Original Research Article

Study on the Prognostication of Crop Diseases using Artificial Intelligence

ABSTRACT

It is universally accepted fact that crop diseases are one of the major threats in agriculture that ultimately result in drastic reduction of food supply. The present research study aims to use artificial intelligence in building a model which is integrated with a user-friendly web application. The web application is created using the Python-based Django framework. This user interface allows the user to choose a crop name and upload an image of a leaf wherein the trained model then begins the process of feature extraction on the image and tries to make an accurate prediction. The final result is displayed to the user confirming whether the crop may be "healthy" or the "diseased "and even the name of the disease that infects the plant will be displayed. The application also suggests a suitable treatment to combat the disease. Thus, the scope of this research study is very scalable as it can be easily be used by amateur gardeners as well as by farmers. The model itself can also be extended to include more plant types along with any new diseases which may arise due to factors like climate change, pest - resistance etc.

Key Words: Convolutional neural network (CNN); deep learning; healthy leaf; infected leaf; plant disease

INTRODUCTION

The agriculture industry is an important part of civilization and the recent demand for the more output from the industry had brought forth the need to study, understand and improve the field. This coupled with the budding application of artificial intelligence as a means and method of problem solving has given tools like machine learning, neural networks, and deep learning to best tackle and solve this problem as the latest generation of neural networks has achieved impressive results in the field of classifying of images.

This research study (project) is on the prognostication of crop diseases using a well designed and developed model that will be able to detect the presence of a leaf and differentiate between healthy leaves and the disease –infected leaves as well as identify the type of disease visually is a step to introduce high-end technology into the agricultural industry. Earlier Researchers like Yann LeCun et al. [1] reported on the concept of deep learning and how it changed and improved the fields of visual object recognition, speech recognition etc. and the breakthroughs made by deep convolutional nets. Saad Albawi et al [2] discussed on the concepts behind a convolutional neural network including an in-depth

analysis of its many layers including convolutional layer, pooling layer, non-linearity layer and fully-connected layer. The paper also mentions the applications of CNN (covolutional neural network) which lie mainly in the fields on computer vision and natural language processing. Further, Hammad Saleem et al. [3] also reported on the importance of the early detection and identification of plant diseases by employing numerous deep learning models, their features and pros and cons of their usage.

METHODOLOGY

The model has been designed to detect the disease a leaf is suffering from when an image is uploaded. This model itself can also be extended to include more plant types along with any new diseases which may arise due to other factors.

Software and Hardware Requirements

The hardware requirements are laptop with minimum 4GB RAM, minimum 4GB GPU, hard disk, plant disease image dataset(894 mb) and crop production CSV Dataset(264 kb). The software requirements are Python 3.6, Jupyter Notebook IDE and Atom IDE.

Plant Disease Detection

In the present research study, the concept of deep learning is applied to classify a plant disease by the construction of a convolutional neural network (CNN). A convolutional neural network is a type of neural network specialized to handle visual imagery. Firstly, the plants employed in the present study along with their probable infectious diseases are mentioned below

- Apple: Apple Scab, Black Rot, Cedar Apple Rust and Healthy
- Cherry: Powdery Mildew and Healthy
- Corn: Common Rust, Crecospora, Northern Leaf Blight and Healthy
- Grape: Black Rot, Esca, Leaf Blight and Healthy
- Peach: Bacterial Spot and Healthy
- Pepper: Bacterial Spot and Healthy
- Potato: Early Blight, Late Blight and Healthy
- Strawberry: Leaf Scorch and Healthy
- Tomato: Bacterial Spot, Early Blight, Late Blight, Leaf Mold, Spider Mites, Target Spot, Mosaic Virus, Yellow Leaf Curl Virus and Healthy

