6]. If such applications are not used, approximately 35% of the product produced each year may be damaged [7]. This problem also causes great concern in the public and private sectors. In order to prevent these damages, it is essential to develop new applications urgently.

In this scope, it is important to monitor agricultural production in real time. There is a great demand for greenhouse applications, especially due to climate change and other environmental conditions. Although the outer cover is used to protect from negative environmental factors in greenhouse applications, it is insufficient in terms of protection from harmful insects or diseases. In this sense, when the studies in the literature are examined, it is seen that greenhouse automation systems have been developed with high technology devices. In addition, it is known that as a result of the developments in the software sector, deep learning architectures used in many applications have started to be used in greenhouse automations [8]. However, one of the important points here is that the producers who can use greenhouse automation technology should have a certain economic power. In addition, a very small part of agricultural activities are carried out in greenhouses. As a result, most of the agricultural production is carried out in open fields. For this reason, it is an important issue to use the technology that can be used in greenhouses in production in open areas. On the other hand, the majority of the producers who carry out agricultural activities in open areas consist of producers with low income levels [9].

Another important problem encountered during production in agricultural activities is the low number of trained specialist personnel. For this reason, limited expert personnel cannot serve all manufacturers at the same time. At this point, there are some studies on detecting disease states by monitoring agricultural lands from satellite with deep learning applications [10] [11] [12]. However, most of these studies are in the initial stages. In order to increase the widespread effect of the developed systems, it is of vital importance for agriculture and the economy that depends on agriculture that it appeals especially to low-income producers. It is a very important issue that especially deep learning-based applications that can be developed for agricultural production can run on lower hardware such as mobile devices with high accuracy. When the literature is examined, it is seen that the studies carried out are capable of working on devices with high equipment in a laboratory environment. As a result, the studies to be carried out should work on simpler devices and have a widespread effect. These reasons have been our main motivation source in the study.

The study was carried out in three basic steps. In the first step, we focused on Cassava, which is an important plant in terms of nutrition, and a data set about this plant species was provided. Consideration was given to the presence of more than one disease type in the provided data set. For this reason, the “cassava-leaf-disease-classification” data set, which is an up-to-date data set in the literature and also suitable for the study subject, was preferred. [13]. In the second step, transfer learning was performed on ResNet50 and MobileNetV2 architectures, which are current and popular deep learning architectures. A new architecture is proposed by making revisions in the classification layers of the specified architecture. In the last step, the tested architectures were compiled with MATLAB Builder NE to work in a server environment and converted to a web server environment. Thus, applications with high transaction costs are run on a single server, enabling thousands of manufacturers with low hardware to use the application over a server. With this application, producers can control the disease status of the cassava plant whenever they want. In this sense, a decision support system has been developed that can be used by expert personnel. General presentation of the study; In the second part, the literature related to the subject is mentioned. In Section 3, information about the data used in the study is given. In Section 4, the methods used are explained. In Section 5, the experimental study is presented. In Section 6, the performance analysis of the methods is compared. The final results obtained in Section 7 are presented.

2. RELATED WORKS

Continuous monitoring of the disease status of plants in the production environment by the producers is both economically and physically impossible. Therefore, it is extremely important to develop practical technologies for real-time disease monitoring that manufacturer can use. It is important to diagnose plant diseases early and classify the disease, especially using non-destructive methods. However, taking precautions against these diseases is a vital issue for producers. For this purpose, there are studies in the literature [14][15]. When it comes to non-destructive methods, the first method that comes to mind and is applied is image processing methods [6]. In this regard, there are methods developed to detect diseases caused by various factors in plants by using different imaging techniques [16].

As a result of the development of hardware technology in recent years, many different applications related to Machine and Deep Learning methods have been developed in embedded systems [17], [18]. Deep learning architectures are a structure consisting of many more layers [19] than classical artificial neural network methods such as Recurrent Neural Network, Radial Base Function Neural Network [20]. The features used in these architectures are extracted automatically, unlike the classical artificial neural network models [21] [22]. The mentioned feature extraction process is generally based on convolutional neural networks [23].

As a reflection of the development of hardware technology and deep learning architectures, applications using deep learning architectures have started to be developed in the agricultural sector as well. In recent years, there are some studies, albeit limited, on the application of deep learning methods to agriculture in the literature. Some of these are presented below;

Carranza-Rojas et al. aimed to classify plant species using herbarium images and deep learning models. As a result of their studies, they claim that they can classify plants more quickly and accurately with the deep learning model they developed instead of the classical plant classification taxonomy [24]. Lu et al. carried out a study on automatic identification of rice diseases. In his studies, he managed to classify 10 different rice diseases on 500 images using the CNN method [25]. Mohanty et al. trained a convolutional neural network to classify 14 different plants and 26 different disease types belonging to these plants. In his study, he used 54,306 images obtained under controlled environment conditions as data set. They have succeeded in classifying plant species and diseases with the deep learning architecture they have trained [2]. Zhang et al. carried out a study on fruit classification using image processing and convolutional neural networks. In their study, they used a dataset consisting of 3600 images from 18 different classes and 200 images in each class. They developed a 13-layer convolutional neural network on this dataset [26]. Steinbrener et al. classified hyperspectral fruit and vegetable images using convolutional neural networks. They stated that they used a dataset consisting of 2700 images in total containing 13 different fruits and vegetables in their studies [27]. Torres et al reviewed various CNN architectures and achievements such as AlexNet, VGG16 and GoogLeNet for fruit detection, classification, sequencing and quality control tasks [28].

As seen in the literature review, the majority of the studies are images taken under controlled conditions and applications run only in computer environments. These applications do not directly address the end user in the production environment. At this point, manufacturers need solutions that can instantly show if there are any problems with their products. When the studies in the literature are evaluated from a different perspective, it is seen that deep learning applications are made on computers with high hardware capacity. It is clear that this requires high cost and time. On the contrary, the budgets and time of those working in the production environment are limited. It is also known that the vast majority of them do not have any expertise in deep learning. It is extremely important for manufacturers to run the architectures designed to solve this problem, especially by transferring them to the server and web environment. Thus, manufacturers with low hardware technology will be able to directly access the results of transactions with extremely high transaction costs by using the internet on their mobile devices. Based on this focus, it is aimed to develop an application that can also work in the web environment. In line with this goal, training and testing processes of the deep learning algorithm were carried out by using images taken from the real production environment, instead of images taken under conditional conditions. The next step is made available to the manufacturers via the prepared interface. Thanks to this system, producers with low income will be able to have information about the disease status of the cassava plant by simply uploading the pictures of their products from any mobile device or computer to the prepared application.

3. MATERIAL

The success of the deep learning architectures used is directly related to the preferred datasets. Especially when specific fields are preferred as the subject of study, users have to create special datasets due to the lack of ready-made datasets related to the subject [29]. As a result of the development and spread of technology, many private datasets open to the public have been made available to researchers[30]. The "Cassava-Leaf-Disease-Classification (CLDC)" dataset preferred in the study is one of these datasets created in 2020 [13].

This dataset contains images of the cassava plant, the second largest carbohydrate source in Africa. It is an important agricultural plant that is generally preferred by farmers because it can withstand harsh conditions. At least 80% of their farms in Sub-Saharan Africa grow this plant. However, viral diseases may occur in a plant that is so preferred and may reduce plant yield. In particular, the presence of any disease in this plant greatly harms small farmers. For this reason, farmers seek the assistance of specialized personnel from the state to visually inspect and diagnose plants. From time to time, problems may arise in the financing and organization of all these transactions by the state. Every problem experienced is reflected as harm to the farmers. Such negativities in production, including industrial farmers, also disrupt the supply-demand balance, including in production planning.

The disease state developed in this plant has a structure that can be visually diagnosed by expert personnel. For this reason, with the help of developing technology, the same process can be performed with deep learning algorithms. The only problem here is that African farmers have low-end mobile devices. In this case, it overlapped with the main motivation values of the study and it was aimed to prevent this problem by running deep learning architectures in the server environment. Four different disease categories in the data set and the number of images belonging to these categories are presented in detail in Table 1, together with their class labels.

Table 1. Classes in the CLDC dataset

Label number	Label name	Count
0	Cassava Bacterial Blight (CBB)	1087
1	Cassava Brown Streak Disease (CBSD)	2189
2	Cassava Green Mottle (CGM)	2386
3	Cassava Mosaic Disease (CMD)	13158
4	Healthy (Ht)	2577

In Fig. 1, images of the leaves whose labels are presented in Table 1 are presented.

