Application of VGG-19 for Optimized Prediction of Illness in Plants

[1]Vijaya Durga Mudunuri, [2]V. Anjani Kranthi, [3]Rajesh Varma Vegesna, [4]Adina Karunasri
[1]jayavarmavijju@gmail.com, [2]vak@srkrec.ac.in, [3]vegesnarajeshvarma@gmail.com,
[4]karunasri.adina@gmail.com

[1]Department of Computer Science and Engineering, Bonam Venkata Chalamayya Engineering College, Odalarevu, Andhra Pradesh, India
[2]Department of Computer Science and Engineering, Sagi Rama Krishnam Raju Engineering College, Bhimavaram, Andhra Pradesh, India
[3]Department of Mechanical Engineering, Sagi Rama Krishnam Raju Engineering College, Bhimavaram, Andhra Pradesh, India
[4]Department of Computer Science and Engineering, Vishnu Institute of Technology, Bhimavaram, Andhra Pradesh, India

Abstract:

Food is a basic need for every living being on this planet. Therefore, increasing the quality of agricultural products and proper management of the crops is essential. Generally, it is estimated that various pests such as insects, weeds, nematodes, animals, and diseases cause crop yield losses of about 20-40%. More precisely, some data states that crop diseases cause average yield losses of 42% for the most significant food crops. For instance, Leaf Spot Diseases make the tress and the shrubs weak by interrupting photosynthesis, the process responsible for the productivity of plants. At times, crop diseases destroy the entire crop production. Due to this reason, farmers must discover the crop diseases at an early stage, identify them, and apply the suitable remedy before it gets too late. Traditional methods of detecting a disease include manually examining a leaf and predicting the disease. However, a farmer cannot accurately determine the ailment by this technique. Moreover, he will also be faced with the pressure of capturing the appropriate remedy within the expected time frame. Therefore, this project aims to utilize accurate image processing techniques to recognize the disease of the plant leaf. We have used VGG-19 to build our classification model. A database that contains all the information about the plant names and diseases will be built using JavaScript. Using this database, we will build an interactive GUI, using Python, where users can upload images and check the disease type.

Keywords: Crop Diseases, VGG-19, GUI, JavaScript, Python.

DOI Number: 10.14704/NQ.2022.20.15.NQ88104 **NeuroQuantology2022;20(15): 1183-1192**

1. INTRODUCTION

Machine Learning refers to the automated detection of meaningful patterns in data. In the past couple of decades, it has become a common tool in nearly every task that requires information extraction from large datasets. In this project, the input to the algorithm is the images of plant leaves. The model makes use of these images to classify various diseases and suggest a remedy. The acquired datasets are stored on local drive and the model will be trained on Anaconda. The project implements VGG-19 neural networks and Tensor Flow to detect leaves and leaf diseases from images to classify them. Hence, to train the model, we used a dataset that contains images of leaves with and without diseases. The dataset used, has been gathered from multiple sources on the internet. The dataset contains around 55,000 images of different plant leaves that are

healthy and that are not healthy.

They are stored in various folders, named after plant name and their health status respectively. Based on that, the images have been segregated into different folders. These folders are then mounted into the VGG-19 model to be trained and tested. The folders are generally stored in local drive for convenience. After mounting the dataset, the directory in the code is set to access the dataset. Once the dataset has been successfully mounted, these images are used for training the model on Anaconda. Convolution Neural Networks (CNN) is a class of artificial neural networks, most commonly applied to analyze visual imagery. This is a DL algorithm which takes an image as an input, assigns importance to various aspects in the image and is able to classify one from another.

VGG-19 is a Convolution Neural Network that is 19 layers deep. It is pertained version of the network trained on more than a million images from Image Net. The pertained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. Since this project deals with a wide range of plant images, VGG-19 is a suitable model for dealing with such large datasets. The network has an image input size of 224-by-224. In addition, it does not explicitly require Image Segmentation as the model is pertained.