The entire flow of this proposed system is explained in the flowchart below

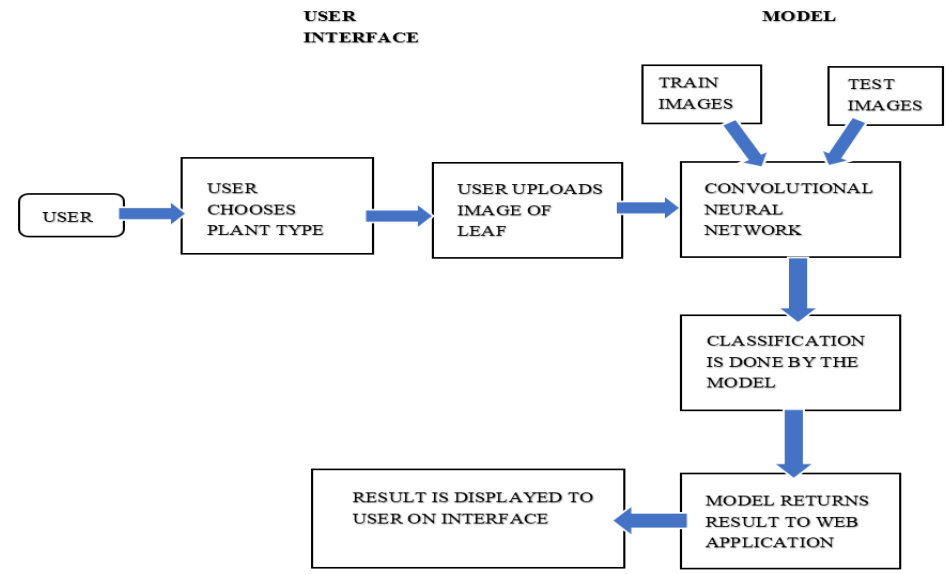

Figure 1. *Proposed architecture for detecting disease on a plant leaf*

The flow proceeds as follows

- The user selects the plant type from the navigation bar present on the user interface.
- Once a plant type is selected, the user is directed to another page where they will be prompted to upload an image of the plant's leaf.
- Once the requested image is uploaded, it is given to the neural network which performs classification.
- If the diagnosis is healthy, then the result healthy along with a fertilizer to boost the yield of the crop.
- If a certain disease is diagnosed, then the result is returned with the name of the disease and a suitable treatment plan using any pesticides or herbicides.
- The final result is displayed to the user on the user interface.
- The user can upload another image to repeat the process or choose another plant type.

RESULTS AND DISCUSSION

Once the user interface is opened, the following home page is shown

Figure 2. *Home Page.*

The navigation bar on the top of the page is used to navigate through the interface. Suppose we click on ‘About the Project’, we are navigated to the new page as shown in Figure 3.

Figure 3. *Project Page.*

Similarly, we can use the navigation bar to click on Plant Disease Classification and choose a plant type from the drop down menu (Figure 4).

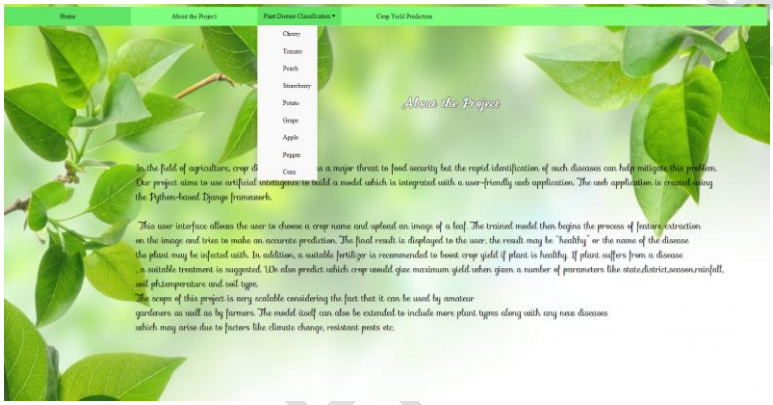

Figure 4. *Choosing a plant type from the drop down menu.*

Thus now suppose we click on plant type tomato. We can now illustrate how our model tries classifying the leaves of a tomato plant. First, we upload an image of a tomato leaf suffering from late blight as shown in the figure below.

