Deep Convolution Neural Network to solve Problems for Appel Leaf Disease Detection

Sutriawan[*, a,1], Ahmad Zaniul Fanani[b,2], Farrikh Alzami[c,3], Ruri Suko Basuki[d,4]

[a] Departmen of Computer Science, Universitas Muhammadiyah Bima, Bima, Indonesia
[a,b,c,d,] Departmen of Computer Science, Dian Nuswantoro University, Semarang, Indonesia
[1] sutriawan1612@gmail.com*, [2] a.zainul.fanani@dsn.dinus.ac.id; [3] Alzami@dsn.dinus.ac.id; [3] ruri.basuki@dsn.dinus.ac.id

Abstract

Orchardists are now concerned about apple leaf infections since they could result in crop failure. It is challenging for growers to identify the type of illness on apple leaves due to the large variety of diseases that can affect apple leaves. In this study, we cover four different illness categories: healthy, numerous diseased, rusty, and scabby. a deep convolutional neural network method of processing. using a number of suggested methods, including data preprocessing and the pre-configured VGG-16 deep convolutional neural network (CNN) architecture. The Adam optimization model's beta 2 = 0.999 parameter value at Ephoch to 85/100 with an accuracy of 0.7582 and epsilon = 1e-07 parameter value at Ephoch to 32/100 with an accuracy of 0.7582 both produced the best accuracy outcomes.

Keywords: Deep Learning, Convolution Neural Network, VGG16, Adam, SGD

I. INTRODUCTION

For the past few years, apple leaf diseases have caused anxiety among orchardists. 4-6 weeks after the removal of unproductive twigs and leaves, the disease's symptoms start to show on the leaves. Beginning with the oldest leaves and progressing to the youngest leaves, the leaves initially develop erratic white patches that are brown in hue with black dots on the upper surface, lasting until the entire leaf collapses. Conventional diagnosis techniques, which include manually checking the trees, are costly, ineffective, and challenging. In order to address this issue, there is an increasing demand for an automated system that can readily identify illnesses at an early stage, safeguard crops from production failure, and generate higher-quality crops [1]. Deep learning models are currently the subject of numerous studies looking at disease detection in fruit and vegetable crops. Apple leaf veins are intricate and diverse, making it challenging to identify illnesses with comparable values [2]. Since almost all apple illnesses have an impact on the leaves, they are a crucial and reliable source of information for disease identification [3]. However, depending entirely on these sources does not promote speedy and effective disease identification and may also lead to other issues connected to subjectivity. subjective. Monitoring the health of apples is a time-consuming process, but rapid automatic detection of illnesses during production allows for far more thorough and accurate monitoring, and it also aids orchardists in making more informed disease assessments. It is crucial to implement timely disease prevention and control measures to prevent widespread disease in order to ensure healthy apple growth and maximize returns [4]. Figure 1 shows an example of apple leaf data with a texture-based disease category that can be recognized based on the shape, color, and type of illness present on the apple leaf.

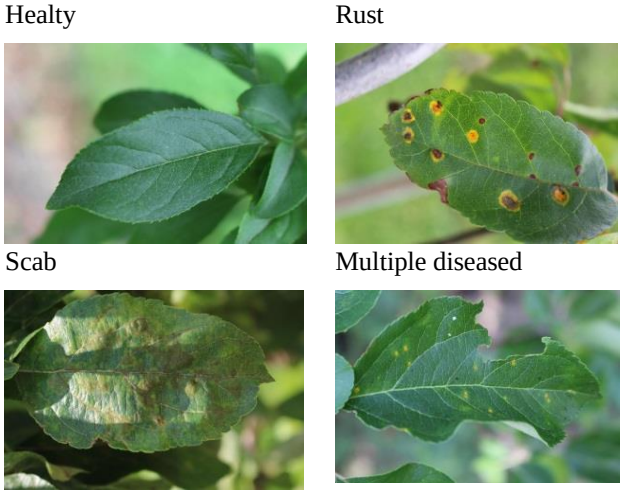

Figure 1. Examples of variations of apple leaf diseases

Diagnosing plant diseases can be challenging. Diseases like scab, black rot, and cedar rust are especially perilous because they rapidly spread to other parts of the leaf and ultimately kill the plant if they go unchecked [5]. This is a serious problem when trying to diagnose many different diseases, and it highlights the need of basing diagnosis on disease appearance.

