Leaf Disease Detection Using Deep Learning

OnkarUmesh Kadam
Dept. of Information Tech.
Gov. College of Engg.
Karad
onkarkadam27@gmail.com

Sanket Jaising Bendale
Dept. of Information Tech.
Gov. College of Engg.
Karad
sanketbendale25@gmail.com

Kshitij Ganesh Bargale
Dept. of Information Tech.
Gov. College of Engg.
Karad
bargalekshitij7@gamil.com

Prof. N . S . Deokule
Dept. of Information Tech.
Gov. College of Engg.
Karad
niranjan.deokule@gmail.com

Abstract— The fundamental objective of this paper is to build up a product model, to recommend healing measures for vermin or sickness the board in agrarian harvests. Utilizing this product, the client can filter a tainted leaf to recognize the types of leaf, nuisance or illness frequency on it and can get answers for its control. The product framework is separated into modules in particular: Leaves Processing, Network Training, Leaf Recognition, and Expert guidance. In the primary module edge of the leaf and token qualities found. The Second module manages the preparation of the leaf to the neural system and finding the blunder chart. The third and fourth modules are to perceive the types of the leaf and recognize the nuisance or sickness frequency. The last module is planned for coordinating the perceived vermin or sickness test on to the database where in bother infection picture tests and rectifying healing measures for their administration are put away.

Keywords—

Introduction

India is an Agriculture based nation. Wherein 70% of the populace relies upon Agriculture. At the point when bugs and illnesses influence the yields, there will be a gigantic reduction underway. In a large portion of the cases vermin or infections are seen on the leaves or stems of the plant. About all the ranchers are encircled with the innovation. Genuine advantages of innovation have not been reached to the townspeople particularly to the rancher network. Proposed task will concentrate on prosperity of ranchers. The ranchers can utilize this model remotely from anyplace, whenever. The fundamental point of this task is to build up a product model for recognition of leaf sickness just as to recommend medicinal measures for nuisance or ailment the board in rural harvests. Utilizing this product, the client can examine a contaminated leaf and get arrangements. So as to expand the harvest efficiency, ranchers approach specialists to look for their recommendation with respect to the treatment of frequency of nuisance and ailments to their yields and proposals for control. Once in a while they need to go significant distances to contact specialists. Despite the fact that they go such separations master may not be accessible around then. Now and again, the master whom a rancher contacts, may not be in a situation to prompt the rancher with the accessible data and information. For these situations looking for the master exhortation is extravagant and tedious. Henceforth, it turns into a proper time for ranchers to pick up innovation advantage so as to apply proficient cultivating that gives them rewarding benefits for long terms.

Problem Statement

The software model mainly aims to save time of farmers using image processing. Like all other sectors which are gaining advantage of advanced mobile technology, Agricultural sector is also in the race to gain technology advantage. It is always said that advancement in technology is boon for a human being if it utilized for benefits of them.

In order to increase the crop productivity, farmers approach experts to seek their advice regarding the treatment of incidence of pest and diseases to their crops and suggestions for control. Sometimes they have to go long distances to contact experts. Even though they go such distances expert may not be available at that time. Sometimes, the expert whom a farmer contacts, may not be in a position to advise the farmer with the available information and knowledge. In this case, seeking the expert advice is very expensive and time consuming.

Hence, it becomes an appropriate time for farmers to gain technology advantage in order to apply efficient farming that gives them lucrative profits for long terms.