In this project, we will be improving various layers as a first step of building our CNN Model. The layers are as follows:

- Activation
- Conv2D
- MaxPooling2D
- Dense
- Flatten
- Dropout

2. LITERATURE REVIEW

Mohanty et al. [1] recommended a proposal to identify the diseases in plants by giving instructions using convolution neural network. The CNN approach is trained to detect hygienic and infected plants of 14 kinds. Moreover, the proposed approach attained with 99.35% of reliability on validated design data. While applying the proposed approach on images taken from reliable online platforms, it accomplishes 31.4% of reliability; whereas, it is superior to an existed model of arbitrarily choice, some more divergent set of predicted data may helps to improve the reliability. In addition, few more varieties of approach or neural network validation can be aided with higher reliability, that may generates the path for detecting various diseases of various species of plants and it is very easy to operate.

Authors [2] stated that there are four major steps are to be followed in the process of detecting the infected portion of the leaves of various plant species with the help of texture attributes. Initially the color transfiguration anatomy is considered as the process variable of RGB image; thereby green pixels are being identified and disengaged using peculiar threshold parameter followed by decomposition operation. Further, the numerical data depicted by the texture attributes may be computed for getting favorable segments. Finally, the diseases are categorized by extracting the characteristics using classifier module.

By observing the literature in various research articles [3-6], authors proposed that

eISSN 1303-5150

image processing of the infected leaf. Deep forward back propagation algorithm has been implemented in adaptive neural networks in a directional path by considering the feature variables to classify the infected and non-infected plant species. Also, the proposed approach predicted with around 80% of reliability. Kulkarni et al. [7] followed the algorithm for speedy and reliability in the process of identifying the infections of various plant species using adaptive neural networks along with convergent image monitoring unit incorporated to it. For extracting the attributes of various parts of the plants, the introduced model biased with neural network classifier and Gabor detector unit which gives optimistic outputs with a utility identification rate of around 91%.

With reference to the article [8-9], CNN and ANN have been employed to identify the plant infection spots using background decomposition for establishing the suitable attribute isolation and target projection. However, it's been clear that Gans may be used to categorize the habitant issues of plant species; frame work relevant background procedure hadn't improved the reliability comparatively with other models. In comparison with many image processing techniques, deep learning integrated image processing and identification module need not to encrypt the suitable attributes of various diseased spots of plant species. Moreover, numerical approach of iterative based learning may recognize the suitable characteristics that can extract spatial and contextual attributes of the processed images having strengthened validity and more prediction stability. [10-11]. On other hand, deep learning plays a major role in the aspect of identifying the diseases spots of infected plants and pests with image processing identification. Currently, various modules of deep learning are being proposed with an integration of neural network tools that constituters DBN, DBM, SDAE and DCNN [12]. As time passes on, deep learning has been recognized as a tremendous tool as compared with tradition neural network approach and that may be practiced by both industry and academia [13-14].

3. PROPOSED APPROACH

The model operates in four modules, namely:

3.1. Preprocessing Module

Initially, all the required libraries, such as Numpy, Tensor Flow, Keras, CV2, are imported. Then the body of the code begins by defining of two one-dimensional arrays called 'images' and 'labels' which will be used later to store images from the datasets and their labels respectively. Then, all the input images from the dataset are converted into grayscale for easier classification by the model. Using RGB versions of images will only overcomplicate the classification model and it can also lead to a slower and inaccurate classification.

eISSN 1303-5150

```
import numpy as np
import pickle
import cv2
from os import listdir
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```

```
Using TensorFlow backend.
```

```
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(np_image_list, imag
e_labels, test_size=0.2, random_state = 42)
```

```
label_binarizer = LabelBinarizer()
image_labels = label_binarizer.fit_transform(label_list)
pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))
n_classes = len(label_binarizer.classes_)
```

Fig. 1. Pre-processing module

Following this, the grayscale images are re-sized to 224 x 224 for normalizing the input images. This will allow the VGG-19 model to accept the input more conveniently. Then one hot encoding is performed with the help of label binarizer, since all the labels in the dataset are in the form of text. Finally, the images are split into two groups for training and testing purpose, in this case, 80% of the dataset is used for training and the rest is used for validation purpose. Observe Fig. 1 for a better understanding of the above steps.

3.2. VGG-19 Module

Firstly, all the required libraries like keras and Tensor Flow are imported. Then the model parameters (include top, weights, pooling, etc.) are defined and the VGG-19 model is built. Number of classes is declared as 40 here. Next, adding all the layers into the VGG-19 model is done. The batch size is set to a value of 32. The layers are added into the model in the form layer groups called blocks. The layers used in the VGG-19 module include conv2D, Max Pooling, Flatten, and Activation layers with 'relu' activation. This model contains 16 Convolution layers and 3 dense layers accounting for a total of 19, as observed in Fig. 2.

Usually, when dealing with large datasets such as this, there is a chance of over fitting of the model during training process. This over fitting is said to occur when the model cannot generalize the data after being trained with a dataset. The 'dropout' layer is utilized to mitigate this possibility. Optimal values for this layer range between 0.5 and 0.8. Finally, model.summary() method is used to print the details of the VGG model. The output screen will show the shape of our VGG model including the total number of parameters along with both trainable and non-trainable ones.

eISSN 1303-5150

