AI-Driven Pest Detection and Crop Health Monitoring for Small Farmers

Sudhanshu Gupta

Suraj Kumar

1. **Crop Disease Prediction** (Crop Disease Prediction End-to-End):

   In this the person, has worked on identifying plant diseases.

About dataset used:

We use the PlantVillage dataset [1] by Hughes et al. consists of about 87,000 healthy and unhealthy leaf images divided into 38 categories by species and disease. Here we provide a subset of our experiments on working with this data. We also end up transfer learning from MobileNet and use the weights from pre-training on ImageNet.

The 38 classes here represent that dataset is categorised into these 38 categories. Each class contains various images of healthy and unhealthy images of leafs that helps in identifying the plant diseases.

- Apple___Apple_scab
- Apple___Black_rot
- Apple___Cedar_apple_rust
- Apple___healthy
- Blueberry___healthy
- Cherry_(including_sour)___Powdery_mildew
- Cherry_(including_sour)___healthy
- Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot
- Corn_(maize)___Common_rust_
- Corn_(maize)___Northern_Leaf_Blight
- Corn_(maize)___healthy', 'Grape___Black_rot
- Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
- Grape___healthy
- Orange___Haunglongbing_(Citrus_greening)
- Peach___Bacterial_spot
- Peach___healthy
- Pepper,_bell___Bacterial_spot
- Pepper,_bell___healthy
- Potato___Early_blight
- Potato___Late_blight

- Potato___healthy
- Raspberry___healthy
- Soybean___healthy
- Squash___Powdery_mildew
- Strawberry___Leaf_scorch
- Strawberry___healthy
- Tomato___Bacterial_spot
- Tomato___Late_blight
- Tomato___Leaf_Mold
- Tomato___Septoria_leaf_spot
- Tomato___Spider_mites Two-spotted_spider_mite
- Tomato___Target_Spot
- Tomato___Tomato_Yellow_Leaf_Curl_Virus
- Tomato___Tomato_mosaic_virus
- Tomato___healthy

- The model used for training the dataset is MobileNet.
- Created a small upstream model on top of the MobileNet using the functional API

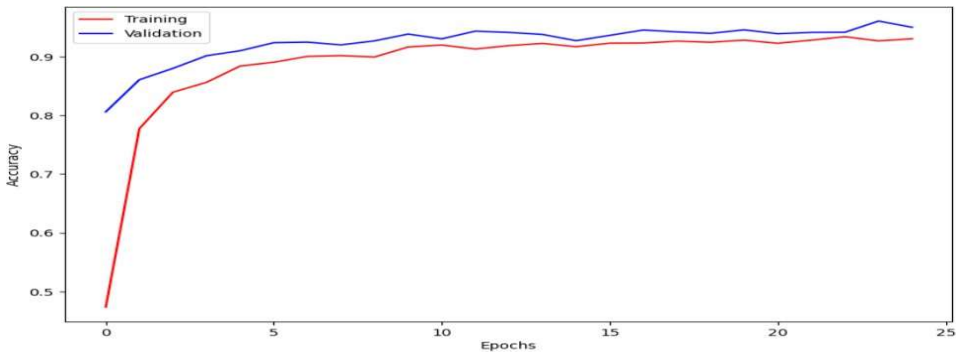
```
inputs = keras.Input(shape = input_shape)

x = base_model(inputs, training = False)
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(len(categories),
                          activation="softmax")(x)

model = keras.Model(inputs = inputs,
                    outputs = x,
                    name="LeafDisease_MobileNet")
```

Also used Adam optimizer that work really well with it's default learning rate, β1, β2 and ϵ values.

Accuracy:



Key Observations:

- **X-axis (Epochs):** The number of epochs (iterations) the model was trained for.