A. Methodology

I. CONCEPTUAL DIAGRAM

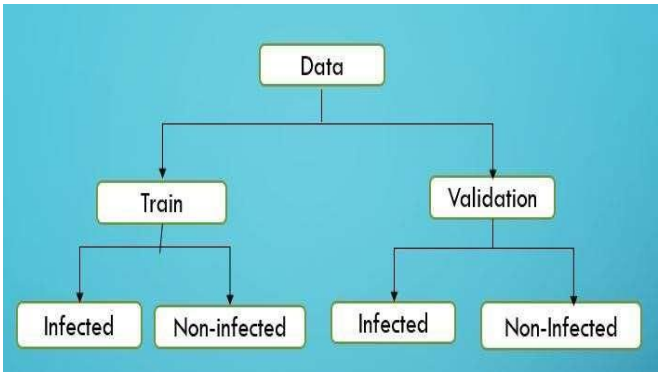


Fig. 1. Implementation Diagram

According to conceptual diagram, leaf image will be taken as input. After pre-processing and the feature extraction processes, the leaf can be identifies as infected or non-infected using convolutional neural network classification. If the leaf is infected then the suggestions regarding will be provided to the user. For making such convolutional neural network, we need to train images from the database.

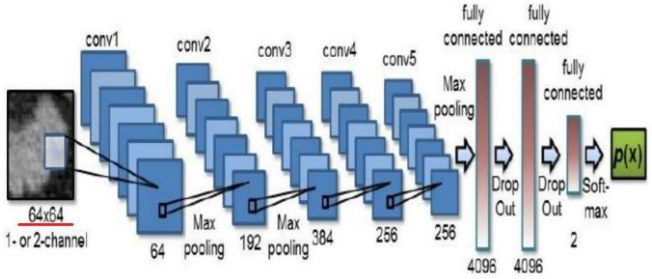


Fig. 2. CNN Architecture

A. Proposed System

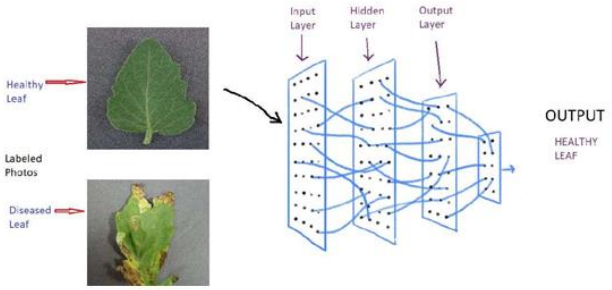
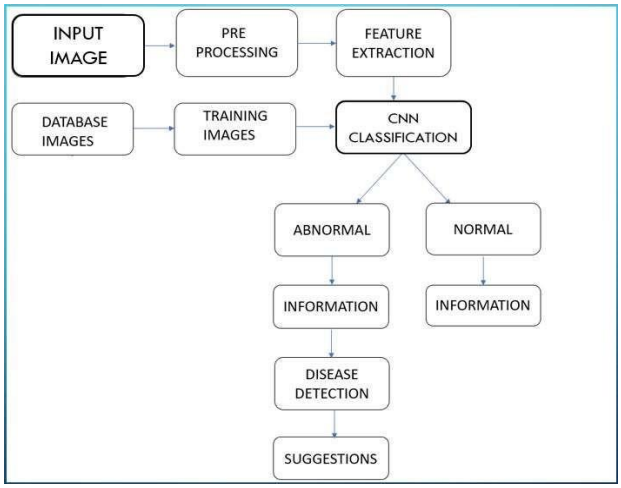


Fig. 3. System Architecture

We propose to recognize the infected or non-infected leaf with the help of convolutional neural network. We are supposed to leaf image into the model so to get the prediction. It would be completely based on Deep Learning which makes it too easy to deploy on CPU as well as on GPU that too at very low cost and can work in all computers.

B. Module Description

We have developed a convolutional neural network to predict the leaf infection. Steps we have used:



- Creating a dataset: As shown in figure we have formed directories.

Fig. 4. Directory Diagram

- Importing Libraries: We have keras library of python with tensorflow as Back-end.
- Generation of Data: We have generate 4 times of the train images by Re-scaling, shear ranging, zoom ranging and by horizontal flipping.
- Creating a Neural Network: We have Conv2D library to extract 64 features from every image and activation functions like ReLU and Sigmoid are used in this model. Then we have saved them into the weights so that if we want to use this model in practical manner, we can use only weights.