Adaptive modes of development for deep learning that are specifically designed to identify diseases in plant leaves, such as The use of color segmentation in conjunction with a modified discrete wavelet transform for the diagnosis of illnesses in apple leaf tissue Apple leaf disease detection based on improved convolutional neural networks (CNN) artificial neural networks Because of the complexity and variation of apple leaf veins, as well as the difficulty in determining which diseases are similar to one another, a detection model with a new target of apple leaf diseases was developed by employing the DF-Tiny-YOLO technique and a deep learning approach. This model was successful in identifying the diseases [1,2]. This research makes use of three different types of prediction models, namely CNN, SVM, and KNN, in order to analyze photographs of apple plant leaves and categorize them as either healthy or diseased. In addition, a number of image pre-processing techniques are utilized in order to enhance image quality and get rid of any noise that might have an impact on the classification results [5]. Apple Leaf Convolutional Disease Modeled Using a Neural Convolutional Network Identification uses CNN in conjunction with the AlexNet model to provide a solution to the problem of illnesses that are damaging the leaves of apple trees. Convolution is used in the model to extract coarse features, which helps to keep a large receptive field while simultaneously lowering the number of parameters [6].

Throughout the course of this investigation, we covered four distinct forms of illness: healthy, multiple diseased, rusty, and scabby. The proposed method can be broken down into three distinct steps: identifying diseased areas, categorizing those areas, and performing analyses. The following are the primary contributions made by this research:

- Incorrect is the practice of separating the channels and then equalizing each channel independently. The intensity levels of the image are what are being adjusted by equalization, not the color components. Hence, histogram equalization cannot be immediately applied to an image that just has the RGB color model. It must be applied in such a way that the intensity levels are balanced without throwing off the image's color harmony in order for it to be effective. So, the first thing that needs to be done is to change the color space of the image from RGB to one of the color spaces that distinguishes between the intensity values and the color components.
- By establishing a threshold, a value is chosen at random to serve as the threshold. The Otsu method, on the other hand, does not require the user to select a value; rather, it calculates the value automatically.
- After the images of the equalized histogram have been obtained from the stage before this one, the Otsu binarization segmentation method should be used to them.

This study is broken up into several discussion chapters, the first of which is titled "Introduction" and explains the historical context as well as the research challenges associated with disease detection in apple leaves. In Chapter II, "Related Work," we discuss the related research as well as the findings of previous research. In Chapter III: Proposed Method, an overview of the source and processing of datasets is provided, as well as a method that is proposed for the identification of apple leaf diseases. explains in depth the formulation of the model as well as any improvements made to it before proceeding to evaluate its performance. The results of the experiments as well as an assessment of how well the proposed model works are detailed in Chapter IV "Results and Analysis" section. In Chapter IV, "Conclusion," you will find a summary of the findings of the research, as well as some suggestions for the development of future research.

II. PROPOSED METHODE

A. Dataset dan Data Augmentation

The Apple Leaf Public dataset, which was downloaded from an open-access repository, is what is utilized here for the analysis. This dataset contains 1821 pictures of Apple leaves for training purposes, and another 1821 pictures for testing purposes.

Train_1 Train_2 Train_3

Train_4 Train_5 Train_6

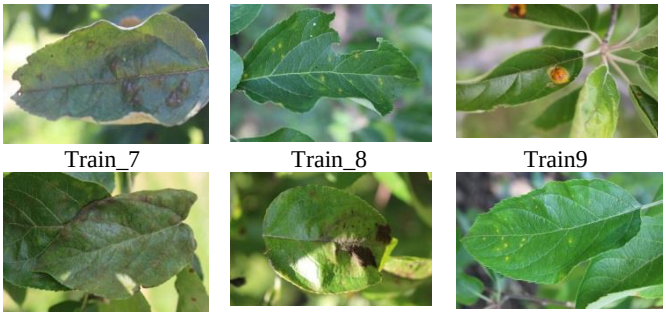

Figure 2. Example of Apple Leaf Disease Traing Data.