Figure 5. *Uploading an image of a tomato leaf suffering from late blight.*

Once the submit button is created, the image is given to the deep learning model which makes an attempt to classify the leaf to one of the learned categories. As seen in the following image (Figure 5), the image is classified to be late blight and a treatment is also suggested. Thus the user can apply this treatment to cure their tomato plant.

Figure 6. *Late blight is identified and treatment is suggested after pressing submit.*

However, suppose the outcome turns out to be healthy. This outcome is illustrated in the following two pictures (Figure 7 and Figure 8). If an image of a healthy tomato leaf is uploaded and submitted, the image is classified as healthy and a group of fertilizers are suggested to improve the yield of the healthy crop.

Figure 7.*Uploading the image of a healthy tomato leaf*

Figure 8. *Fertilizers are suggested after pressing submit.*

Similarly we test this out on some other plant types and view the obtained results. Let us illustrate the outcomes for the plants cherry and pepper. The results for cherry are demonstrated in Figures 9 to 12. Similarly the results for pepper are demonstrated in Figures 13 to 16.

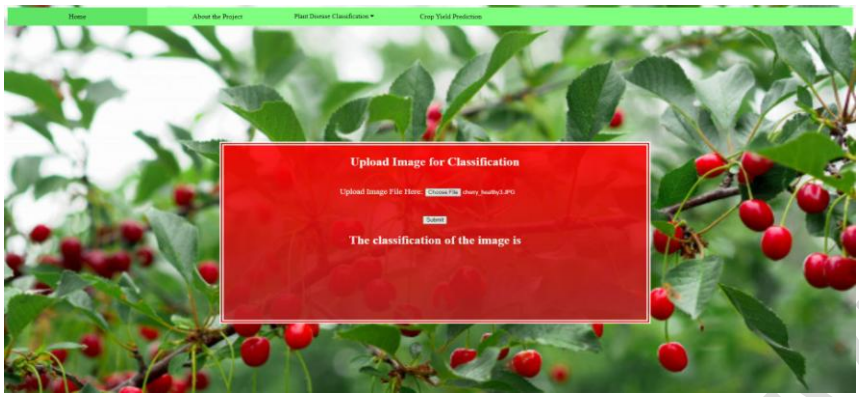

Figure 9. *An image of a healthy cherry leaf is uploaded.*

Figure 10. *Leaf is diagnosed to be healthy and fertilizer is recommended.*

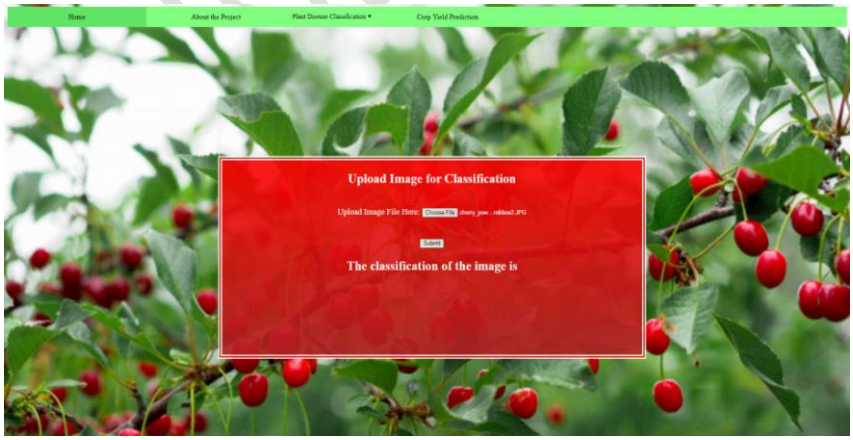

Figure 11. *An image of a cherry leaf suffering from powdery mildew is uploaded.*