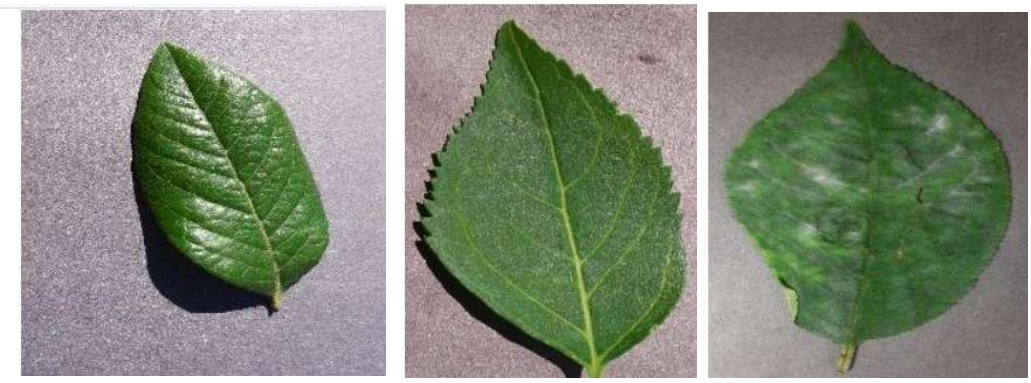
- **Y-axis (Accuracy):** The accuracy of the model.
- **Red Line (Training Accuracy):** Shows how the training accuracy improves over time.
- **Blue Line (Validation Accuracy):** Indicates the accuracy on the validation dataset.

Analysis:

1. **Steady Increase in Accuracy:** Both training and validation accuracy improve as epochs increase.
2. **Gap Between Training and Validation Accuracy:** The validation accuracy is consistently higher than training accuracy. This is unusual and may suggest that:
    - Data augmentation or regularization techniques (e.g., dropout, batch normalization) are helping prevent overfitting.
    - The training set may be more challenging or noisy than the validation set.
3. **Accuracy:** After about 10–15 epochs, accuracy stabilizes, meaning additional training may not significantly improve performance.

2. Plant disease detection(Plant-Disease-Detection/Model/Plant Disease Detection Code.ipynb at main · manthan89-py/Plant-Disease-Detection):

About dataset:



The dataset contain images for different plant leaves that is used for detecting plant diseases. We trained the model using this dataset and then later predict the diseases.

```
length of train size :36584
length of validation size :15679
length of test size :24902
```

Model used:

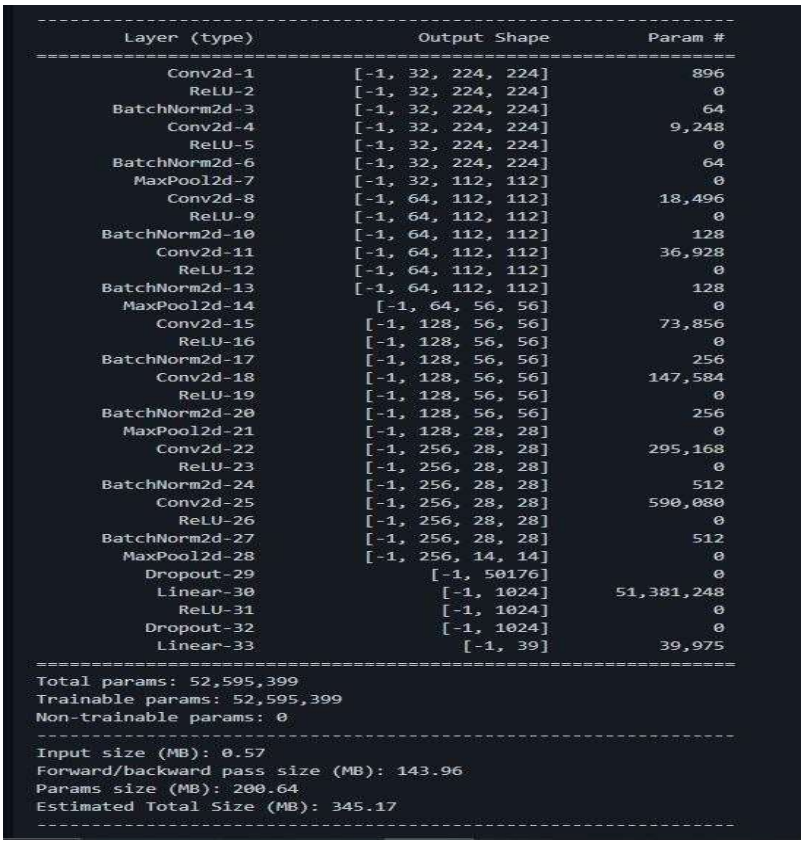
Convolution Aithmetic Equation : (W - F + 2P) / S + 1
W = Input Size
F = Filter Size
P = Padding Size
S = Stride