```
def VGG19(include_top=True, weights='None',
          input_tensor=None, input_shape=None,
          pooling=None,
          classes=40):

    #Block1
    x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv1')(img_input)
    x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv2')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool')(x)

    #Block2
    x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv1')(x)
    x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv2')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool')(x)

    #Block3
    x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv1')(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv2')(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv3')(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv4')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool')(x)

    #Block4
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv1')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv2')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv3')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv4')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool')(x)

    #Block5
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv1')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv2')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv3')(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv4')(x)
    x = MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool')(x)


    # Classification block
    x = Flatten(name='flatten')(x)
    x = Dense(4096, activation='relu', name='fc1')(x)
    x = Dense(4096, activation='relu', name='fc2')(x)
    x = Dense(classes, activation='softmax', name='predictions')(x)
```

Fig. 2. VGG-19 Module

3.3. Training and Testing Module

In the training process, only 80% of the images in the dataset are used for training and the rest for validation. Using too much data for training may lead to the model not being able to generalize the data. In addition, Adam optimizer is used to improve the training efficiency. Also known as Adaptive Moment Estimation, this algorithm is very efficient when working with very large datasets involving lots of parameters. Adam is a combination of two gradient descent Methodologies: Momentum and Root Mean Square Propagation (RMSP). Building upon the strengths of these models, Adam optimizer gives a much faster performance by a huge amount when compared to the models individually.

The value of the epoch is set to 25 because the dataset is already large enough. (Too many epochs with a large dataset will be time consuming and redundant). With each epoch, the model should witness increased accuracy and reduced validation error. And during this process, whenever a model with lesser validation loss is detected, it is immediately saved as the main model and this model will be then trained in the subsequent epoch to further improve its accuracy. The code for this can be observed in Fig. 3(a).

After training and testing the model, the accuracies and errors are plotted against epoch count to observe the graph, as shown in Fig. 3(b). From the graph, it can be observed that the training loss is almost equal to validation loss and the training accuracy is almost equal to validation accuracy, which are both good indications that this model is a good fit. This indicates that the model is able to generalize the dataset during validation process very well. And hence, it is a good fit model. From Fig. 3(b), it can be seen that the accuracy of the trained model evidently is 96.7%.

eISSN 1303-5150

Accuracy above 95% is generally seen as a very good sign for a CNN model this complex. This model, which lies well within the optimal range of 95-100%, is a very accurate model. After evaluating the loss and accuracy of the model, it is saved using the model. save() method, as shown in Fig. 3(c). This model is named as 'cnn_model.pkl'.


```
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
# distribution
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])
# train the network
print("[INFO] training network...")
```

```
[INFO] training network...
```

```
history = model.fit_generator(
    aug.flow(x_train, y_train, batch_size=BS),
    validation_data=(x_test, y_test),
    steps_per_epoch=len(x_train) // BS,
    epochs=EPOCHS, verbose=1
    )
```

Fig. 3(a). Training and Testing Module

Fig. 3(b). Evaluation graph of our Model

```
print("[INFO] Calculating model accuracy")
scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
```

```
[INFO] Calculating model accuracy
591/591 [==============================] - 2s 3ms/step
Test Accuracy: 96.77383080755192
```

Save model using Pickle

```
# save the model to disk
print("[INFO] Saving model...")
pickle.dump(model,open('cnn_model.pkl', 'wb'))
```