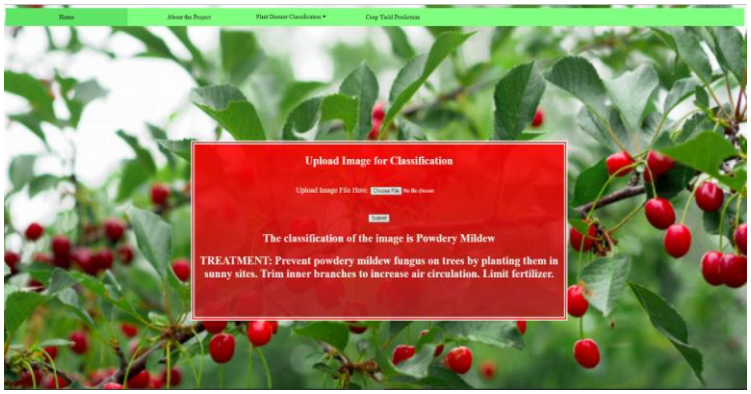

Figure 12. *Powdery mildew is diagnosed and treatment is suggested.*

Figure 13. *An image of a healthy pepper leaf is uploaded.*

Figure 14. *Leaf is diagnosed to be healthy and fertilizer is recommended.*

Figure 15. *An image of a pepper leaf suffering from bacterial spots is uploaded.*

Figure 16. *Bacterial Spots are diagnosed and treatment is suggested.*

The work done by earlier researchers pertaining to the employment of artificial intelligence to be used in the field of agriculture paved way for the current research study (project). R. Neela et al. [4] proposed that a system in which segmentation is done by guided active contour and identification is done using support vector machines and finally a fertilizer is recommended using a disease similarity method and the final accuracy observed was 80%. Konstantinos et al. [5] explained on the role of applications of machine learning in the field of agriculture viz., crop management, yield prediction, weed detection, plant disease detection, water and soil management etc. The present research study (project) was also created which involved a neural network construction for each of these plants. The final model obtained is used to perform the classification. The first step was to construct a neural network to import the required libraries (Figure 17). The most common library used to construct a neural network was keras. Data was loaded using the Image Data Generator (Figure 18).

The Image Data Generator is a preprocessing technique which allows us to augment the images in real time. It provides many transformation techniques like rotation, shifts, brightness change, flips, zooms etc. This class ensures that the model receives new variations of the images at each epoch thus creating a more general model by avoiding over fitting. In the figures below, the construction of a convolutional neural network for the cherry plant is

illustrated. A similar method is employed for the remaining with slight changes when required (Figure 19).

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
```

Figure 17. *Importing the required libraries.*

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```

Figure 18. *ImageDataGenerator.*

```
x_train=train_datagen.flow_from_directory(r"D:\MAJOR PROJECT\cherry\train",target_size=(64,64),batch_size=64,class_mode="binary"
x_test=test_datagen.flow_from_directory(r"D:\MAJOR PROJECT\cherry\test",target_size=(64,64),batch_size=64,class_mode="binary")
```

Figure 19. *Reading the image dataset.*

The second step involves initializing the model wherein the usage of the sequential class for initialization was employed. The sequential class is generally used to define a linear initialization of network layers which together constitute a proper model. Then layers are added to the model using the add method. Convolution layer is the first layer which is used to extract features from an input image. It is a mathematical operation that can take two inputs such as an image matrix and a filter (kernel). The pooling layer reduces the spatial size of the representation to reduce the amount of parameters and computation in the entire network. In this research study (project), MaxPooling wass used. The Flattened layer allows us to change the shape of the data from a 2D matrix into another format. This format can be understood by the dense layers. All these steps are illustrated in Figure 20.

```
model=Sequential()
```

```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation="relu"))
```

```
model.add(MaxPooling2D(pool_size=(2,2)))
```

```
model.add(Flatten())
```

```
model.add(Dense(units=128,kernel_initializer="he_uniform",activation="relu"))
```

```
model.add(Dense(units=1,kernel_initializer="he_uniform",activation="sigmoid"))
```

Figure 20. *Adding layers to the convolutional neural network.*

The next step involves configuring the learning process (Figure 21). As model architecture is built, the model can now be compiled. Compilation normally requires three arguments: optimizer, loss and metrics. As shown in the figure below, for plants like cherry, the three parameters would be adam, binary_crossentropy and accuracy respectively. However for other plants involving multiclass classification, the loss would be categorical_crossentropy instead.