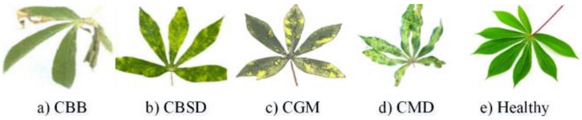


Figure 1. Classes of CLDC dataset

CBB, noted in Table 1, is a type of bacterial disease first reported in Brazil in 1912. This disease is caused by bacteria called "Xanthomonas axonopodis pv. Manihotis” (Xam). CBB, which starts with yellowing at the ends of the leaves, spreads towards the middle of the leaves over time. The disease has a long life cycle [31]. CBSD is a bacterial disease first reported in 1936 from northeastern Tanzania. Although the effect of this disease was limited to the East African coasts for many years, it is now seen in many parts of East and Central Africa [32]. CGM is a type of bacterial Cassava disease. It causes slight deformation in the form of yellow spots and mottled mosaic patterns on the leaf [33]. CMD is a common disease in Africa, India, and Sri Lanka [34,35]. It is usually transmitted by whiteflies. When intact leaves are infected with CMD, they often turn into twisted leaflets with a mosaic pattern. In Uganda, this disease and its derivatives cause an average of 82% root loss in Cassava plant [36]. As seen in Table 1, the data distribution among the classes is not equal. In particular, the number of data belonging to the CGM class is very high compared to other classes. On the other hand, the data belonging to the CBB class are very few compared to other classes. For this reason, images were revised in experimental studies.

4. METHOD

4.1. Segmentation Algorithm

Two types of images were used in the study. First, the images in the data set were used as raw, without any processing. Secondly, the background was removed from the images and only the relevant areas in the image were used. There are different segmentation algorithms to extract the relevant area from the image[23]. In the study, the Chan Vese (C-V) algorithm was used due to its success in the literature. For an image in this segmentation algorithm, the leveling function used in the φ domain is Eq. Presented in (1). The ϕ and C terms in the corresponding function represent the level adjustment parameter and the active curve in the image, respectively. x represents the pixels in the A image.

$$\begin{aligned} &C = \{x \in \varphi : (\phi) = 0\} \\ &inside\ (C) = \{x \in \varphi : (\phi) > 0\} \\ &outside(C) = \{x \in \varphi : (\phi) < 0\} \end{aligned} \tag{1}$$

Eq. (2) is used to set the leveling parameter to zero. Parameters μ, v, ∇, γ_1 and γ_2 provide the smoothness of level adjustment, iteration speed, slope, control of the inner and outer frame of the contour, respectively.

$$\frac{\vartheta\phi}{\vartheta t} = \delta(\phi)\left[\mu\nabla\left(\frac{\nabla\phi}{|\nabla\phi|}\right) - v - \gamma_1(I - c_1)^2 + \gamma_2(I - c_2)^2\right] \tag{2}$$

The content of c_1 used in Eq. (2) is presented in Eq. (3). The content of c_2 is presented in Eq. (4). The heaviside and dirac functions used in Eq. (3) and 4 are H(ϕ) and δ(ϕ), respectively.

$$C_1(\phi) = \frac{\int^{I}(x).H(\phi)dx}{\int^{H}(\phi)dx} \tag{3}$$

$$C_2(\phi) = \frac{\int^{I}(x).(1 - H(\phi))dx}{\int^{H}(1 - H(\phi))dx} \tag{4}$$

In general, the C-V segmentation method is used to segment the desired object in the image with the optimum initial contour. However, only the global segmentation feature is used in this model.

4.2. Deep Learning Algorithms

Recently, it is possible to see deep neural networks applications in many different fields from health to education [37–40]. There are different models such as trained and pre-trained in the literature [41–43]. It is frequently preferred in image processing applications, especially due to its success in feature extraction.

Since it is a method developed based on convolutional neural networks, it has a layered structure. The layers have different structures due to the connection within themselves and the filters used in the relevant layers. Although the filters and connections used are different, as a result, it contains the basic layers of convolutional neural networks. The basic layers in convolutional neural networks are presented in detail in Fig. 2.

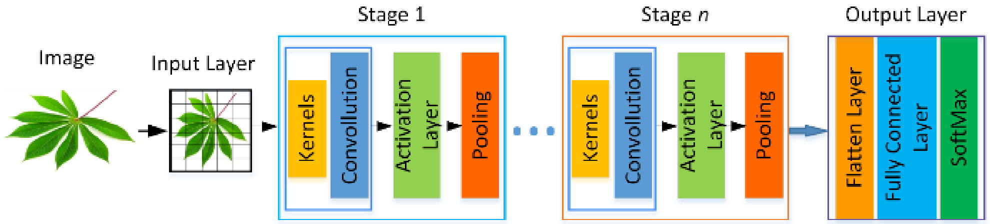


Figure 2. Structure of convolution neural network

Input Layer is the first layer of Convolution or deep learning algorithms. It is used to bring the data presented to the network to the size used by the network. In particular, images must be resized during the training phase of pre-trained deep learning algorithms. Kernels refer to *mxn*-sized *K* filters circulated over the two-dimensional *ixj* -sized *I* image (input matrix).

The user can use these filters as well as some ready-made filters in the literature. In the Convolution Layer, a new convolution matrix ($S_{i,j}$) is created by circulating the *K* kernels on the *I* matrix from the input layer. This matrix can be as many times as the user wants, as shown in Fig. 2. The operation performed in the convolution layer is shown in detail in Eq. (5).

$$S_{i,j} = (I * K)_{i,j} = \sum_{m}\sum_{n} I_{i,j}K_{i-m,j-n} \qquad (5)$$

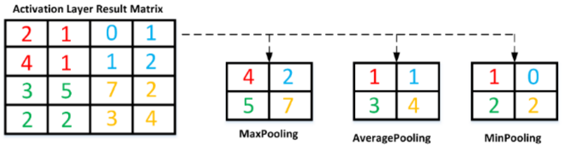
Activation Layer is the layer used to reduce the value ($x = S_{i,j}$) obtained from the convolution layer to certain intervals. In the literature, Rectified Linear Unit (ReLU), Sigmoid and Hyperbolic Tangent functions given in Eq. (6), (7) and (8) are used.

$$f(x) = max(0,x) = \begin{cases} 0 \;\; if \; x < 0 \\ x \; if \; x \geq 0 \end{cases} \qquad (6)$$

$$f(x) = \frac{1}{1+e^{-x}} \qquad (7)$$

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1 \qquad (8)$$

The Pooling Layer is the layer used to reduce the data from the Activation layer to smaller sized matrices. In this layer, a *2x2* matrix is circulated on the data coming from the Activation layer ($A_{i,j} = f(x_{i,j})$) as shown in Fig. 3. This filtering process is sometimes performed by skipping. This jumping process is called "stride". The maxpooling, averagepooling and minpooling functions presented in Eq. (9), (10) and (11) are used in the Pooling Layer, respectively. As a result, although data reduction in the Pooling layer makes the network work faster, it may cause data loss in some cases.


Activation Layer Result Matrix

2	1	0	1
4	1	1	2
3	5	7	2
2	2	3	4

MaxPooling

4	2
5	7

AveragePooling

1	1
3	4

MinPooling

1	0
2	2

Figure 3. Pooling layer functions

$$f_{max}(A) = max_{r,s}(A_{i,j}) \qquad (9)$$

$$f_{avg}(A) = \frac{1}{r+s}\sum_{i=1}^{r}\sum_{j=1}^{s}(A_{i,j}) \tag{10}$$

$$f_{min}(A) = min_{r,s}(A_{i,j}) \tag{11}$$

Output Layer can be created from Flatten Layer, Full Connected Layer and Classification Layer in itself. The Flatten Layer shown in Fig. 2 transforms the data from the previous layers into vectors in 1xD dimensions. The data converted to vector is multiplied by the relevant weight values in the full link layer and the bias value is added. The values obtained after this stage are sent to a classifier depending on the initiative of the user. In the literature, the softmax classifier given in Eq. (12) is generally used in this layer.

$$S(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}} \tag{12}$$

In Eq. (12), e^{y_i} represents the probability of the output of the model, and $\sum_{j=1}^{n} e^{y_j}$ represents the probability sum of all output values of the model.

4.2.1. ResNet 50

There are many deep learning architectures used in the literature today. One of them is the ResNet50 architecture developed by Microsoft. There are also different types, ResNet 18/50/101 and 152, according to the layer and block structures that make up the architecture [44,45]. Compared to classical convolution methods, the ResNet50 architecture provides a higher efficiency in the training phase [46]. Within the ResNet architecture, there are special structures called Residual Building Blocks (RBB). There are shortcut connections in RBBs that bypass layers [46]. With these shortcuts, it is aimed to solve the problem of getting stuck at the local minimum point, which is a general problem in flexible calculation methods [47]. For this purpose, convolutional layers with RBBs can be skipped as blocks [48]. In general, RBBs consist of convolution (Conv), batch normalization (Batch Normalization-BN), activation (ReLU) layers and a shortcut. There are also two different types of RBBs, RBB1 and RBB2. These structures are shown in Fig. 4. Shortcuts used in RBB1 structure are shown with x in Fig. 4. A non-linear function is used for the convolution layer in RBB1 and is denoted by F. The value obtained from RBB-1 is mathematically shown in Eq. (13). In the general structure of RBB-2 in Fig. 4, a non-linear function is used for the convolution layer, as in RBB1. But one more Conv-BN layer has been added. The shortcut in the structure of RBB-2 is denoted by H as shown in Fig. 4.