```
[INFO] Saving model...
```

Fig. 3(c). Accuracy of the model

4. RESULT ANALYSIS

Firstly, all the necessary libraries such as keras sklearn, numpy and pickle are to be imported. The database containing all the diseases and remedies is imported and saved in a csv file format. It will be used in the backend of the GUI. Hence, this will be useful in showing the names of all diseases along with their remedies.

Then the previously trained model is loaded into the prediction module with the help of load_model method. The classifier model is fitted, following which prediction is performed using the "classifier. predict()" command. Finally, prediction on whether the plant is infected or not, is done by the model. Here, a variable called 'pred' to store the prediction output. Different values of pred are added and stored in 'sum'. If sum < 2, it means that the plant is healthy, else it is infected. The GUI of the output can be observed in Fig. 4(a) and 4(b).

The GUI module was built using PyQt5 and it is used to import our VGG module by selecting the "Tensor flow" option. After that, we can press the "Browse" button to select an input image which will be loaded into the model. Finally, press "ok" after selecting the image. An output containing the plant's name, disease and a treatment/remedy will be displayed in the "Disease Prediction" window.

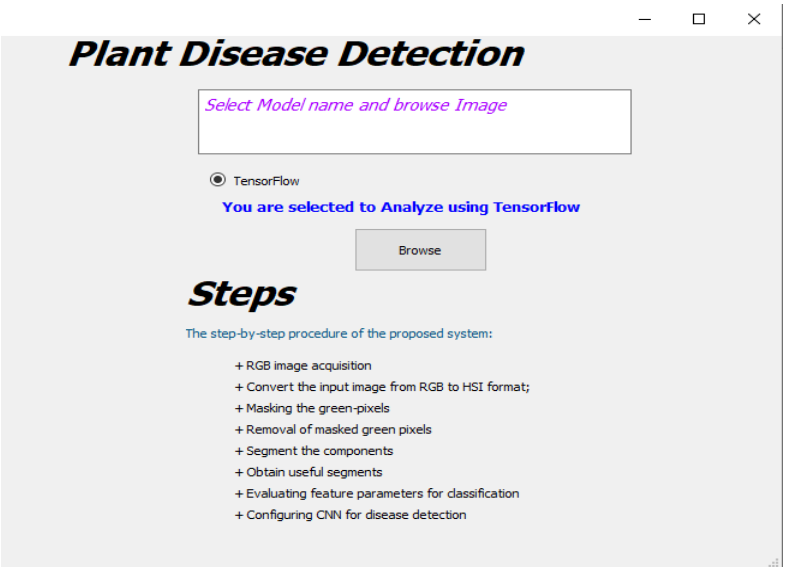

Fig. 4(a). GUI Output

eISSN 1303-5150

Fig. 4(b) GUI Output

For further analysis of results, this VGG-19 model is compared with another trained VGG 16 model to compare performance. In the second model, SIGMOID activation function is used instead of RELU. One advantage RELU has over sigmoid is the reduced likelihood of the gradient vanishing. Also, RELU shows more sparsity compared to sigmoid. Two major benefits of RELUs are sparsity and a reduced likelihood of vanishing gradient. To elaborate, the definition of a RELU function is h= max (0,) where a = W(x) + b. One major benefit is, the reduced probability of the gradient to vanish. This arises when a>0. In RELU, the gradient has a constant value, but same is not the case with sigmoid. The gradient of sigmoid becomes progressively small as the absolute value of x increases.

Another benefit of RELU is sparsity. This arises when a<=0. The more such units that exist in a layer, the sparser the resulting representation will be. But, sigmoid on the other hand always generates some non-zero value resulting in dense representations. Sparse representations seem to be more beneficial than dense representations. Hence, the constant value of gradient and sparsity in RELU ultimately results in faster learning. This difference in learning rate leads to different accuracies in VGG-19 and VGG-16 models tested here. The following graph in Fig. 4(c), shows that the accuracy of VGG-19 model at 10 epochs is over 94% while the accuracy of VGG- 16 at 10 epochs is observed to be less than 92%. A more detailed comparison between accuracies and epochs of VGG-19 and VGG-16 can be seen in the below Table 1.

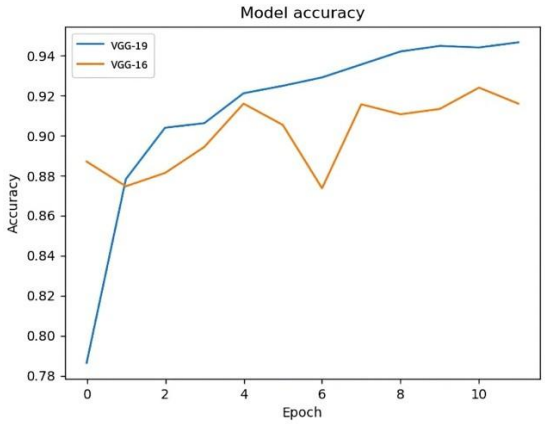

Fig. 4(c) Accuracy comparison between VGG- 19 and VGG-16 models.

Table 1: Comparison of VGG-16 and VGG-19 accuracies at epochs

Model	Accuracy at Epoch 2	Accuracy at Epoch 4	Accuracy at Epoch 6	Accuracy at Epoch 8	Accuracy at Epoch 10
VGG-19	91.8%	92.2%	92.6%	93.7%	94%
VGG-16	87.8%	91%	87%	91%	92%

5. CONCLUSION