```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])
```

Figure 21. *Compiling the Model.*

Finally, the model is fit (Figure 22). Number of epochs is usually a hyper parameter which can be tuned depending on the particular instance.

```
model.fit_generator(x_train,steps_per_epoch=23,epochs=10,validation_data=x_test,validation_steps=10)
```

```
Epoch 1/10
23/23 [==============================] - 9s 405ms/step - loss: 0.5687 - acc: 0.7591 - val_loss: 0.3374 - val_acc: 0.8442
Epoch 2/10
23/23 [==============================] - 9s 382ms/step - loss: 0.2047 - acc: 0.9237 - val_loss: 0.1109 - val_acc: 0.9756
Epoch 3/10
23/23 [==============================] - 9s 382ms/step - loss: 0.1426 - acc: 0.9497 - val_loss: 0.0879 - val_acc: 0.9773
Epoch 4/10
23/23 [==============================] - 9s 383ms/step - loss: 0.0990 - acc: 0.9721 - val_loss: 0.0947 - val_acc: 0.9838
Epoch 5/10
23/23 [==============================] - 9s 383ms/step - loss: 0.0686 - acc: 0.9850 - val_loss: 0.1125 - val_acc: 0.9740
Epoch 6/10
23/23 [==============================] - 9s 387ms/step - loss: 0.0835 - acc: 0.9720 - val_loss: 0.0901 - val_acc: 0.9854
Epoch 7/10
23/23 [==============================] - 9s 382ms/step - loss: 0.0468 - acc: 0.9898 - val_loss: 0.0412 - val_acc: 0.9935
Epoch 8/10
23/23 [==============================] - 9s 387ms/step - loss: 0.0396 - acc: 0.9905 - val_loss: 0.0365 - val_acc: 0.9951
Epoch 9/10
23/23 [==============================] - 9s 377ms/step - loss: 0.0285 - acc: 0.9939 - val_loss: 0.0307 - val_acc: 0.9935
Epoch 10/10
23/23 [==============================] - 9s 384ms/step - loss: 0.0240 - acc: 0.9952 - val_loss: 0.0370 - val_acc: 0.9951

<keras.callbacks.History at 0x18d2cdc5c48>
```

Figure 22. *Model is fit.*

Once the model is fit, it is saved as a .h5 file (Figure 23). This file stores the entire neural network including the weights between the neurons. This model can be deployed in a User Interface to make the necessary predictions.

```
model.save("cherry.h5")
```

Figure 23.*Model is saved.*

Conclusion

Thus the present study was created to effectively diagnose a plant disease based on an input image of a leaf. When the diagnosis turns out to be healthy then a suitable fertilizer would be recommended by the user interface and when a disease is

detected and identified, a suitable treatment plan would be suggested. The main aim was to build a model that could handle various plant types and diseases.

REFERENCES

[1] Yann LeCun, Y.Bengio and Geoffrey Hinton, "Deep Learning", Nature, Vol. 521, (2015) May, pp.436-44.

[2] Saad Albawi, Tareq Abed Mohammed and Saad Alzawi, "Understanding of a Convolutional Neural Network", The International Conference on Engineering and Technology, (2017) August.

[3] Saleem MH, Potgieter J and Arif KM, "Plant Disease Detection and Classification by Deep Learning", Plants, Vol. 8, Issue 11, (2019) October, pp. 468.

[4] R. Neela and P. Nithya, "Fertilizers Recommendation System For Disease Prediction In Tree Leave", International Journal Of Scientific & Technology Research, Vol. 8, Issue 11, (2019) November, pp. 3343-3346.

[5] Konstantinos G Liakos, Patrizia Busato, Dimitrios Moshou, Simon Pearson and Dionysis Bochtis, "Machine Learning in Agriculture:A Review", Sensors, Vol. 18, Issues 8, (2018)August.