$$y = F(x) + x \tag{13}$$

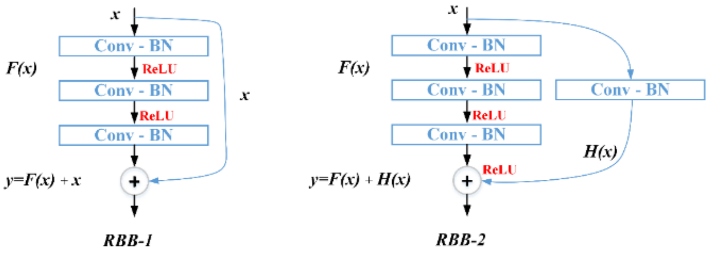
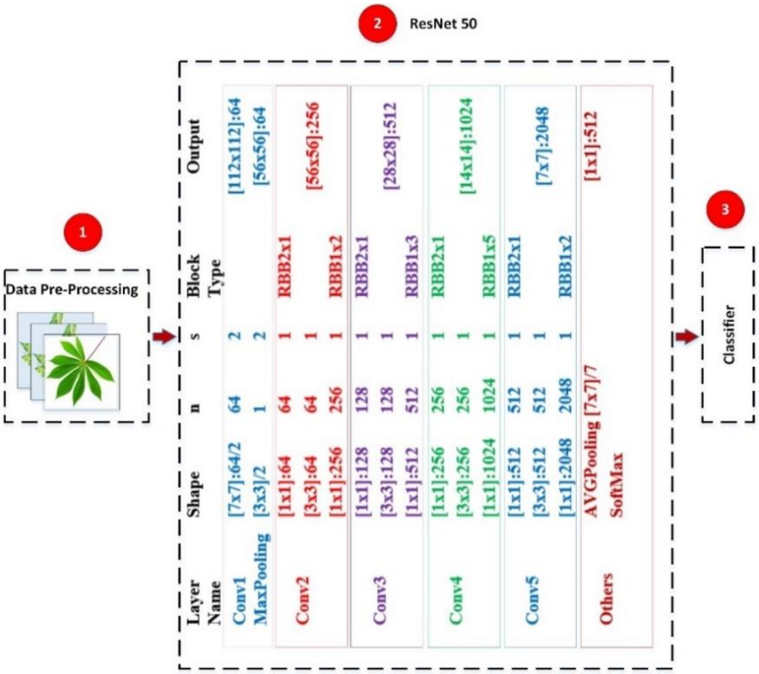


Figure 4. Structure of RBB-1 and RBB-2

$$y = F(x) + H(x) \tag{14}$$

The value obtained from RBB-2 is formulated with Eq. (14). In Fig. 5, the general architecture of ResNet50 is presented, n represents the number of repetitions, s represents the stride value.



Layer Name	Shape	n	s	Block Type	Output
Conv1	[7x7]:64/2	64	2		[112x112]:64
MaxPooling	[3x3]/2	1	2		[56x56]:64
Conv2	[1x1]:64	64	1	RBB2x1	[56x56]:256
	[3x3]:64	64	1		
	[1x1]:256	256	1	RBB1x2	
Conv3	[1x1]:128	128	1	RBB2x1	[28x28]:512
	[3x3]:128	128	1		
	[1x1]:512	512	1	RBB1x3	
Conv4	[1x1]:256	256	1	RBB2x1	[14x14]:1024
	[3x3]:256	256	1		
	[1x1]:1024	1024	1	RBB1x5	
Conv5	[1x1]:512	512	1	RBB2x1	[7x7]:2048
	[3x3]:512	512	1		
	[1x1]:2048	2048	1	RBB1x2	
Others	AVGPooling [7x7]/7				[1x1]:512
	SoftMax				

Figure 5. Structure of ResNet50

4.2.2. MobileNetV2

MobileNetV2 architecture, which reduces the network computing cost to a minimum by reducing the high network size, is a deep learning architecture developed by targeting mobile or lower cost devices [49,50]. It has applications in many fields from medicine to military in the literature [51,52]. The MobileNetV2 architecture has two important features that enable it to be implemented in mobile environments. The first of these is the bottleneck structure shown in Fig. 6. This structure is developed to solve nonlinearity problems in narrow layers of deep learning architectures. Secondly, it is the shortcut structure presented in Fig. 6. With this shortcut structure, data transfer can be provided between bottlenecks.

The Bottleneck structure used in the MobileNetV2 architecture is a kind of RBB that uses 1x1 convolution. Bottlenecks reduce the number of parameter and matrix multiplications thanks to this structure. As a result, it reaches higher depth with fewer parameters. For this reason, the MobileNetV2 architecture has a structure consisting of in-depth (dw) separable filters and combinations. Shortcut (also known as skip) connections, popularized by ResNet, are often used to connect non-Bottleneck layers. MobileNetV2 reverses this concept and connects Bottlenecks directly [48].

As shown in Fig. 6(a) in the created structure, it applies a convolution operation with a resolution of 1×1 pixel for each layer input. Deeply separable convolution filters examine the input information as two separate layers. This reduces both the speed and cost of the model. As shown in Fig. 6(a), the features obtained by separating them with filters are recombined in the merging step. The shortcuts shown in Fig. 6 may also experience some data loss. In Fig. 6, t refers to the expansion value applied to the input dimensions, n refers to the number of repetitions, c refers to the number of output channels, and s refers to the stride value. MobileNetV2 model uses batchnorm and ReLU activation function in its structure. In Batchnorm, data is rescaled based on the mean and standard deviation values. It is a technique that provides and supports a high learning rate. The ReLU activation function also contributes to success by transforming the model into a non-linear structure. The input size of MobileNetV2 has a resolution of 224 × 224 pixels. In the last layer of this model, the Softmax function is used as a classifier.

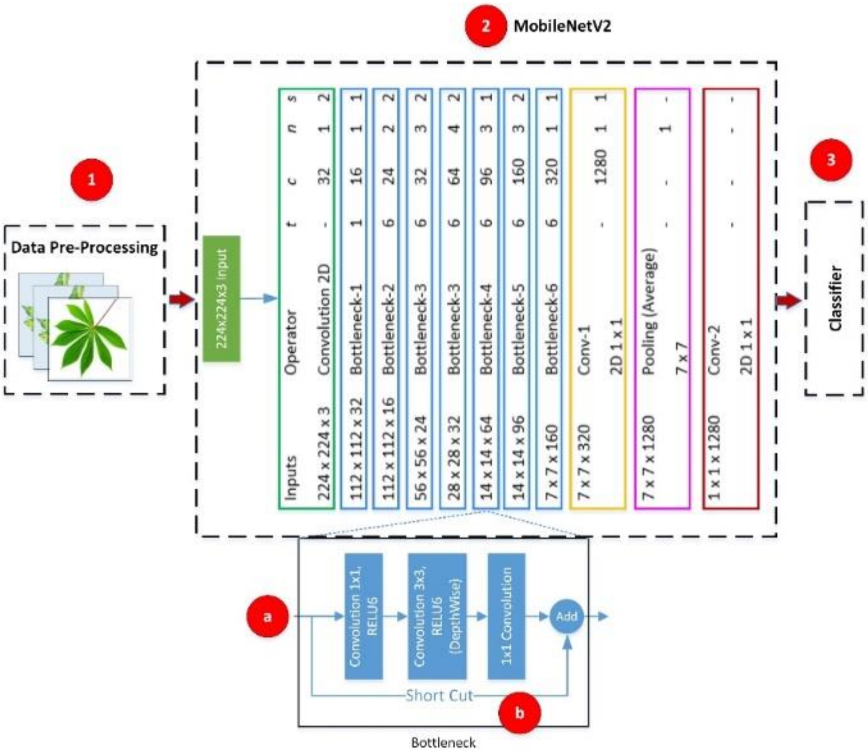

2 MobileNetV2
1
Data Pre-Processing
224x224x3 input
Inputs | Operator | t | c | n | s
224 x 224 x 3 | Convolution 2D | - | 32 | 1 | 2
112 x 112 x 32 | Bottleneck-1 | 1 | 16 | 1 | 1
112 x 112 x 16 | Bottleneck-2 | 6 | 24 | 2 | 2
56 x 56 x 24 | Bottleneck-3 | 6 | 32 | 3 | 2
28 x 28 x 32 | Bottleneck-3 | 6 | 64 | 4 | 2
14 x 14 x 64 | Bottleneck-4 | 6 | 96 | 3 | 1
14 x 14 x 96 | Bottleneck-5 | 6 | 160 | 3 | 2
7 x 7 x 160 | Bottleneck-6 | 6 | 320 | 1 | 1
7 x 7 x 320 | Conv-1 2D 1 x 1 | - | 1280 | 1 | 1
7 x 7 x 1280 | Pooling (Average) 7 x 7 | - | - | 1 | -
1 x 1 x 1280 | Conv-2 2D 1 x 1 | - | - | - | -
3
Classifier
a
Convolution 1x1, RELU6
Convolution 3x3, RELU6 (DepthWise)
1x1 Convolution
Add
Short Cut
b
Bottleneck

Figure 6. Structure of MobilNetV2

5. EXPERIMENTAL STUDIES

The study was carried out in three basic steps, which are data preprocessing, system design and web application presented below.