In the training set, we can see four different illness categories (health, numerous diseases, rust, and scab) as depicted in Figure 3.

	image_id	healthy	multiple_diseases	rust	scab
0	Train_0	0	0	0	1
1	Train_1	0	1	0	0
2	Train_2	1	0	0	0
3	Train_3	0	0	1	0
4	Train_4	1	0	0	0
...	...	...	...	...	...
1816	Train_1816	0	0	0	1
1817	Train_1817	1	0	0	0
1818	Train_1818	1	0	0	0
1819	Train_1819	0	0	1	0
1820	Train_1820	0	0	0	1

Gambar 3. Training data class description

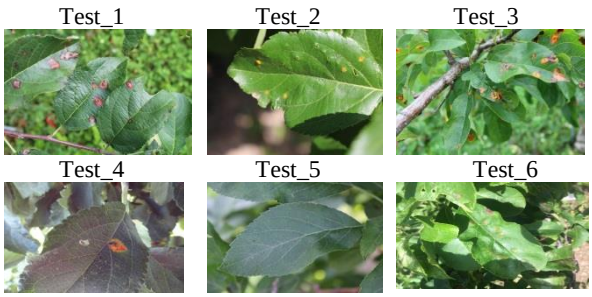

Figure 4. Example of Apple Leaf Disease Testing Data.

Figure 5. Percentage Chart of Apple Leaf Disease Classes

Gambar 3, yang mengilustrasikan persentase setiap kelas yang termasuk dalam kumpulan data, mengungkapkan informasi berikut:

Table 1. Data Class Description

Healty	516 gambar
Multi Diseased	91 gambar
Rust	622 gambar
Scab	592 gambar

From the training and testing data, each of which has a total of 1821 total images of apple leaves with a total of four disease classes, each image of a leaf has several different types of diseases, including healthy, multi-diseased, rust, and scab. The total number of images of apple leaves in the training and testing data sets is 3621. There are 516 samples in the Healthy class, 91 samples in the multi-diseases class, 622 samples of rust, and 592 samples of scab. This results in the dataset having an uneven distribution of values. An unbalanced dataset has the potential to produce skewed and biased classification results.

B. Preprocessing

Some of the preprocessing methods that were utilized in this research to retouch images of apple leaves are depicted in Figure 6.

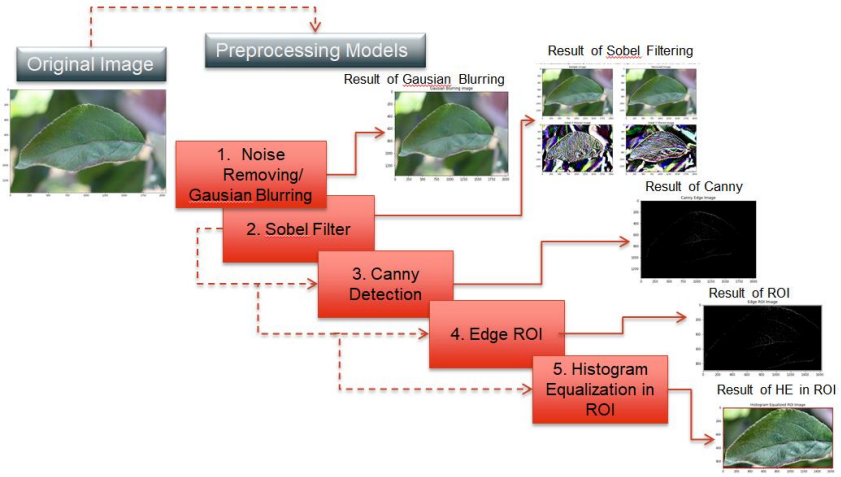

Figure 6. Proposed Preprocessing Technique

- Gausian Blurring

  Many image smoothing techniques such as Gaussian Blurring, Median Blurring, etc., are to some extent good at removing a small amount of noise. In those techniques, we take a small neighborhood around a pixel and perform some operations such as gaussian weighted average, median value, etc. to replace the central element.
- Sobel Filtering.

  A discrete first-order difference operator, the Sobel operator is named after its namesake. The brightness function of the image can be calculated with the help of this operator. It is necessary to perform the calculation of the convolution of the image matrix with the horizontal and vertical direction templates Gx and Gy. As a result, we are able to obtain a rough approximation of the brightness difference in the form of a gradient in both the X and Y directions [7].