The main objective of this project is to reduce human involvement in detecting the presence of diseases in plants by performing image classification and identifying the disease, along with an appropriate remedy. Users can upload images of plant leaves in the GUI. The output will be presented on the screen in the form of the name of the classified disease, along with the recommendation of remedies to combat the identified disease, to take early action. We believe that Plant Disease Detection System will be one of the leading digital solutions for most industries, such as agriculture, healthcare, and corporate sectors in a future where plant diseases are likely to be the norm.

6. FUTURE SCOPE

To expand the project in future, the following elements can be added. Firstly, the interface can be displayed in both English and Hindi languages for better accessibility. Secondly, simplifying the interface and developing a mobile version of this software. Lastly, incorporating the software into a CCTV which will periodically take pictures of crops and analyze them for the presence of diseases and sends alerts to farmers, prompting them to take action. This monitoring system will not only help farmers but also researchers in biological labs and greenhouses in early disease detection and resolution.

REFERENCES

[1] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in plant science*, *7*, 1419.

[2] Arivazhagan, S., Shebiah, R. N., Ananthi, S., & Varthini, S. V. (2013). Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features. *Agricultural Engineering International: CIGR Journal*, *15*(1), 211-217.

[3] Riley, M. B., Williamson, M. R., & Maloy, O. (2002). *Plant disease diagnosis. The Plant Health Instructor. DOI: 10.1094*.

eISSN 1303-5150

PHI-I-2002-1021-01.

[4] Sankaran, S., Mishra, A., Ehsani, R., & Davis, C. (2010). A review of advanced techniques for detecting plant diseases. *Computers and electronics in agriculture*, *72*(1), 1-13.

[5] Georgakopoulou, K., Spathis, C., Petrellis, N., & Birbas, A. (2015). A capacitive-to-digital converter with automatic range adaptation for readout instrumentation. *IEEE Transactions on Instrumentation and Measurement*, *65*(2), 336-345.

[6] Arnal Barbedo, J. G. (2013). Digital image processing techniques for detecting, quantifying and classifying plant diseases. *SpringerPlus*, *2*(1), 1-12.

[7] Kulkarni, A. H., & Patil, A. (2012). Applying image processing technique to detect plant diseases. *International Journal of Modern Engineering Research*, *2*(5), 3661-3664.

[8] Mix, C., Picó, F. X., & Ouborg, N. J. (2003). A comparison of stereomicroscope and image analysis for quantifying fruit traits. *Seed Technology*, 12-19.

[9] Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, *147*, 70-90.

[10] Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. *Computers and electronics in agriculture*, *145*, 311-318.

[11] Liu, J., & Wang, X. (2021). Plant diseases and pests detection based on deep learning: a review. *Plant Methods*, *17*(1), 1-18.

[12] Atila, Ü., Uçar, M., Akyol, K., & Uçar, E. (2021). Plant leaf disease classification using Efficient Net deep learning model. *Ecological Informatics*, *61*, 101182.

[13] Dhaka, V. S., Meena, S. V., Rani, G., Sinwar, D., Ijaz, M. F., & Woźniak, M. (2021). A survey of deep convolutional neural networks applied for prediction of plant leaf diseases. *Sensors*, *21*(14), 4749.

[14] Sujatha, R., Chatterjee, J. M., Jhanjhi, N. Z., & Brohi, S. N. (2021). Performance of deep learning vs machine learning in plant leaf disease detection. *Microprocessors and Microsystems*, *80*, 103615.