5.1. Data Pre-Processing

Algorithms were developed in MATLAB environment, on a workstation equipped with 2x Intel Xeon Gold 6130 (32 Core, 2.10GHz, 3.70GHz Turbo, 22MB - CPU), 64GB DDR4, Quadro P4000 256 Bit GDDR5 8GB, X99 Huananzhi Dual Processor DDR4 and 500GB SSD. The basic algorithm of the path followed in the data preprocessing phase is presented in Fig. 7.

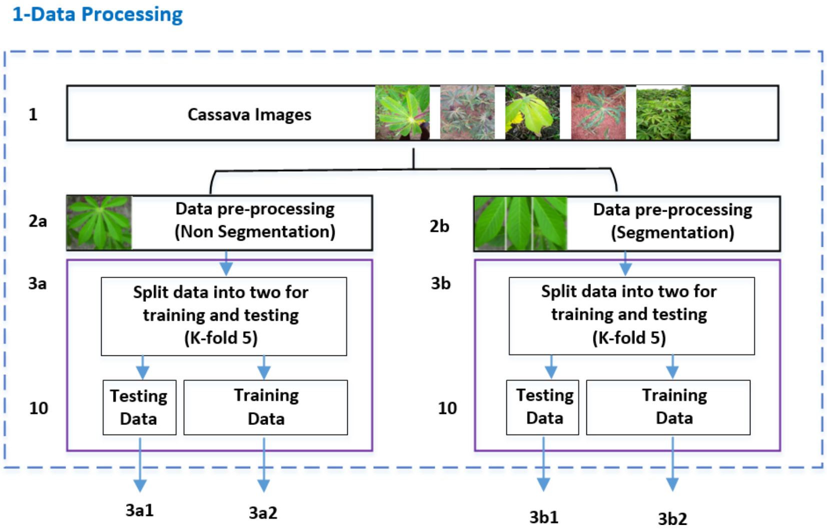

1-Data Processing
1
Cassava Images
2a
Data pre-processing (Non Segmentation)
2b
Data pre-processing (Segmentation)
3a
Split data into two for training and testing (K-fold 5)
3b
Split data into two for training and testing (K-fold 5)
10
Testing Data
Training Data
10
Testing Data
Training Data
3a1
3a2
3b1
3b2

Figure 7. Data Pre-Processing

In the preprocessing stage, the images in the CLDC dataset, which consists of five different classes, were first examined one by one. At this stage, some images that are not related to data classes were removed from the data set, as shown in Fig. 8. In addition, when the number of images in the classes given in Table 1 is examined, it is seen that the number of CBB data is very low, whereas the number of CMD data is much higher than that of other classes. For this reason, 1,000 images from each class were used in order to have a balanced distribution between the classes.

Figure 8. Irrelevant images

The data were subjected to the CV segmentation process described in Section 3.2 in 2b, as shown in Fig. 7. In Fig. 9, images before and after segmentation are presented.

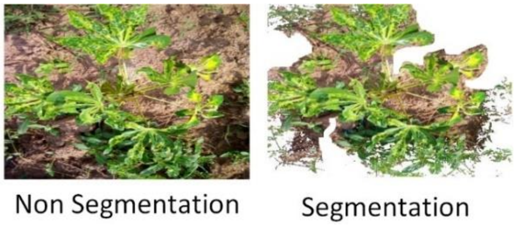


Figure 9. The difference of the segmentation process

5.2. Proposed System Design

As shown in step 4 in Fig. 10, it is aimed to train the system by sending both segmented and non-segmented images to the ResNet50 and MobilNetV2 algorithms separately. For this purpose, the training process of the system shown in step 6 was carried out with the hyperparameters shown in step 5. During the training phase, the training process of the system was ensured to continue until the minimum error value shown in step 7 was reached. If the system did not reach the desired error value, the hyperparameters used for the model were changed as shown in step 8.

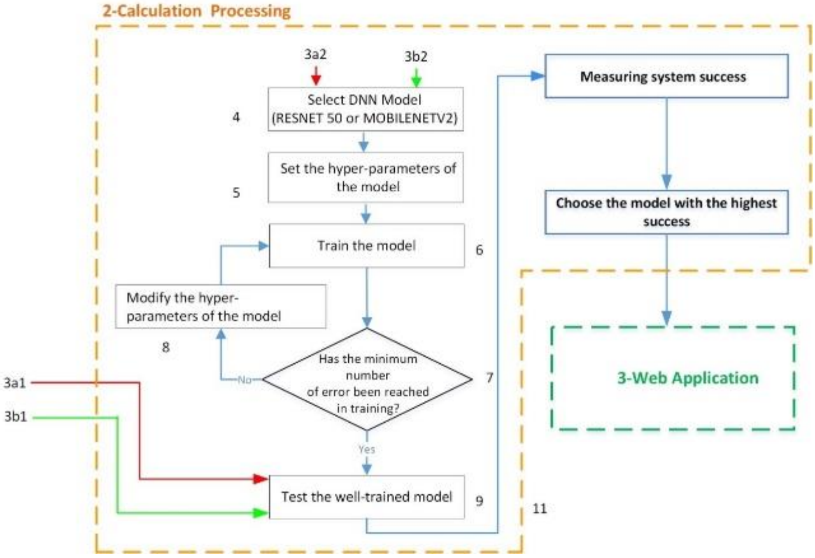

2-Calculation Processing
3a2
3b2
4
Select DNN Model (RESNET 50 or MOBILENETV2)
5
Set the hyper-parameters of the model
Train the model
6
Modify the hyper-parameters of the model
8
Has the minimum number of error been reached in training?
No
7
Yes
3a1
3b1
Test the well-trained model
9
11
Measuring system success
Choose the model with the highest success
3-Web Application

Figure 10. System design

Thus, each deep learning algorithm used was trained with two different image sets and four different models were prepared. In step 11, the success of the well-trained model obtained in step 9 was measured with the test data shown in step 10 in Fig. 7. Comparison matrices were used to measure the success of the system [53]. Accuracy Rate (AR), Recall (RC), Sensitivity (SS) ve F1 Score values obtained from the comparison matrix were used in the study. The success results of the system were evaluated and the architecture trained with the data set providing the highest accuracy rate of the ResNet50 and MobilNetV2 architectures was transferred to the web environment.

5.3. Proposed Web Application

All algorithms in the study were carried out in the MATLAB environment. The MATLAB platform provides support to WEB platforms as well as the opportunities it provides for algorithm development [54]. For this reason, the tested ResNet50 and MobilNetV2 architectures were transferred to the ASP.NET platform using the MATLAB Compile Runtime (MCR) library and MATLAB Builder NE.

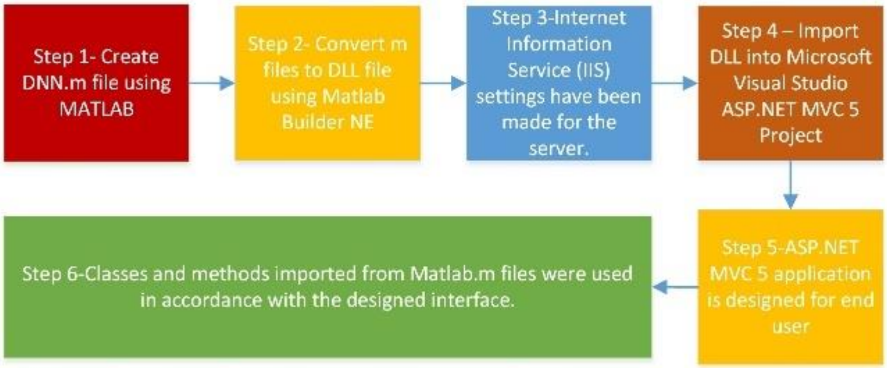
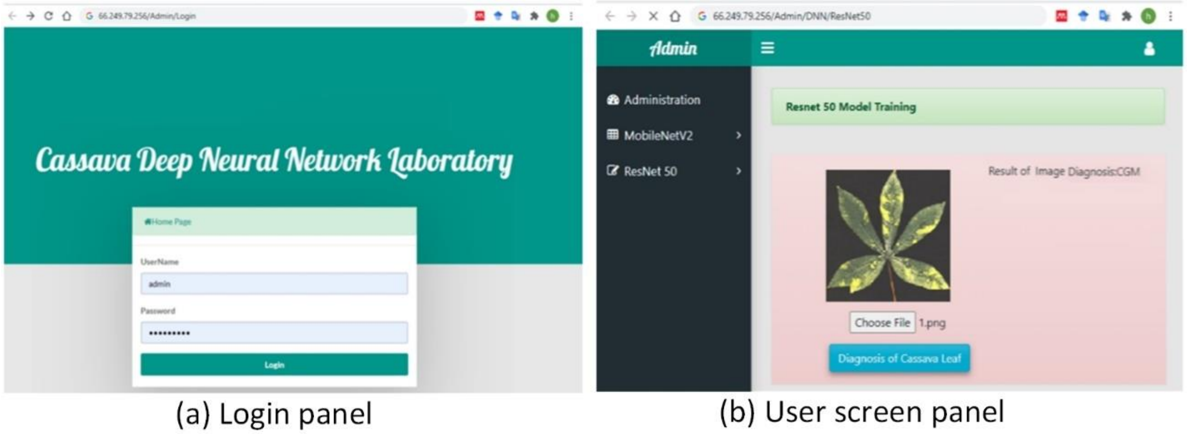

Step 1- Create DNN.m file using MATLAB
Step 2- Convert m files to DLL file using Matlab Builder NE
Step 3-Internet Information Service (IIS) settings have been made for the server.
Step 4 – Import DLL into Microsoft Visual Studio ASP.NET MVC 5 Project
Step 5-ASP.NET MVC 5 application is designed for end user
Step 6-Classes and methods imported from Matlab.m files were used in accordance with the designed interface.