  The traditional method of sobel edge detection involves performing an operation on a 3x3 pixel area using the corresponding template convolution, inferring the gradient value of the center pixel, and then comparing that value to the preset threshold gradient value in order to determine the pixel of the image that corresponds to the edge of the image if it is greater than the threshold criterion [8].
- Canny Detection

  The use of Gaussian filtering is typically the initial step in any intelligent edge identification technique. By smoothing out the image, the noise can be removed using a technique called Gaussian filtering. After that comes the calculation of the intensity gradient. Following the execution of the non-maximum suppression method comes the application of the double thresholding process. This guarantees that the detection is

limited to noticeable edges only. When dealing with photos that contain a lot of noise, the Canny operation delivers good results in terms of performance [7].

The implementation of the Canny Edge detection algorithm can be broken down into five distinct steps, which are as follows:

1. Smoothing: An operation known as smoothing or blurring is performed on the image in order to remove noise from it.
2. Locating the gradient Once the gradient of the image has been located, the edges of the image should only be marked in regions that have a magnitude that is relatively high.
3. Suppression of non-maximum values Only values that are at their local maximum should be considered edges.
4. Double thresholding: This technique is used to determine where prospective edges lie.
5. Edge tracking with hysteresis: Once all edges that are not connected to a very definite or strong edge have been suppressed, the remaining edge will be taken into consideration as the final edge [9].

C. Deep Learning Modeling (DLM)

- A Comparison characteristics of Apple Leaf Disease:

Tabel 2. Characteristic of Appel leaf Diseased

Diseased Type	Characteristic
Healty	They have a shape that is either elliptical or oval, and the leaf edges are serrated in a blunt manner. There are no spots or rust on either side of the leaf because it is still in its juvenile stage.
Multi diseasesd	leaf illnesses include red patches, empty leaves, and aberrant forms are common.
Rust	The oily spots are initially little and orange-red in color, but as they grow larger, their appearance changes to become spherical, orange-yellow, and red-edged.
Scab	Disease hotspots tend to be roughly circular or radial. The fungus begins as a greenish brown on the leaves and eventually turns black. Some disease spots can spread across an entire leaf, giving it a scorched appearance.

- Model training

Table 3, Comparison of Optimizers type

Optimizers Type	Parameter Name	Parameter Value
Adam	learning_rate	0.001
	beta_1	0.9
	beta_2	0.999
	epsilon	1e-07
	amsgrad	False
SGD	learning_rate	0.01
	momentum	0.0
	nesterov	False

When the training process is carried out by comparing the performance of two different optimizer models using several parameters, each parameter is set to its default value as shown in Table 3, and the Adam Optimizer and the SGD Optimizer are the two types of optimizers that are utilized in the training process.

D. Deep Convolution Neural Network

Deep learning is a subfield of machine learning that simulates human thought processes by processing information in a way similar to how humans would. Processing information and understanding human discourse

are both accomplished through the use of layers in deep learning. The structure-property connection of essential engineering alloys can be modeled using a classification technique called deep neural learning, which consists of four different layers. Figure 7 provides a visual representation of the structure of deep neural learning.

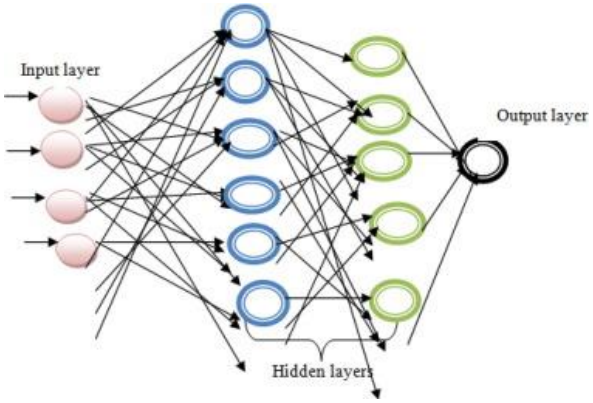

Figure 7. Illustration of deep learning structure

Figure 7 depicts the construction of a deep neural learning classifier for effective structure-property analysis. Input and output layers, as well as one or more hidden layers, are just some of the layers that make up a neural network. Neurons, which function as activating nodes, are fully connected from one layer to the next in order to create a network at each level. Amount of data gathered by the input layer at time t, represented as Id(t), is gathered at this moment. The neurons are then linked from one layer to the next by means of dynamic weights, which can be determined in the following way:

$$Input(t) = \sum_{i=1}^{n} data_i * weight_i + bias \quad (1)$$

III. DESING EXPERIMENT RESULT AND ANALYSIS

Training and testing on the apple leaf dataset allows us to assess the efficacy of our proposed method. Figure 8 depicts the proposed model.