B. System Design

Convolutional neural networks (CNNs) are the current state-of-the-art model architecture for image classification tasks. CNNs apply a series of filters to the raw pixel data of an image to extract and learn higher-level features, which the model can then use for classification.

CNNs contains three components:

A. Convolutional layers:

This layer applies a specified number of convolution filters to the image. For each sub region, the layer performs a set of mathematical operations to produce a single value in the output feature map. Convolutional layers then typically apply a ReLU activation function to the output to introduce non linearities into the model.

B. Pooling layers:

Pooling layers down example the picture information removed by the convolutional layers to diminish the dimensionality of the element map so as to diminish preparing time. A normally utilized pooling calculation is max pooling, which concentrates sub locales of the component map (e.g., 2x2-pixel tiles), keeps their most extreme worth, and disposes of every single other worth.

C.Dense (fully connected) layer:

Conveying this information forward we utilize these element maps (yield picture of convolution layer) identified by CNN model to recognize the tainted leaf.

II. TECHNICAL SPECIFICATIONS

Hardware requirement: -

- CPU – Recommended CPU Specification
- Processor - Intel® Core™ i5-7200U CPU @ 2.50Ghz 2.71 GHz Ram – 8 GB
- System – 64-bit OS

Software Requirements: -

- Jupyter Notebook, Anaconda Navigator.
- Python – Python 3.6 version is required.

Additional Python Libraries required:

- numpy, keras, tensorflow.
- Tensorflow – Tensorflow 2.0 version is required.