Figure 11. Web platform migration steps

In the flow diagram shown in Fig. 11, m. files have been created. m files created using MATLAB Builder NE in step 2 were converted to DLL files. Since the application is aimed to be run on a server in the 3rd step, IIS (Internet Information Service) adjustments were made. DLL files are included in the ASP NET MVC 5 project developed in the Visual Studio environment in step 4. In the 5th step, ASP NET MVC 5 interface design was designed. In the last step, it was made available to the manufacturers. Fig. 12 shows the interface offered to users. With the designed interface, users can use the system with standard server services without the need for any other software. Thus, manufacturers will be able to access the system directly on the browser with the help of a computer or mobile device connected to the Internet.


66.249.79.256/Admin/Login

Cassava Deep Neural Network Laboratory

UserName: admin

Password: •••••••••

66.249.79.256/Admin/DNN/ResNet50

Resnet 50 Model Training

Result of Image Diagnosis:CGM

Choose File 1.png

(a) Login panel

(b) User screen panel

Figure 12. Designed system interface

In the use of the designed system, manufacturers log in using their user names and passwords through the menu presented in Fig. 12(a). After logging into the system, the menu in Fig. 12(b) is used. At the same time, deep learning architectures trained within the scope of the study can be preferred with the links in the left menu in Fig. 12(b). After clicking ResNet50, the user can determine the disease type of the product by uploading the Cassava images on his mobile device or computer to the system with the "Choose File" button.

6. RESULTS AND DISCUSSION

In this part of the study, the results obtained from the ResNet50 and MobileNet V2 architectures are discussed in detail. The features obtained from both architectures used were trained and tested with two different classifiers. The training and test performance of the models used were examined with Accuracy Rate, Recall, Sensitivity and F1-Score values. The training data given to the architectures used in the study and the complexity matrices according to the segmentation and classifier states are presented in Fig. 13-16.

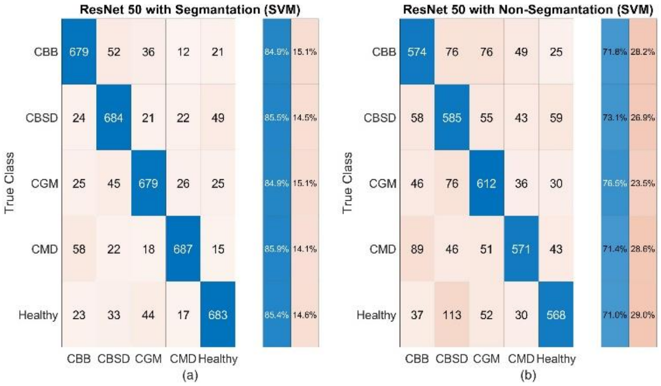


Figure 13. Results of the structure in which ResNet50 architecture and SVM are used together in the training process

The SVM classification results on the training images with and without segmentation with the ResNet50 architecture are shown in Fig. 13. In Fig. 13 (a), the highest success rate was obtained in the CBB class, while in Fig. 13 (b) it was obtained in the CGM class.

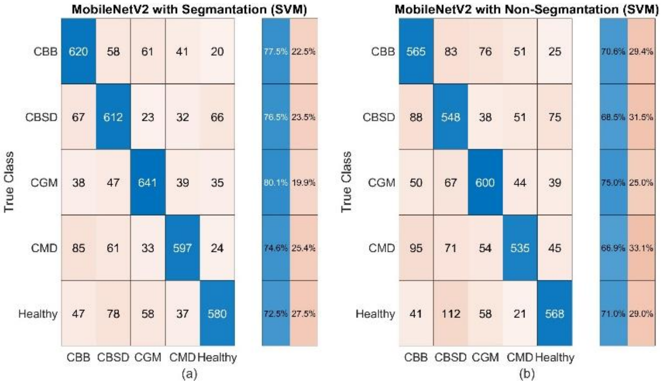


Figure 14. Results of the structure in which MobileNetV2 architecture and SVM are used together in the training process

In Fig. 14, SVM classification results on training images with and without segmentation with MobileNetV2 architecture are presented. In Fig. 14 (a) and (b), the highest achievement was obtained in the CGM class. Fig. 15 shows the difference between performing and not performing segmentation with the KNN classification algorithm on the training images with the ResNet50 architecture. In Fig. 15 (a) and (b), the highest success rate was obtained in the CGM class. In Fig. 16, the features obtained with the MobileNetV2 architecture on the training data are classified with the KNN algorithm. In Fig. 16 (a) and (b), the highest success rate was obtained in the CBB class.

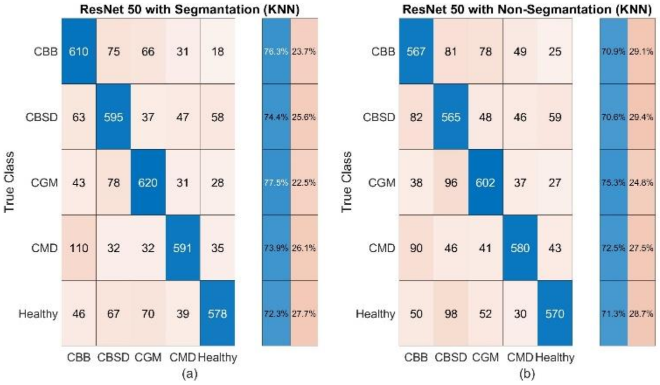


Figure 15. Results of the structure in which ResNet50 architecture and KNN are used together in the training process

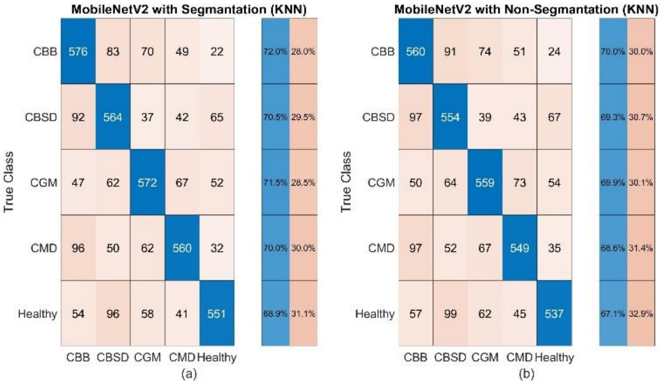


Figure 16. Results of the structure in which MobileNetV2 architecture and KNN are used together in the training process

The complexity matrix values obtained from the images separated as test images according to the K-fold 5 value are presented in detail in Fig. 17-20. The results obtained with ResNet50 with MobileNetV2 and SVM classifier are presented in Fig. 17. In Fig. 17 (a), the highest success rates were obtained in the CGM and Healthy classes at the same rate. In Fig. 17 (b), the highest achievement was obtained in the CDM class.

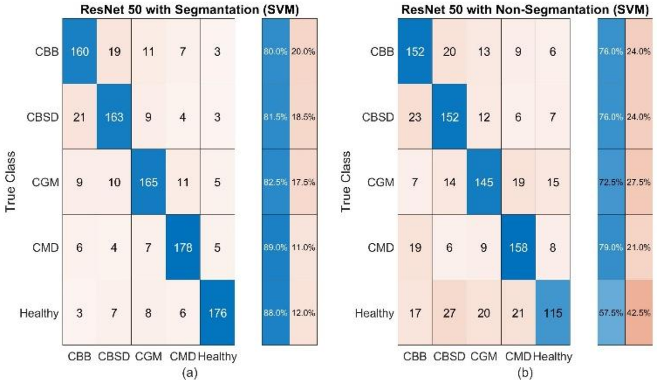
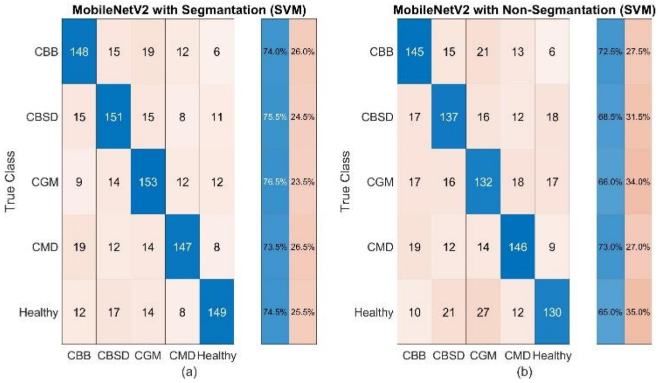


Figure 17. Results of the structure in which ResNet50 architectures and SVM are used together in the testing process

Confusion matrix of the results obtained with MobileNetV2 and SVM classifier on test images are presented in Fig. 18. In Fig. 18 (a), the highest classification success was obtained with CGM, while in Fig. 18 (b) it was obtained on CMD type.