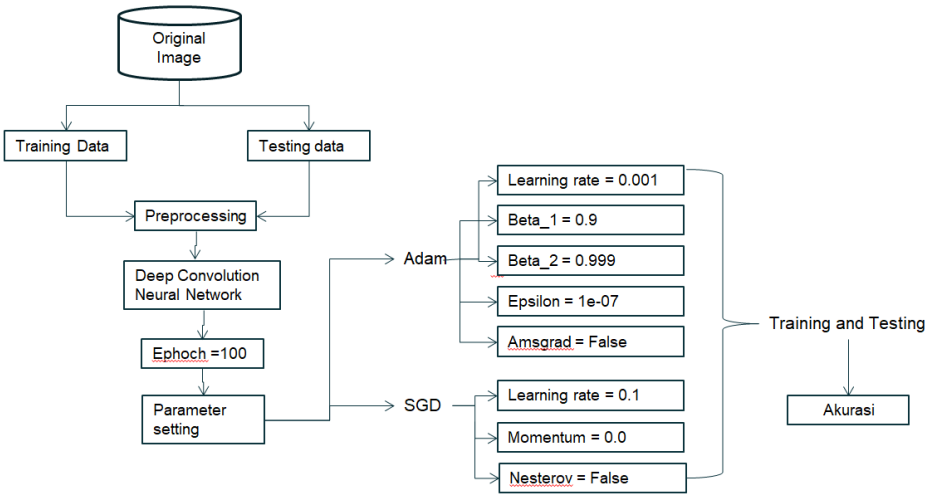

Figure 8. Proposed Model

All of the results of the experiments were obtained by dividing the total time spent training and testing by 80:20 and applying a validation factor of 0.1. Figure 4 provides a concise rundown of the total number of images for each class, as well as the number of samples that were utilized in testing and training.

```
VALIDATION_FACTOR = 0.1

val_size = int(len(x_train) * VALIDATION_FACTOR)

train_x = x_train[: len(x_train) - val_size]
train_y = y_train[: len(y_train) - val_size] # len(x_train) = len(y_train)

val_x = x_train[len(x_train) - val_size : len(x_train)]
val_y = y_train[len(y_train) - val_size : len(y_train)]

print("Shape of training data = ", train_x.shape, train_y.shape)
print("Shape of validation data = ", val_x.shape, val_y.shape)
```

```
Shape of training data =  (1639, 224, 224, 3) (1639, 4)
Shape of validation data =  (182, 224, 224, 3) (182, 4)
```

Figure 9. Training data sharing details

A. VGG-16 Arsitektur

In this test, we employed the VGG16 architecture, which is a deep convolutional neural network architecture with a total of 16 layers. ImageNet is typically utilized when looking for picture datasets to use for training deep learning architectures. ImageNet is a dataset that consists of one thousand different categories and contains millions of photos. ImageNet is frequently utilized in competitions involving machine learning. The very first thing you should do is engage in some form of transfer learning. In order to perform transfer learning, we need to uninstall the most recent layer, which is the one that has the fewest connections to the other layers in the model, and then install the following layer. Seven different layers make up the architecture of the VGG-16 model. Convolutional layers make up five of the layers, and fully connected layers make up the other two ones make up the layers [10].

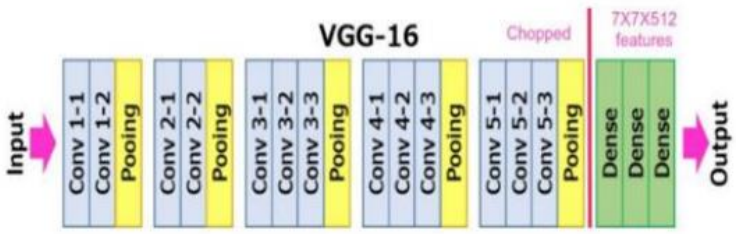

Figure 9. VGG-16 Architecture

- Input Layer

  The raw image is taken in at this input layer before being passed on to the following layer in order to have the subsequent feature extracted from it.