Your model is a **Convolutional Neural Network (CNN)** implemented using **PyTorch**. It is designed for image classification, particularly for detecting crop diseases based on image inputs.

The model has **four convolutional blocks**, each consisting of:

1. **Two convolutional layers:** Learn spatial hierarchies of features.
2. **ReLU Activation Function:** Introduces non-linearity to enhance learning.
3. **Batch Normalization:** Normalizes activations to improve stability and training speed.
4. **MaxPooling Layer:** Reduces spatial dimensions and retains important features.

Batch Gradient Descent (BGD) for training a **Convolutional Neural Network (CNN)** using PyTorch. It iterates over multiple epochs, computing the training and validation loss while updating the model's weights after processing the entire dataset in each epoch. The function begins by initializing arrays to track training and validation losses. During each epoch, it processes mini-batches from `train_loader`, performing a forward pass to generate predictions, computing the loss using a given criterion, and applying backpropagation to update weights using `optimizer.step()`.



Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 32, 224, 224]	896
ReLU-2	[-1, 32, 224, 224]	0
BatchNorm2d-3	[-1, 32, 224, 224]	64
Conv2d-4	[-1, 32, 224, 224]	9,248
ReLU-5	[-1, 32, 224, 224]	0
BatchNorm2d-6	[-1, 32, 224, 224]	64
MaxPool2d-7	[-1, 32, 112, 112]	0
Conv2d-8	[-1, 64, 112, 112]	18,496
ReLU-9	[-1, 64, 112, 112]	0
BatchNorm2d-10	[-1, 64, 112, 112]	128
Conv2d-11	[-1, 64, 112, 112]	36,928
ReLU-12	[-1, 64, 112, 112]	0
BatchNorm2d-13	[-1, 64, 112, 112]	128
MaxPool2d-14	[-1, 64, 56, 56]	0
Conv2d-15	[-1, 128, 56, 56]	73,856
ReLU-16	[-1, 128, 56, 56]	0
BatchNorm2d-17	[-1, 128, 56, 56]	256
Conv2d-18	[-1, 128, 56, 56]	147,584
ReLU-19	[-1, 128, 56, 56]	0
BatchNorm2d-20	[-1, 128, 56, 56]	256
MaxPool2d-21	[-1, 128, 28, 28]	0
Conv2d-22	[-1, 256, 28, 28]	295,168
ReLU-23	[-1, 256, 28, 28]	0
BatchNorm2d-24	[-1, 256, 28, 28]	512
Conv2d-25	[-1, 256, 28, 28]	590,080
ReLU-26	[-1, 256, 28, 28]	0
BatchNorm2d-27	[-1, 256, 28, 28]	512
MaxPool2d-28	[-1, 256, 14, 14]	0
Dropout-29	[-1, 50176]	0
Linear-30	[-1, 1024]	51,381,248
ReLU-31	[-1, 1024]	0
Dropout-32	[-1, 1024]	0
Linear-33	[-1, 39]	39,975

Total params: 52,595,399
Trainable params: 52,595,399
Non-trainable params: 0

Input size (MB): 0.57
Forward/backward pass size (MB): 143.96
Params size (MB): 200.64
Estimated Total Size (MB): 345.17