MobileNetV2 with Segmantation (SVM)

True Class	CBB	CBSD	CGM	CMD	Healthy		
CBB	148	15	19	12	6	74.0%	26.0%
CBSD	15	151	15	8	11	75.5%	24.5%
CGM	9	14	153	12	12	76.5%	23.5%
CMD	19	12	14	147	8	73.5%	26.5%
Healthy	12	17	14	8	149	74.5%	25.5%

(a)

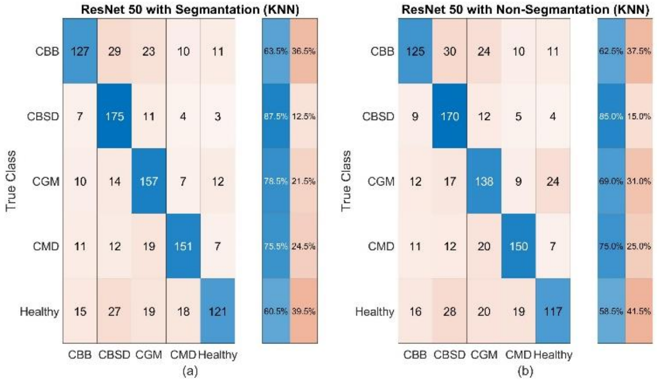
MobileNetV2 with Non-Segmantation (SVM)

True Class	CBB	CBSD	CGM	CMD	Healthy		
CBB	145	15	21	13	6	72.5%	27.5%
CBSD	17	137	16	12	18	68.5%	31.5%
CGM	17	16	132	18	17	66.0%	34.0%
CMD	19	12	14	146	9	73.0%	27.0%
Healthy	10	21	27	12	130	65.0%	35.0%

(b)

Figure 18. Results of the structure in which MobileNetV2 architecture and SVM are used together in the testing process

In Fig. 19, the results obtained with ResNet50 and KNN classifier are shown in confusion matrices. In Fig. 19(a) and (b), the highest achievement was achieved in the CBSD class. The results obtained with MobileNetV2 and KNN classifier in Fig. 20 on test images are presented in confusion matrices. In Fig. 20 (a) and (b), the highest achievement was obtained in the CBSD class, while the lowest achievement was obtained in the Healthy class.



ResNet 50 with Segmantation (KNN)

True Class	CBB	CBSD	CGM	CMD	Healthy		
CBB	127	29	23	10	11	63.5%	36.5%
CBSD	7	175	11	4	3	87.5%	12.5%
CGM	10	14	157	7	12	78.5%	21.5%
CMD	11	12	19	151	7	75.5%	24.5%
Healthy	15	27	19	18	121	60.5%	39.5%

(a)

ResNet 50 with Non-Segmantation (KNN)

True Class	CBB	CBSD	CGM	CMD	Healthy		
CBB	125	30	24	10	11	62.5%	37.5%
CBSD	9	170	12	5	4	85.0%	15.0%
CGM	12	17	138	9	24	69.0%	31.0%
CMD	11	12	20	150	7	75.0%	25.0%
Healthy	16	28	20	19	117	58.5%	41.5%

(b)

Figure 19. Results of the structure in which ResNet50 architecture and KNN are used together in the testing process

When the confusion matrices in Fig. (13-16), in which the performance results of the training data are presented, are examined, the highest and lowest success rates are generally close to each other in each segmentation situation. It is seen in confusion matrices that the success rate increases especially in cases where the segmentation process is applied. When the results of the study are examined in general, it can be foreseen that the success rate can be increased a little higher. Because in the data set used in the study and added to the literature in 2020, it is seen that some images that can be found in the same class have been placed in different classes by experts.

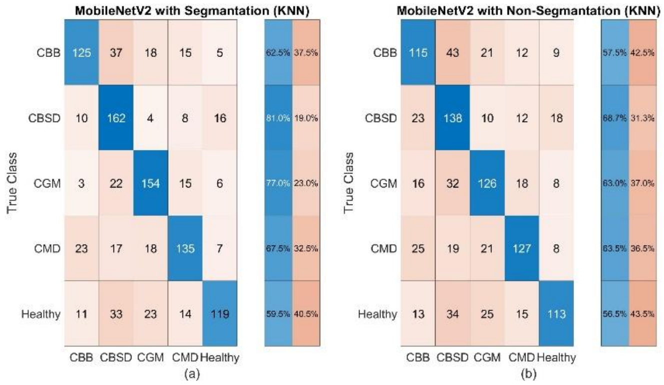


Figure 20. Results of the structure in which MobileNetV2 architecture and KNN are used together in the testing process

Secondly, the backgrounds of the images obtained while creating the data set are too mixed with the cassava plant in the image. In addition, a single cassava plant image is captured, while multiple cassava images are in a single image. In this case, the system has difficulty in distinguishing the images belonging to different classes among the specified images. Performance results obtained from Confusion Matrices are presented in detail in Table 2-5. AR, SS, RC, F1 metrics referred to in Section 5.2 are used in these tables.

Table 2. ResNet50 architecture with SVM training and test results

DNN Algorithm			ResNet50										
Pre-Processing			Segmentation					Non-Segmentation					
Classification			SVM					SVM					
Kind of diseases			CBB	CBSD	CGM	CMD	Healthy		CBB	CBSD	CGM	CMD	Healthy
Training (AR)	**85.4%**	AR	0.85	0.86	0.85	0.86	0.85	72.8%	0.72	0.73	0.77	0.71	0.71
		SS	0.84	0.82	0.85	0.90	0.86		0.71	0.65	0.72	0.78	0.78
		RC	0.84	0.82	0.85	0.90	0.86		0.71	0.65	0.72	0.78	0.78
		F1	0.84	0.84	0.85	0.88	0.86		0.72	0.69	0.74	0.75	0.74
Test (AR)	**84.4%**	AR	0.80	0.82	0.83	0.89	0.88	72.4%	0.76	0.76	0.73	0.79	0.58
		SS	0.80	0.80	0.82	0.86	0.92		0.70	0.69	0.73	0.74	0.76
		RC	0.80	0.80	0.83	0.86	0.92		0.70	0.69	0.73	0.74	0.76
		F1	0.80	0.81	0.83	0.88	0.90		0.73	0.73	0.73	0.77	0.66

Table 3. ResNet50 architecture with KNN training and test results

DNN Algorithm			ResNet50										
Pre-Processing			Segmentation					Non-Segmentation					
Classification			KNN					KNN					
Kind of diseases			CBB	CBSD	CGM	CMD	Healthy		CBB	CBSD	CGM	CMD	Healthy
Training (AR)	74.8%	AR	0.76	0.74	0.78	0.74	0.72	72.2%	0.71	0.71	0.75	0.73	0.71
		SS	0.70	0.70	0.75	0.80	0.81		0.69	0.64	0.73	0.78	0.79
		RC	0.70	0.70	0.75	0.80	0.81		0.69	0.64	0.73	0.78	0.79
		F1	0.73	0.72	0.76	0.77	0.76		0.70	0.67	0.74	0.75	0.75
Test (AR)	73.6%	AR	0.64	0.88	0.79	0.76	0.61	70.2%	0.63	0.85	0.69	0.75	0.59
		SS	0.75	0.68	0.69	0.79	0.79		0.72	0.66	0.64	0.78	0.72
		RC	0.75	0.68	0.69	0.79	0.79		0.72	0.66	0.64	0.78	0.72
		F1	0.69	0.77	0.73	0.77	0.68		0.67	0.74	0.67	0.76	0.64

Table 4. MobileNetV2 architecture with SVM training and test results

DNN Algorithm			MobileNetV2										
Pre-Processing			Segmentation					Non-Segmentation					
Classification			SVM					SVM					
Kind of diseases			CBB	CBSD	CGM	CMD	Healthy		CBB	CBSD	CGM	CMD	Healthy
Training (AR)	76.6%	AR	0.78	0.77	0.80	0.75	0.73	70.6%	0.71	0.69	0.75	0.67	0.71
		SS	0.72	0.71	0.79	0.80	0.80		0.67	0.62	0.73	0.76	0.76
		RC	0.72	0.71	0.79	0.80	0.80		0.67	0.62	0.73	0.76	0.76
		F1	0.75	0.74	0.79	0.77	0.76		0.69	0.65	0.74	0.71	0.73
Test (AR)	75.2%	AR	0.74	0.76	0.77	0.74	0.75	69.2%	0.73	0.69	0.66	0.73	0.65
		SS	0.73	0.72	0.71	0.79	0.80		0.70	0.68	0.63	0.73	0.72
		RC	0.73	0.72	0.71	0.79	0.80		0.70	0.68	0.63	0.73	0.72
		F1	0.73	0.74	0.74	0.76	0.77		0.71	0.68	0.64	0.73	0.68