- Convolutionals

  Convolutional processing is applied to the subsequent input layer. In this layer, a variety of filters will be used to the object in order to discover new object features. These features or properties are employed in the calculation of whether or not a test process was a match.

$$Output = \frac{n + 2p - f}{s} + 1 \quad (2)$$

  Where:
  n = input length or input height
  f = kernel filter length or kernel feature height
  p = padding
  s = step

- Pooling

  The unification layer will receive the features that were extracted from the data. This layer is helpful for reducing the dimensions of image objects with large sizes and reducing the parameters of those image objects in order to save crucial information. It is responsible for ensuring that the maximum value of each layer is preserved. This layer contributes no weight to the overall total. The process of polling itself is

separated into two distinct steps: max-pooling, in which the system selects the result of the convolution calculation that has the largest value, and global average pooling, in which it selects the value that is averaged across all of the convolution results.

$$Outputmaxpool = \frac{n-f}{s} + 1 \quad (3)$$

- Fully Connected Layer

The topmost layer is a completely connected one, and the system operates by first converting images that have been subjected to the most stringent level of filtering into a categorical labeling system. At this layer, the input will be obtained from the results of the processing that came before it in order to determine whether characteristics are interconnected (correlated) with other classes. The goal of this layer is to consolidate all of the dimensions' individual nodes into a single dimension.

$$Z_j = \sum_{j=1}^{c} w_{i,j}^{T} X_i + b_j$$

Where:
Zj = Output value of the network
Xi = Input value from feature extraction
Wi,j = network weight value of size i x j
i = number of input features
j = number of target classes
bj = bias of the network

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0
 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792
 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928
 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0
 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856
 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584
 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0
 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168
 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080
 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0
 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160
 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808
 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0
 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808
 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

Figure 10. Model VGG-19 architecture

```
Model: "sequential_25"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_75 (Dense)            (None, 512)               262656
 dropout_50 (Dropout)        (None, 512)               0
 dense_76 (Dense)            (None, 512)               262656
 dropout_51 (Dropout)        (None, 512)               0
 dense_77 (Dense)            (None, 4)                 2052
=================================================================
Total params: 527,364
Trainable params: 527,364
Non-trainable params: 0
```

Figure11. Sequential model

B. Validation and Testing

During this stage of validation and testing, the validation factor is set to 0.1. Type optimization with Adam and SGD is performed using the following Adam parameter settings: learning rate = 0.1, beta 1 = 0.9, beta 2 = 0.999, epsilon = 1e-07, and amsgrad = false. Optimization with Adam uses the following Adam parameter settings: learning rate = 0.1, momentum = 0.0, and nesterov = false.

Table 3. Comparison of model test accuracy

Optimizers	Parameter Name	Parameter Value	Ephoch	Best Accuracy
Adam	learning_rate	0.001	Epoch 86/100	0.7418
	beta_1	0.9	Epoch 55/100	0.7473
	beta_2	0.999	Epoch 85/100	**0.7582**
	epsilon	1e-07	Epoch 32/100	**0.7582**
	amsgrad	False	Epoch 75/100	0.7418
SGD	learning_rate	0.1	Epoch 81/100	0.6758
	momentum	0.0	Epoch 97/100	0.7088
	nesterov	False	Epoch 99/100	0.6813