Accuracy:

```
Train Accuracy : 96.7
Test Accuracy : 98.9
Validation Accuracy : 98.7
```

3. Plant Disease Detection with Keras and FastAPI

Link : https://github.com/Nneji123/Plant-Disease-Detection-Keras?tab=readme-ov-file

Plant Disease Detection model built with Tensorflow and deployed as an API on Heroku with FastAPI. An end-to-end Machine Learning Project carried out by Group 1 Zummit Africa AI/ML Team to detect disease (Rust, Powdery Mildew) of infected plants.

Table of Contents

Repository File Structure

```
├── app.py # main fastapi app
├── aws.txt # requirements for deploying on aws
├── docker-compose.yml # for running docker compose
├── Dockerfile
├── download.py # script to download model hosted on google drive
├── fastapi_setup # for deploying on aws
├── LICENSE
├── Notebook #Notebook folder
│   └── plant_disease_detection.ipynb
├── README.md
├── requirements.txt
├── Research_Papers
│   └── R_paper1.pdf
├── setup.sh # bash script for deploying on aws ec2
└── test_images # test images for inference
    ├── 8fd27998ae52a4a6.jpg
    ├── 9be41b823d13e3c6.jpg
    └── 9d0f6e60819f9a5a.jpg
```

Problem Statement

In this project, a neural network model was built using Tensorflow. The model detects if a plant is suffering from a disease(Rust or Powdery Mildew). The model was then deployed as an API using the FastAPI framework.

Rusts

Rusts are plant diseases caused by pathogenic fungi of the order Pucciniales (previously known as Uredinales). An estimated 168 rust genera and approximately 7,000 species, more than half of which belong to the genus Puccinia, are currently accepted. Rust fungi are highly specialized plant pathogens with several unique features. Taken as a group, rust fungi are diverse and affect many kinds of plants. However, each species has a very narrow range of hosts and cannot be transmitted to non-host plants. In addition, most rust fungi cannot be grown easily in pure culture. A single species of rust fungi may be able to infect two different plant hosts in different stages of its life cycle, and may produce up to five morphologically and cytologically distinct spore-producing structures viz., spermogonia, aecia, uredinia, telia, and basidia in successive stages of reproduction. Each spore type is very host specific, and can typically infect only one kind of plant. Rust fungi are obligate plant pathogens that only infect living plants. Infections begin when a spore lands on the plant surface, germinates, and invades its host. Infection is limited to plant parts such as leaves, petioles, tender shoots, stem, fruits, etc. Plants with severe rust infection may appear stunted, chlorotic (yellowed), or may display signs of infection such as rust fruiting bodies. Rust fungi grow intracellularly, and make spore-producing fruiting bodies within or, more often, on the surfaces of affected plant parts.

Powdery Mildew

Powdery mildew is a fungal disease that affects a wide range of plants. Powdery mildew diseases are caused by many different species of fungi in the order Erysiphales. Powdery mildew is one of the easier plant diseases to identify, as its symptoms are quite distinctive. Infected plants display white powdery spots on the leaves and stems. The lower leaves are the most affected, but the mildew can appear on any above-ground part of the plant. As the disease progresses, the spots get larger and denser as large numbers of asexual spores are formed, and the mildew may spread up and down the length of the plant. Powdery mildew grows well in environments with high humidity and moderate temperatures. Greenhouses provide an ideal moist, temperate environment for the spread of the disease. This causes harm to agricultural and horticultural practices where powdery mildew may thrive in a greenhouse setting. In an agricultural or horticultural setting, the pathogen can be controlled using chemical methods, bio organic methods, and genetic resistance. It is important to be aware of powdery mildew and its management as the resulting disease can significantly reduce important crop yields.