Table 5. MobileNetV2 architecture with KNN training and test results

DNN Algorithm			MobileNetV2										
Pre-Processing			Segmentation					Non-Segmentation					
Classification			KNN					KNN					
Kind of diseases			CBB	CBSD	CGM	CMD	Healthy		CBB	CBSD	CGM	CMD	Healthy
Training (AR)	70.8%	AR	0.72	0.71	0.72	0.70	0.69	69%	0.70	0.69	0.70	0.69	0.67
		SS	0.67	0.66	0.72	0.74	0.76		0.65	0.64	0.70	0.72	0.75
		RC	0.67	0.66	0.72	0.74	0.76		0.65	0.64	0.70	0.72	0.75
		F1	0.69	0.68	0.72	0.72	0.72		0.67	0.67	0.70	0.70	0.71
Test (AR)	69.8%	AR	0.63	0.81	0.77	0.68	0.60	62.2%	0.58	0.69	0.63	0.64	0.57
		SS	0.73	0.60	0.71	0.72	0.78		0.60	0.52	0.62	0.69	0.72
		RC	0.73	0.60	0.71	0.72	0.78		0.60	0.52	0.62	0.69	0.72
		F1	0.67	0.69	0.74	0.70	0.67		0.59	0.59	0.63	0.66	0.63

In the literature, there are similar data sets published in 2017 and 2019 about the Cassava plant. Although there are much fewer images in these datasets than the dataset used in the study, the data classes are much more properly separated. In addition, some data sets consist of data obtained with high-resolution cameras [55]. Some have their backgrounds removed from the images. The data set used in the study, on the other hand, is a much newer data set, but mostly consists of images obtained by farmers. Therefore, there are problems with most of the data. In addition, due to the fact that it is a new data set, the number of studies using the same data set is very small. The study with this new data set in the literature is presented comparatively in Table 6.

Table 6. Comparison chart

No	Used Methods	Used Class Number	Success Rate	Ref.
1	Separable ConvolutionsUNet	2 – (Healthy, UnHealthy)	83.9 %	[56]
2	Proposed Model (Segmantation+ResNet50+SVM)	5- (CBB, CBSD, CGM, CMD, Healthy)	84.4%	

The study indicated in Table 6 distinguishes only two classes, healthy and unhealthy, instead of distinguishing five different classes. When the study is evaluated in terms of producers, it can already be visually distinguished whether the cassava plant is sick or not. The tricky part here is to distinguish between the five different classes.

7. CONCLUSION

Within the scope of this study, training and testing processes were carried out in 4 different experimental studies in order to classify Cassava images. In half of these experimental studies, C-V segmentation operation was performed on the images and the features obtained from two different deep learning architectures were applied on two different classification algorithms. When the classification algorithms used in the ResNet50 and MobileNetV2 architectures in Table 2-5 are evaluated within themselves, it is seen that the successful results are generally close to each other. On the other hand, when the classification algorithm results are compared with each other, the success rates in Table 7 are obtained.

Table 7. System performance results

	ResNet50				MobileNetV2			
	Segmentation		Non - Segmentation		Segmentation		Non - Segmentation	
	SVM	KNN	SVM	KNN	SVM	KNN	SVM	KNN
Training Average Accuracy	0.85	0.74	0.73	0.72	0.76	0.70	0.70	0.69
Test Average Accuracy	**0.84**	0.73	0.72	0.70	**0.75**	0.69	0.69	0.62

The web application was developed based on the ResNet50 and MobileNetV2 architectures, which give the best performance results in Table 7. Algorithms giving these performance results were compiled with MATLAB Builder NE and converted to API format. All functions in the specified algorithms have been made available in the web environment. The prepared algorithms were run in real-time with the help of MCR on a web server where the MATLAB program was not installed. As a result of these processes, farmers in the production sector will have access to the bootstrap-based web application, which is also prepared with low-equipped devices. By uploading Cassava leaf image to the system from any web-supported platform, it will be able to detect disease class information.

Thus, since the farmers will be informed about the diseases encountered during the production phase, they will be able to intervene without losing time. As a result, the loss of yield caused by diseases during production will be prevented. In future studies, it is aimed to recommend drugs that can be used directly for the disease by cooperating with the relevant institutions for the drug recommendation on the basis of the disease. In addition, it is aimed to increase the success rate of the study even more with data sets that will be prepared more carefully.

REFERENCES

1. Bahar, N. H. A., Lo, M., Sanjaya, M., Van Vianen, J., Alexander, P., Ickowitz, A., Sunderland, T., "Meeting the food security challenge for nine billion people in 2050: What impact on forests?", Global Environmental Change, Vol. 62, Pages 102056, 2020.

2. Mohanty, S. P., Hughes, D. P., Salathé, M., "Using Deep Learning for Image-Based Plant Disease Detection", Frontiers in Plant Science, Vol. 7, Pages 1419, 2016.

3. Harvey, C. A., Rakotobe, Z. L., Rao, N. S., Dave, R., Razafimahatratra, H., Rabarijohn, R. H., Rajaofara, H., MacKinnon, J. L., "Extreme vulnerability of smallholder farmers to agricultural risks and climate change in Madagascar", Philosophical Transactions of the Royal Society B: Biological Sciences, Vol. 369, Pages 20130089, 2014.

4. Savary, S., Ficke, A., Aubertot, J.-N., Hollier, C., "Crop losses due to diseases and their implications for global food production losses and food security", Food Security, Vol. 4, Pages 519–537, 2012.

5. Jiang, H., Li, X., Safara, F., "IoT-based Agriculture: Deep Learning in Detecting Apple Fruit Diseases", Microprocessors and Microsystems, , Pages 104321, 2021.

6. Han, Z., Xu, A., "Ecological evolution path of smart education platform based on deep learning and image detection", Microprocessors and Microsystems, Vol. 80, Pages 103343, 2021.

7. Pallagani, V., Khandelwal, V., Chandra, B., Udutalapally, V., Das, D., Mohanty, S. P., "dCrop: A Deep-Learning Based Framework for Accurate Prediction of Diseases of Crops in Smart Agriculture", In 2019 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), Pages 29–33, Rourkela, 2019

8. Zhou, C., Hu, J., Xu, Z., Yue, J., Ye, H., Yang, G., "A Novel Greenhouse-Based System for the Detection and Plumpness Assessment of Strawberry Using an Improved Deep Learning Technique", Frontiers in plant science, Vol. 11, Pages 559, 2020.

9. Chambers, R., Ghildyal, B. P., "Agricultural research for resource-poor farmers: The farmer-first-and-last model", Agricultural Administration, Vol. 20, Pages 1–30, 1985.

10. Kontoes, C., Wilkinson, G. G., Burrill, A., Goffredo, S., Mégier, J., "An experimental system for the integration of GIS data in knowledge-based image analysis for remote sensing of agriculture", International Journal of Geographical Information Systems, Vol. 7, Pages 247–262, 1993. http://doi:10.1080/02693799308901955.

11. Harris, R., "Remote sensing of agriculture change in Oman", International Journal of Remote Sensing, Vol. 24, Pages 4835–4852, 2003.

12. Sahoo, R. N., Ray, S. S., Manjunath, K. R., "Hyperspectral remote sensing of agriculture", Current Science, Vol. 108, Pages 848–859, 2015.

13. Makerere University, A. L., "Cassava Leaf Disease Classification", https://www.kaggle.com/c/cassava-leaf-disease-classification/overview/description, September 2, 2021.

14. Li, J., Tao, H., Shuhong, L., Salih, S. Q., Zain, J. M., Yankun, L., Vivekananda, G. N., Thanjaivadel, M., "Internet of things assisted condition-based support for smart manufacturing industry using learning technique", Computational Intelligence, Vol. 36, Pages 1737–1754, 2020.

15. López, M. M., Bertolini, E., Olmos, A., Caruso, P., Gorris, M. T., Llop, P., Penyalver, R., Cambra, M., "Innovative tools for detection of plant pathogenic viruses and bacteria", International Microbiology, Vol. 6, Pages 233–243, 2003.

16. Udendhran, R., Balamurugan, M., Suresh, A., Varatharajan, R., "Enhancing image processing architecture using deep learning for embedded vision systems", Microprocessors and Microsystems, Vol. 76, Pages 103094, 2020.

17. Gonzalez-Huitron, V., León-Borges, J. A., Rodriguez-Mata, A. E., Amabilis-Sosa, L. E., Ramírez-Pereda, B., Rodriguez, H., "Disease detection in tomato leaves via CNN with lightweight architectures implemented in Raspberry Pi 4", Computers and Electronics in Agriculture, Vol. 181, Pages 105951, 2021.