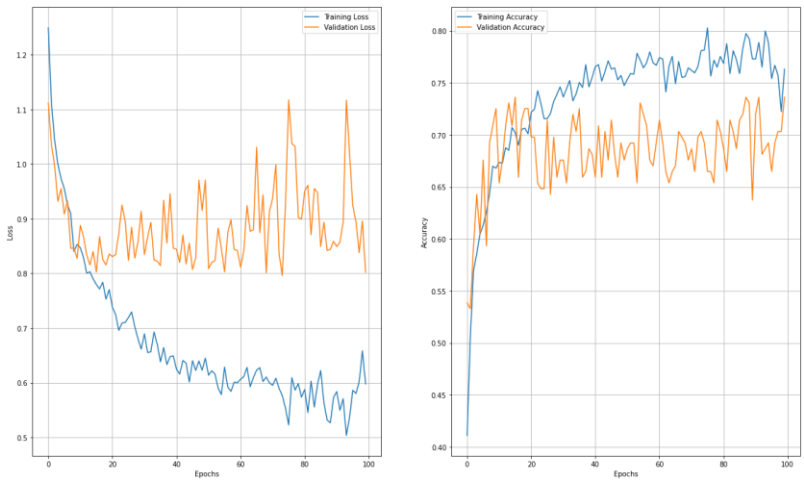

Figure 12. Validation loss and accuracy validation

IV. CONCLUSION

The findings of this study have allowed for significant advancements to be made in the diagnosis of apple leaf diseases. In order to identify abnormalities at an earlier stage, this work makes use of preprocessing techniques that incorporate a number of methodological approaches. When more conventional procedures appear to have hit a wall, preprocessing methods can improve image quality by employing techniques such as histogram equalization, edge detection, and others. This has the effect of improving the efficiency with which deep learning is carried out. After putting the proposed model through its paces, we turned to a pair of optimization models known as Adam and SGD. The parameter values for each optimization model are distinct from one another. The outcomes of this test were analyzed in order to make comparisons about the effectiveness of various optimization models. Table 3 presents the results of deep learning experiments conducted with various optimization models, employing convolutional neural networks (CNN). When we use the Adam optimization model with a parameter value of beta 2 = 0.999 at Ephoch to 85/100 with an accuracy of 0.7582, as well as when we use the epsilon

parameter value of 1E-07 at Ephoch to 32/100 with an accuracy of 0.7582, we are able to reach the highest level of precision possible. Due to the high level of accuracy achieved, it is clear that the problem of disease detection on apple leaves was successfully managed.

REFERENCES

[1] S. Hasan, S. Jahan, and M. I. Islam, "Disease detection of apple leaf with combination of color segmentation and modified DWT," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7212–7224, 2022, doi: 10.1016/j.jksuci.2022.07.004.

[2] J. Di and Q. Li, "A method of detecting apple leaf diseases based on improved convolutional neural network," *PLoS One*, vol. 17, no. 2 February, pp. 1–15, 2022, doi: 10.1371/journal.pone.0262629.

[3] A. R. Ram, "Plant Disease Detection Using Leaf Pattern : A Review," vol. 2, no. 6, pp. 774–780, 2015.

[4] M. A. Khan *et al.*, "CCDF: Automatic system for segmentation and recognition of fruit crops diseases based on correlation coefficient and deep CNN features," *Comput. Electron. Agric.*, vol. 155, no. February, pp. 220–236, 2018, doi: 10.1016/j.compag.2018.10.013.

[5] S. Alqethami, B. Almtanni, W. Alzhrani, and M. Alghamdi, "Disease Detection in Apple Leaves Using Image Processing Techniques," *Eng. Technol. Appl. Sci. Res.*, vol. 12, no. 2, pp. 8335–8341, 2022, doi: 10.48084/etasr.4721.

[6] L. Fu, S. Li, Y. Sun, Y. Mu, T. Hu, and H. Gong, "Lightweight-Convolutional Neural Network for Apple Leaf Disease Identification," *Front. Plant Sci.*, vol. 13, no. May, pp. 1–10, 2022, doi: 10.3389/fpls.2022.831219.

[7] R. A. A S and S. Gopalan, "Comparative Analysis of Eight Direction Sobel Edge Detection Algorithm for Brain Tumor MRI Images," *Procedia Comput. Sci.*, vol. 201, no. C, pp. 487–494, 2022, doi: 10.1016/j.procs.2022.03.063.

[8] K. Zhang, Y. Zhang, P. Wang, Y. Tian, and J. Yang, "An improved sobel edge algorithm and FPGA implementation," *Procedia Comput. Sci.*, vol. 131, pp. 243–248, 2018, doi: 10.1016/j.procs.2018.04.209.

[9] S. M. Abid Hasan and K. Ko, "Depth edge detection by image-based smoothing and morphological operations," *J. Comput. Des. Eng.*, vol. 3, no. 3, pp. 191–197, 2016, doi: 10.1016/j.jcde.2016.02.002.

[10] T. Sulistyowati, F. Al Zami, and R. A. Pramunendar, "VGG16 Deep Learning Architecture Using Imbalance Data Methods For The Detection Of Apple Leaf Diseases," vol. 11, no. 1, pp. 41–53, 2023.