III. RESULTS

```
(tensorflow3) D:\>python pro1.py
Using TensorFlow backend.
Found 3000 images belonging to 2 classes.
Found 600 images belonging to 2 classes.
2020-05-08 03:29:42.089189: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 254, 254, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 254, 254, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 127, 127, 32)      0
=================================================================
Total params: 896
Trainable params: 896
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 254, 254, 32)      896
_________________________________________________________________
activation_1 (Activation)    (None, 254, 254, 32)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 127, 127, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 125, 125, 32)      9248
_________________________________________________________________
activation_2 (Activation)    (None, 125, 125, 32)      0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 62, 62, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 60, 60, 64)        18496
_________________________________________________________________
activation_3 (Activation)    (None, 60, 60, 64)        0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 57600)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                3686464
_________________________________________________________________
activation_4 (Activation)    (None, 64)                0
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_5 (Activation)    (None, 1)                 0
=================================================================
Total params: 3,715,169
Trainable params: 3,715,169
Non-trainable params: 0
_________________________________________________________________
Epoch 1/50
50/50 [==============================] - 86s 2s/step - loss: 0.7240 - accuracy: 0.6613 - val_loss: 0.4406 - val_accuracy: 0.6400
Epoch 2/50
50/50 [==============================] - 83s 2s/step - loss: 0.4436 - accuracy: 0.8407 - val_loss: 0.0342 - val_accuracy: 0.9900
Epoch 3/50
50/50 [==============================] - 78s 2s/step - loss: 0.2363 - accuracy: 0.9213 - val_loss: 0.0865 - val_accuracy: 0.9533
Epoch 4/50
50/50 [==============================] - 79s 2s/step - loss: 0.1995 - accuracy: 0.9547 - val_loss: 0.0211 - val_accuracy: 0.9900
Epoch 5/50
50/50 [==============================] - 86s 2s/step - loss: 0.1108 - accuracy: 0.9640 - val_loss: 0.0304 - val_accuracy: 0.9900
Epoch 6/50
50/50 [==============================] - 84s 2s/step - loss: 0.1740 - accuracy: 0.9520 - val_loss: 0.1970 - val_accuracy: 0.9500
Epoch 7/50
50/50 [==============================] - 89s 2s/step - loss: 0.1465 - accuracy: 0.9720 - val_loss: 0.0682 - val_accuracy: 0.9967
Epoch 8/50
50/50 [==============================] - 84s 2s/step - loss: 0.2216 - accuracy: 0.9827 - val_loss: 0.1574 - val_accuracy: 0.9800
Epoch 9/50
50/50 [==============================] - 84s 2s/step - loss: 0.1562 - accuracy: 0.9873 - val_loss: 0.0048 - val_accuracy: 0.9833
Epoch 10/50
50/50 [==============================] - 82s 2s/step - loss: 0.0314 - accuracy: 0.9920 - val_loss: 0.0106 - val_accuracy: 1.0000
Epoch 11/50
50/50 [==============================] - 88s 2s/step - loss: 0.1433 - accuracy: 0.9773 - val_loss: 0.0106 - val_accuracy: 0.9967
Epoch 12/50
50/50 [==============================] - 83s 2s/step - loss: 0.0666 - accuracy: 0.9767 - val_loss: 5.3834e-04 - val_accuracy: 0.9867
```

```
Epoch 34/50
50/50 [==============================] - 75s 1s/step - loss: 0.0305 - accuracy: 0.9887 - val_loss: 3.1125e-04 - val_accuracy: 1.0000
Epoch 35/50
50/50 [==============================] - 76s 2s/step - loss: 0.0526 - accuracy: 0.9893 - val_loss: 1.4004e-04 - val_accuracy: 0.9833
Epoch 36/50
50/50 [==============================] - 75s 2s/step - loss: 0.1269 - accuracy: 0.9900 - val_loss: 6.4038e-05 - val_accuracy: 0.9967
Epoch 37/50
50/50 [==============================] - 78s 2s/step - loss: 0.1321 - accuracy: 0.9847 - val_loss: 1.6840e-04 - val_accuracy: 0.9933
Epoch 38/50
50/50 [==============================] - 76s 2s/step - loss: 0.0117 - accuracy: 0.9953 - val_loss: 2.3349e-08 - val_accuracy: 1.0000
Epoch 39/50
50/50 [==============================] - 77s 2s/step - loss: 0.0758 - accuracy: 0.9773 - val_loss: 3.0863e-04 - val_accuracy: 1.0000
Epoch 40/50
50/50 [==============================] - 76s 2s/step - loss: 0.0497 - accuracy: 0.9893 - val_loss: 0.0889 - val_accuracy: 0.9833
Epoch 41/50
50/50 [==============================] - 77s 2s/step - loss: 0.7553 - accuracy: 0.9840 - val_loss: 2.0941e-06 - val_accuracy: 1.0000
Epoch 42/50
50/50 [==============================] - 76s 2s/step - loss: 5.8439e-04 - accuracy: 1.0000 - val_loss: 1.6204e-06 - val_accuracy: 0.9967
Epoch 43/50
50/50 [==============================] - 77s 2s/step - loss: 0.7022 - accuracy: 0.9840 - val_loss: 1.0237e-05 - val_accuracy: 0.9967
Epoch 44/50
50/50 [==============================] - 76s 2s/step - loss: 0.5856 - accuracy: 0.9793 - val_loss: 3.8781e-06 - val_accuracy: 1.0000
Epoch 45/50
50/50 [==============================] - 77s 2s/step - loss: 0.0026 - accuracy: 0.9987 - val_loss: 5.3950e-06 - val_accuracy: 1.0000
Epoch 46/50
50/50 [==============================] - 76s 2s/step - loss: 0.1117 - accuracy: 0.9940 - val_loss: 9.3492e-09 - val_accuracy: 0.9933
Epoch 47/50
50/50 [==============================] - 77s 2s/step - loss: 0.0550 - accuracy: 0.9913 - val_loss: 2.1316e-08 - val_accuracy: 0.9933
Epoch 48/50
50/50 [==============================] - 76s 2s/step - loss: 0.0818 - accuracy: 0.9893 - val_loss: 1.6073e-10 - val_accuracy: 0.9967
Epoch 49/50
50/50 [==============================] - 77s 2s/step - loss: 8.6720e-05 - accuracy: 1.0000 - val_loss: 6.0177e-15 - val_accuracy: 0.9933
Epoch 50/50
50/50 [==============================] - 77s 2s/step - loss: 0.0700 - accuracy: 0.9913 - val_loss: 5.2066e-09 - val_accuracy: 0.9967
[[0.]]
The given leaf is diseased.
```