18. Ngo, T. N., Rustia, D. J. A., Yang, E.-C., Lin, T.-T., "Automated monitoring and analyses of honey bee pollen foraging behavior using a deep learning-based imaging system", Computers and Electronics in Agriculture, Vol. 187, Pages 106239, 2021.

19. Kayaalp, K., Metlek, S., "Classification of Robust and Rotten Apples by Deep Learning Algorithm", Sakarya University Journal of Computer and Information Sciences, Vol. 3, Pages 111–119, 2020.

20. LeCun, Y., Bengio, Y., Hinton, G., "Deep learning", Nature, Vol. 521, Pages 436–444, 2015.

21. Cetiner, I., Var, A. A., Cetiner, H., "Classification of Knot Defect Types Using Wavelets and KNN", Elektronika ir Elektrotechnika, Vol. 22, Pages 67–72, 2016.

22. Metlek, S., Kayaalp, K., "Detection of bee diseases with a hybrid deep learning method", Journal of the Faculty of Engineering and Architecture of Gazi University, Vol. 36, Pages 1715–1732, 2021.

23. Rastogi, S., Singh, J., "A systematic review on machine learning for fall detection system", Computational

Intelligence, Vol. 37, Pages 951–974, 2021.

24. Carranza-Rojas, J., Goeau, H., Bonnet, P., Mata-Montero, E., Joly, A., "Going deeper in the automated identification of Herbarium specimens", BMC Evolutionary Biology, Vol. 17, Pages 181, 2017.

25. Lu, Y., Yi, S., Zeng, N., Liu, Y., Zhang, Y., "Identification of rice diseases using deep convolutional neural networks", Neurocomputing, Vol. 267, Pages 378–384, 2017.

26. Zhang, Y.-D., Dong, Z., Chen, X., Jia, W., Du, S., Muhammad, K., Wang, S.-H., "Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation", Multimedia Tools and Applications, Vol. 78, Pages 3613–3632, 2019.

27. Steinbrener, J., Posch, K., Leitner, R., "Hyperspectral fruit and vegetable classification using convolutional neural networks", Computers and Electronics in Agriculture, Vol. 162, Pages 364–372, 2019.

28. Naranjo-Torres, J., Mora, M., Hernández-García, R., Barrientos, R. J., Fredes, C., Valenzuela, A., "A Review of Convolutional Neural Network Applied to Fruit Image Processing", Applied Science, Vol.10, Pages 1-31, 2020.

29. Zheng, Y.-Y., Kong, J.-L., Jin, X.-B., Wang, X.-Y., Zuo, M., "CropDeep: The Crop Vision Dataset for Deep-Learning-Based Classification and Detection in Precision Agriculture", Sensors, Vol. 19, Pages 1058, 2019.

30. Wegner, J. D., Branson, S., Hall, D., Schindler, K., Perona, P., "Cataloging Public Objects Using Aerial and Street-Level Images — Urban Trees", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6014–6023, Las Vegas, 2016.

31. Fanou, A., Valerien, Z., Wydra, K., "Cassava Bacterial Blight: A Devastating Disease of Cassava", Waisundara, V, Cassava, Pages 13-24, Erfurt, 2018

32. Tomlinson, K. R., Bailey, A. M., Alicai, T., Seal, S., Foster, G. D., "Cassava brown streak disease: historical timeline, current knowledge and future prospects", Molecular Plant Pathology, Vol. 19, Pages 1282–1294, 2018.

33. McCallum, E. J., Anjanappa, R. B., Gruissem, W., "Tackling agriculturally relevant diseases in the staple crop cassava (Manihot esculenta)", Current Opinion in Plant Biology, Vol. 38, Pages 50–58, 2017.

34. Legg, J. P., Lava Kumar, P., Makeshkumar, T., Tripathi, L., Ferguson, M., Kanju, E., Ntawuruhunga, P., Cuellar, W., "Cassava Virus Diseases", In Advances in virus research, Vol. 91, Pages 85–142, 2015.

35. Patil, B. L., Fauquet, C. M., "Cassava mosaic geminiviruses: actual knowledge and perspectives", Molecular plant pathology, Vol. 10, Pages 685–701, 2009.

36. Owor, B., Legg, J. P., Okao-Okuja, G., Obonyo, R., Ogenga-Latigo, M. W., "The effect of cassava mosaic geminiviruses on symptom severity, growth and root yield of a cassava mosaic virus disease-susceptible cultivar in Uganda", Annals of Applied Biology, Vol. 145, Pages 331–337, 2004.

37. Rajan, J. P., Rajan, S. E., Martis, R. J., Panigrahi, B. K., "Fog Computing Employed Computer Aided Cancer Classification System Using Deep Neural Network in Internet of Things Based Healthcare System", Journal of Medical Systems, Vol. 44, Pages 34, 2019.

38. Dawud, A. M., Yurtkan, K., Oztoprak, H., "Corrigendum to “Application of Deep Learning in Neuroradiology: Brain Haemorrhage Classification Using Transfer Learning”", Computational Intelligence and Neuroscience, Vol. 2020, Pages 1–1, 2020.

39. Metlek, S., Kılınç, E. E., Determination of Heart Disease By Machine Learning Methods. In 5th International Gap Mathematics-Engineering-Science and Health Sciences Congress; pp. 48–74, Urfa, 2020.

40. Mengi, D. F., Metlek, S., "Modeling Belongs To Turkey’s Mediterranean Coast Wind Of Exergy Multılayer Neural Network", International Journal of Engineering and Innovative Research, Vol. 2, Pages 102–120, 2020.

41. Metlek, S., Kayaalp, K., Basyigit, I. B., Genc, A., Dogan, H., "The dielectric properties prediction of the vegetation depending on the moisture content using the deep neural network model", International Journal of RF and Microwave Computer-Aided Engineering, Vol. 31, Pages e22496, 2021.

42. Basyigit, I. B., Doğan, H., "The analytical and artificial intelligence methods to investigate the effects of aperture dimension ratio on electrical shielding effectiveness", International Journal of Electronics and Telecommunications, Vol. 65, Pages 359–365, 2019.

43. Basyigit, I. B., Genc, A., Dogan, H., Senel, F. A., Helhel, S., "Deep learning for both broadband prediction of the radiated emission from heatsinks and heatsink optimization", Engineering Science and Technology, an International Journal, Vol. 24, Pages 706–714, 2021.

44. Rajpal, S., Lakhyani, N., Singh, A. K., Kohli, R., Kumar, N., "Using handpicked features in conjunction with ResNet-50 for improved detection of COVID-19 from chest X-ray images", Chaos, Solitons & Fractals, Vol. 145, Pages 110749, 2021.

45. Narin, A., Kaya, C., Pamuk, Z., "Automatic detection of coronavirus disease (COVID-19) using X-ray images and deep convolutional neural networks", Pattern Analysis and Applications, Vol. 24, Pages 1207–1220, 2021.

46. Liu, T., Chen, M., Zhou, M., Du, S. S., Zhou, E., Zhao, T., "Towards understanding the importance of shortcut connections in residual networks", Computer Science, Vol. 1, Pages 1–27, 2019.

47. Kensert, A., Harrison, P. J., Spjuth, O., "Transfer Learning with Deep Convolutional Neural Networks for Classifying Cellular Morphological Changes", Slas Dıscovery: Advancing the Science of Drug Discovery, Vol. 24, Pages 466–475, 2019.

48. He, K., Zhang, X., Ren, S., Sun, J., "Deep Residual Learning for Image Recognition", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, Las Vegas, 2016.

49. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C., "MobileNetV2: Inverted Residuals and Linear Bottlenecks", IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4510–4520, Salt Lake City, 2018.

50. Souid, A., Sakli, N., Sakli, H., "Classification and Predictions of Lung Diseases from Chest X-rays Using MobileNet V2", Applied Sciences, Vol. 11, Pages 2751, 2021.

51. Tsai, C., Lai, Y., Perng, J., Tsui, I., Chung, Y., "Design and Application of an Autonomous Surface Vehicle with an AI-Based Sensing Capability" IEEE Underwater Technology (UT), pp. 1–4, Kaohsiung, 2019.

52. Toğaçar, M., Cömert, Z., Ergen, B., "Intelligent skin cancer detection applying autoencoder, MobileNetV2 and spiking neural networks", Chaos, Solitons & Fractals, Vol. 144, Pages 110714, 2021.

53. Chicco, D., Tötsch, N., Jurman, G., "The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation", BioData Mining, Vol. 14, Pages 13, 2021.

54. Kuşcu, Ö., Çetiner, H., Çetin, Ö., "Development of a web interface for performing morphological operations on CUDA platform", Computer Applications in Engineering Education, Vol. 24, Pages 787–798, 2016.

55. Mwebaze, E., Gebru, T., Frome, A., Nsumba, S., Tusubira, J., "iCassava 2019 fine-grained visual categorization challenge", Computer Science, Vol. 1, Pages 28–47, 2019.

56. Rao, P. K., "Cassava Leaf Disease Classification using Separable Convolutions UNet", Turkish Journal of Computer and Mathematics Education (TURCOMAT), Vol. 12, Pages 140–145, 2021.