Project Outline

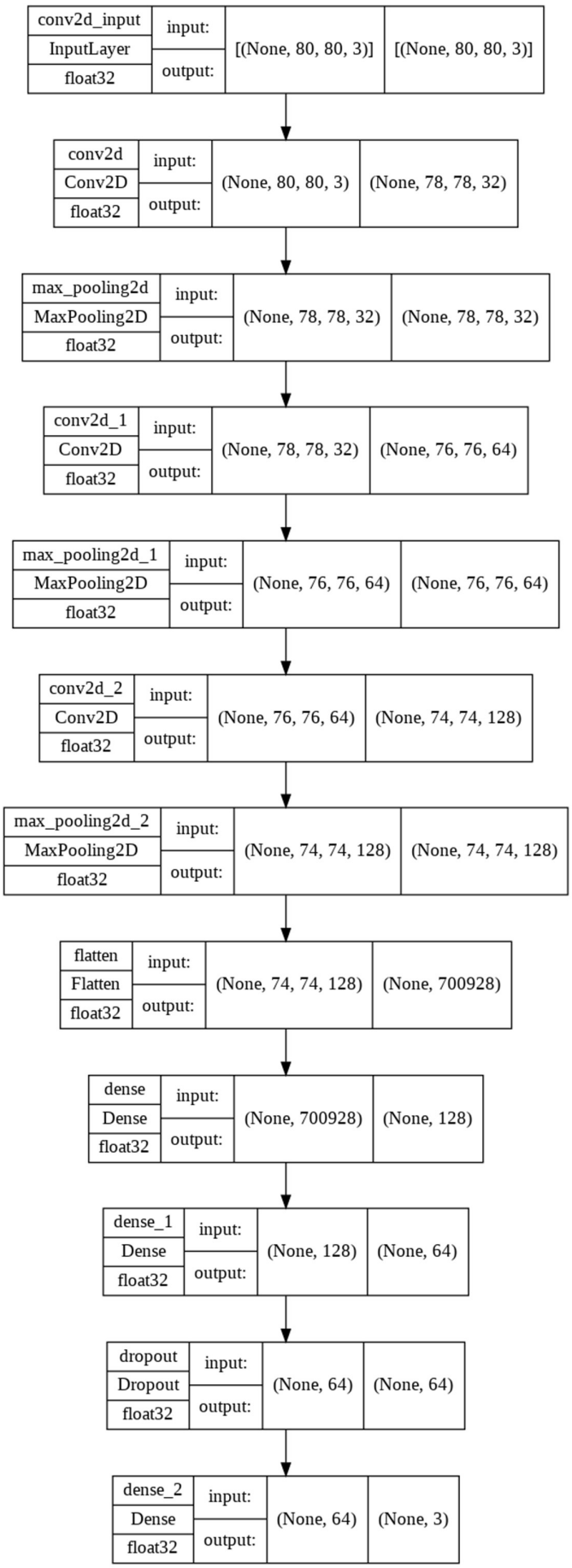
1. Analyse the image data.
2. Build a Neural Network with Keras for classifying the images.
3. Design, build and deploy the model as a FastAPI Application.

Data Preparation
The dataset used to train the neural network contains three labels, "Healthy", "Powdery", "Rust" referring to plant conditions. There is a total of 1530 images divided into train, test, and validation sets.