If we load non infected image, then it will predict leaf as healthy. If we load infected leaf, then it will predict leaf as diseased as well as it will provide preventive measures to be taken.

The software model will also provide agricultural tips according to season.

```
: import datetime

tday=datetime.date.today()

if tday.month>1 and tday.month<6:
    print("Summer Season Agricultural Tips:\n\n    I]Always keep your farm machinery clean and free of any combustible materia
elif tday.month>5 and tday.month<9:
    print("Rainy Season Agricultural Tips:\n\n    I]Keep Farm Produce Away from Rain.\n    II]Make Sure Water Areas are Fence
else:
    print("Winter Season Agricultural Tips:\n\n    I]Block The Wind.\n    II]Provide Dry Shelter.\n    Raise Hardy Breeds")
```

```
Summer Season Agricultural Tips:

    I]Always keep your farm machinery clean and free of any combustible materials. Make sure all the engine compartments are
clean and well maintained.
    II]Exhaust systems such as manifolds, turbochargers and mufflers are free of any possible leaks and are in good working
condition;
    III]Follow both the maintenance schedules and instructions when operating and installing farm machinery
    IV]Replace all the damaged and worn out electrical components
```

IV. CONCLUSION

The software model is designed to take any leaf image as input and predicted whether it is infected or not. During this project we learnt about deep learning concepts like neural networks with extension of convolutional neural network. The program can be further extended to include more functions in this software model.

ACKNOWLEDGMENT

The authors would like to thank the Department of Information Technology of Government college of Engineering, Karad, India for their generous support. The authors would like to thank the Principal Dr. A. T. Pise, HoD Dr. S. J. Wagh and all staff members of Information Technology Department for sharing pearls of wisdom with us during the course of this research, and we thank “anonymous” reviewers for their so-called insights. The authors are also immensely grateful to for their comments on an earlier version of the manuscript, although any errors are our own and should not tarnish the reputations of these esteemed persons.

As author student

Mr. Onkar Umesh Kadam pursuing Bachelor of Engineering degree in Information Technology from Government College of Engineering, Karad in the current academic year 2019-20.

Mr. Sanket Jaysing Bendale pursuing Bachelor of Engineering degree in Information Technology from Government College of Engineering, Karad in the current academic year 2019-20.

Mr. Kshitij Ganesh Bargale pursuing Bachelor of Engineering degree in Information Technology from Government College of Engineering, Karad in the current academic year 2019-20.

As Author Guide

Prof. Niranjan S. Deokule. He is presently working as an assistant professor in Government College of Engineering, Karad, India. He received Bachelor of Engineering degree in Computer Science and Engineering from Marathwada Mitra Mandal College of Engineering, Pune in 2011 and Post Graduation(M. E.) in 2014 from Pune Institute of Computer Technology (PICT), Pune.

REFERENCES

[1] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.3632&rep=rep1&type=pdf

[2] https://www.researchgate.net/publication/318109025_Leaf_Disease_Detection_using_Image_Processing

[3] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.3632&rep=rep1&type=pdf

[4] https://www.ijert.org/research/application-of-image-processing-techniques-in-plantdisease-

[5] recognition-IJERTV4IS030829.pdf

[6] https://www.spiedigitallibrary.org/conference-proceedings-of-spie/0528/0000/Artificial-Intelligence-In-Image-Processing/10.1117/12.946419.short?SSO=1