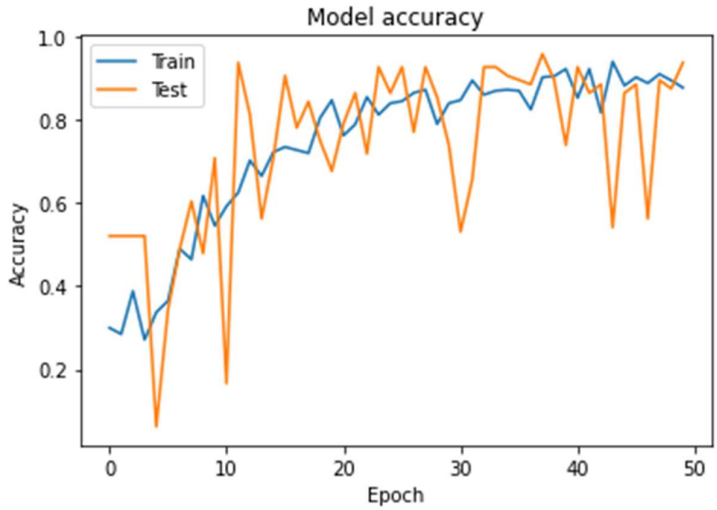
Dataset Link

Model Building
The model was built using Keras(Tensorflow as backend) and was trained using the train images in the dataset and then evaluated using the test images in the dataset. The model was trained using the following parameters:

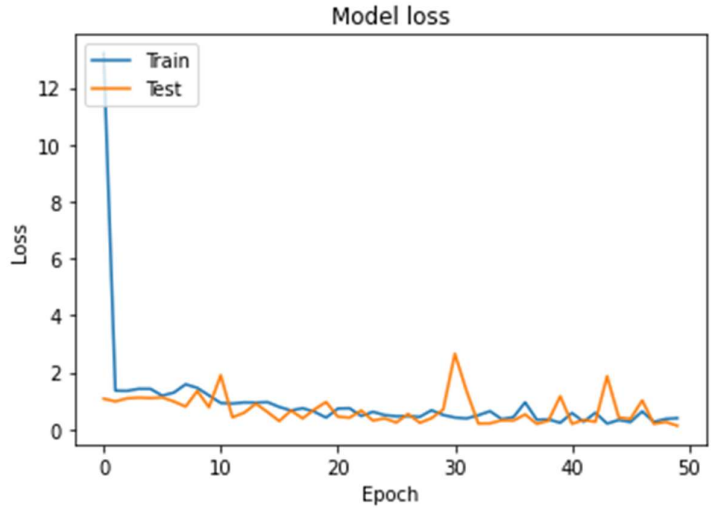
- nb_train_samples = 400
- nb_validation_samples = 100
- nb_eval_samples = 60
- epochs = 50
- batch_size = 16
- num_of_class = 3
- **After training, the model had an accuracy of 93.75%**
- *Model Diagram*


conv2d_input
InputLayer
float32
input:
output:
[(None, 80, 80, 3)]
[(None, 80, 80, 3)]
conv2d
Conv2D
float32
input:
output:
(None, 80, 80, 3)
(None, 78, 78, 32)
max_pooling2d
MaxPooling2D
float32
input:
output:
(None, 78, 78, 32)
(None, 78, 78, 32)
conv2d_1
Conv2D
float32
input:
output:
(None, 78, 78, 32)
(None, 76, 76, 64)
max_pooling2d_1
MaxPooling2D
float32
input:
output:
(None, 76, 76, 64)
(None, 76, 76, 64)
conv2d_2
Conv2D
float32
input:
output:
(None, 76, 76, 64)
(None, 74, 74, 128)
max_pooling2d_2
MaxPooling2D
float32
input:
output:
(None, 74, 74, 128)
(None, 74, 74, 128)
flatten
Flatten
float32
input:
output:
(None, 74, 74, 128)
(None, 700928)
dense
Dense
float32
input:
output:
(None, 700928)
(None, 128)
dense_1
Dense
float32
input:
output:
(None, 128)
(None, 64)
dropout
Dropout
float32
input:
output:
(None, 64)
(None, 64)
dense_2
Dense
float32
input:
output:
(None, 64)
(None, 3)

Model Accuracy



Model Loss



Classification Report

	precision	recall	f1-score	support
Healthy	0.95	0.90	0.92	20
Powdery	0.91	1.00	0.95	20
Rust	1.00	0.95	0.97	20
accuracy			0.95	60
macro avg	0.95	0.95	0.95	60
weighted avg	0.95	0.95	0.95	60

Confusion Matrix